Efficient Boosted Weak Classifiers for Object Detection

Xiaopeng Hong[1], Guoying Zhao[1], Haoyu Ren[2], and Xilin Chen[2,1]

[1] Department of Computer Science and Engineering, University of Oulu, Finland
[2] Key Lab of Intelligent Information Processing of Chinese Academy of Sciences, Institute of Computing Technology, CAS, P.R. China
{xhong,gyzhao}ee.oulu.fi, hyren@jdl.ac.cn, xlchen@ict.ac.cn

Abstract. This paper accelerates boosted nonlinear weak classifiers in boosting framework for object detection. Although conventional nonlinear classifiers are usually more powerful than linear ones, few existing methods integrate them into boosting framework as weak classifiers owing to the highly computational cost. To address this problem, this paper proposes a novel nonlinear weak classifier named *Partition Vector weak Classifier* (PVC), which is based on the histogram intersection kernel functions of the feature vector with respect to a set of pre-defined *Partition Vectors* (PVs). A three-step algorithm is derived from the kernel trick for efficient weak learning. The obtained PVCs are further accelerated via building a look-up table. Experimental results in the detection tasks for multiple classes of objects show that boosted PVCs significantly improves both learning and evaluation efficiency of *nonlinear* SVMs to the level of boosted linear classifiers, without losing any of the high discriminative power.

1 Introduction

Object detection of a particular class, such as pedestrian and car, is an important and challenging task for many computer vision applications [1-6]. While object detection has advanced significantly in the past decades, most of its applications now demand not only high accuracy but also computation efficiency. One common solution is using the sliding window searching scheme, where an object classifier is applied at all possible positions and scales to make decisions on whether the hypothesis patches contain a desirable object or not. However, computational burden of searching all possible candidates is usually non-trivial.

To address this problem, boosting family algorithms in cascade structures which are able to eliminate most of the negative samples very fast, are designed to learnt the object classifier and evaluate each hypothesized object locations [7-14]. Most of the boosting algorithms combine a collection of weak classifiers with different weights to a final strong classifier, though their methods of weighting the training samples as well as the weak classifiers can be different. Generally speaking, there are usually hundreds of weak classifiers for one object classifier to achieve acceptable accuracy. Each of the weak classifiers is effectively built via an aggressive weak learning mechanism which first learns and evaluates hundreds of hypothesized weak classifiers (*e.g.*, 250 and 200 in [8, 11] respectively), and then select the one which best separate

J.-K. Kämäräinen and M. Koskela (Eds.): SCIA 2013, LNCS 7944, pp. 205–214, 2013.
© Springer-Verlag Berlin Heidelberg 2013

the positive and the negative samples. Therefore, the efficiency of weak classifiers largely determines the efficiency of the object classifier, including both the evaluation and the learning (also *a.k.a.* '*training*' in object detection task) procedures.

To ensure high efficiency, there are two common ways: the first one is to restrict the weak classifiers each of which is only connected to a single descriptor extracted from an image block [5, 8-15, 19] (such as the Histogram of Oriented Gradient (HOG) [5, 8, 9]); the second one is to adopt efficient *linear* algorithms, such as the linear Support Vector Machine (SVM) [8], least square [11] and Fisher linear discriminative analysis [9, 13], to learn *linear* weak classifiers from the training samples.

The assumption under *linear* classifiers is that samples are separable by a hyperplane in the feature space. This assumption cannot be valid for some complicated tasks such as human and car detection. Hence it is not surprisingly nonlinear classifiers such as the non-linear kernel SVM has proven to outperform the *linear* ones in object detection [5, 16, 17]. However, to our best knowledge, there are few existing methods utilizing them as the weak classifier in the boosting framework. That is mainly because of the highly computational complexities in both learning and evaluation of traditional nonlinear classifiers.

This paper focuses on how to improve the efficiency of non-linear weak classifiers and make them feasible in the boosting framework. Recent, Maji, *et al.* propose the sparse encoding based SVM, which significantly improves the efficiency of the *nonlinear* additive kernel SVMs [17]. Nevertheless, the sparse encoding taking account of the huge dimensionality problem brings additive computational burdens. Hence, it is shown in [17] that the most efficient implementation is still 5 times lower compared to the linear classifiers.

Partly motivated by the sparse encoding based SVM [17], this paper utilizes the dense encoding scheme [20] rather than the sparse encoding in [17] to further improve the efficiency, and proposes an efficient *nonlinear* weak classifier named the Partition Vector weak Classifier (PVC). PVC as one variants of additive kernel SVMs is a weighted combination of a series of kernel functions of the (input) feature vector with respect to a set of pre-defined vectors, namely the Partition Vectors (PVs).

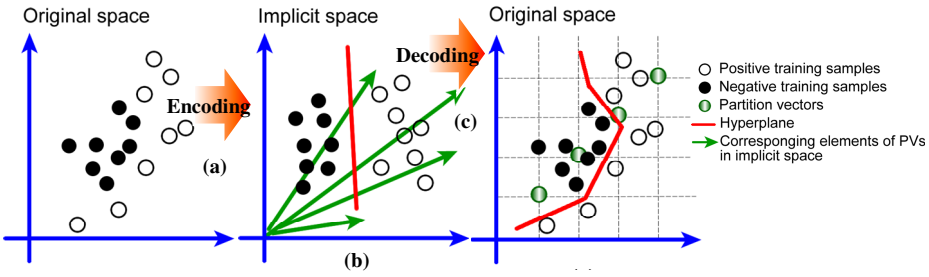


Fig. 1. PVC learning. (a) *Encoding* maps samples in the original space to the implicit space. (b) Learning the hyper-plane using the encoded samples in the implicit space. (c) *Decoding* transforms the learnt hyper-plane to the original space. This figure is better seen in color.

Experiments in object detection of multiple classes show that boosted PVCs makes the weak learning and evaluation of the *nonlinear* weak classifiers efficient enough to be used in the boosting framework, without losing the high discriminative power.

The rest of the paper is structured as follows. Section 2 details the proposed PVC. Section 3 shows the experimental results on INRIA and PASCAL VOC 2007 datasets. And conclusions are provided in the last section.

2 Partition Vector Based Weak Classifier

The learning of the proposed PVCs includes three key steps: *encoding*, *hyper-plane learning*, and *decoding*. Firstly, as shown in Fig. 1 (a), *encoding* maps the vectors in the feature space to the implicit high dimensional space. Secondly, a *linear* SVM is then learnt to get the classification hyper-plane in the implicit space (as the red line in Fig. 1(b)). Finally, *decoding* transforms the obtained classification hyper-plane to a series of kernel functions referring to a set of partition vectors (as the green dots in Fig. 1 (c)) in the feature space, where a *nonlinear* classification function (as the red broken lines in Fig. 1 (c)) is formed. The classifiers can be further accelerated using piecewise constant functions, such that it ensures the computational cost proportional to the dimension of the features during evaluation, as the conventional *linear* classifiers do. The following subsection provides more details.

2.1 Background

For any classifiers based on kernel functions satisfying the Mercer's Condition [20], given two feature vectors $\mathbf{x}, \mathbf{z} \in R^d$, there exists a mapping $\varphi: R^d \to R^D$ whose range is in an inner product space of a possibly high dimensionality D:

$$\mathrm{K}(\mathbf{x}, \mathbf{z}) = \langle \varphi(\mathbf{x}), \varphi(\mathbf{z}) \rangle, \tag{1}$$

where $\langle \cdot \rangle$ denotes the inner product of vectors; the mapping φ usually implies a high dimensional implicit space R^D (i.e., $D \gg d$), where the data are much more sparse and thus more likely to be linearly separable [20]. The kernel trick is a way of mapping observations from the original space S into an implicit inner product space I (equipped with its natural norm), without having to compute the mapping explicitly. For clarity, the points in the feature space and the *implicit* space are denoted by bold lowercase letters (*e.g.*, $\mathbf{x}$) and bold lowercase letters with a hat (*e.g.*, $\hat{\mathbf{x}}$) respectively.

The histogram intersection kernel is one of the most commonly used additive kernels [17]. Specifically, HIK between $\mathbf{x}$ and $\mathbf{z}$ is defined as follows:

$$K_{HI}(\mathbf{x}, \mathbf{z}) = \sum_{i=1}^{d} \min(x_i, z_i), \tag{2}$$

where x_i and z_i are the i-th entry of $\mathbf{x}$ and $\mathbf{z}$ respectively for $i = 1, \cdots, d$. Recent study reveals that the nonlinear mapping φ of HIK could be explicitly defined if each entry of input vectors is non-negative, discrete and bounded [17, 20]. In consideration that firstly, this condition can be easily satisfied by using histogram based descriptor

(*e.g.*, the Histogram of Oriented Gradient (HOG) [5, 8, 9]) or performing feature scaling and quantization[1]; secondly, differences between different additive kernels tend to be trivial on large training sets [17]; thirdly, the calculation of some other additive kernels could also be implemented through HIK, this paper only focuses on HIK in the following context, without losing any generalization.

2.2 Encoding

Encoding maps the feature vectors to the corresponding implicit space. The goal is to find a mapping φ for HIK such that $K_{HI}(\mathbf{x},\mathbf{z}) = \langle\varphi(\mathbf{x}),\varphi(\mathbf{z})\rangle$, given the feature vector $\mathbf{x}$ and $\mathbf{z}$ whose entry $x_i$and $z_i, i = 1,\cdots,d$ are integrals belong to $[0,\cdots,n], n \in \mathrm{Z}$.

Considering that HIK is additive, φ can be divided into separated parts, each of which corresponds to an entry of the feature vector. In specific, $\varphi_i: \mathrm{Z}^1 \to \{1,0\}^n$ for $x_i,\ i = 1,\cdots,d$, is defined as follows [20]:

$$\varphi_i(x_i) = \left[\mathbf{1}_{1\times x_i}, \mathbf{0}_{1\times(\mathrm{n}-x_i)}\right], \tag{3}$$

where $\mathbf{1}_{1\times x_i}$ and $\mathbf{0}_{1\times(\mathrm{n}-x_i)}$ are a $1\times x_i$ row vector of 1 and a $1\times(\mathrm{n}-x_i)$ vector of 0 respectively. φ_i is thus a binary representation of the x_i, $0 \le x_i \le \mathrm{n}$. For example, given two integers 3 and 5, supposed n is 6, we have $\varphi_i(3) = [1,1,1,0,0,0]$ and $\varphi_i(5) = [1,1,1,1,1,0]$, $\min(3,5) = \langle\varphi_i(3),\varphi_i(5)\rangle = 3$ hence Eq. (1) is satisfied. When covering all dimensions, we have $\hat{\mathbf{x}} = \varphi(\mathbf{x}) = [\varphi_1(x_1),\cdots,\varphi_n(x_n)]^T$.

Having φ, all d dimensional vectors for learning the classifier (also as known as the training samples) are mapped to the $D = nd$ dimensional space, where a hyper-plane (i.e., a *linear* classification function) is then learnt. In specific, for N labeled training samples $\hat{\mathbf{x}}^k \in \{1,0\}^{nd}$ with the labels $\mathrm{y}^k \in \{-1,+1\}$, the learning procedure involves finding the *best* hyper-plane $\hat{\mathrm{f}}$ (i.e., the maximum-margin hyper plane learnt by efficient linear SVM in this paper), which can be express as follows:

$$\hat{\mathrm{f}}(\hat{\mathbf{x}}) = \langle\widehat{\mathbf{w}},\hat{\mathbf{x}}\rangle, \tag{4}$$

where $\widehat{\mathbf{w}} = [\widehat{\mathbf{w}}_1\cdots\widehat{\mathbf{w}}_d]^T \in R^{nd}$ is the obtained hyper-plane in the form of a column vector; and $\widehat{\mathbf{w}}_i$ is the corresponding part (in the form of a row vector) related to φ_i as well as the i-th entry of the feature vectors[2] for $i = 1,\cdots,d$.

2.3 Decoding

As the inverse of *encoding*, *decoding* derives f in the feature space, the counterpart of the function $\hat{\mathrm{f}}$ to satisfy $\mathrm{f}(\mathbf{x}) = \hat{\mathrm{f}}(\hat{\mathbf{x}})$.

Considering Eq. (1) and (4), the mapping φ may not be invertible, hence it is difficult, and perhaps impossible at all to find such a vector $\mathbf{w}$ in the feature space that

[1] Though quantizing feature vectors to integral value may lead to approximation errors, we observe that they could be minimized by averaging over weak learners during learning the boosted classifiers. Consistent observation can also be found in [11].

[2] The bias usually used in describing a linear function is omitted in this paper for clarity.

$\varphi(\mathbf{w}) = \hat{\mathbf{w}}$. To address this problem, this paper loses this constraint so that there can be a set of vectors rather than only one vector in the feature space. As a result, $\hat{\mathrm{f}}$ in Eq. (4) can be expressed as a weighted combination of kernel functions with respect to those vectors. Actually, this paper realized that this set of vectors can be pre-defined and the combination weights can be efficiently solved as follows.

In specific, we define the set of Partition Vectors (PVs) P as a set of n vectors (recall that n is the upper bound of all entries of the input vectors). Each PV has all its entries equivalent to a unified value, *i.e.*, $P = \{\mathbf{p}_j | \mathbf{p}_j = \mathbf{j}_{d\times 1}, j = 1, \cdots, n\}$. The goal of *decoding* is to find a $n \times d$ matrix of coefficients $\mathbf{A}$ satisfying

$$\mathrm{f}(\mathbf{x}) = \hat{\mathrm{f}}(\hat{\mathbf{x}}) = \mathrm{trace}(\mathbf{AM}), \tag{5}$$

where the j-th row of $\mathbf{A}$ is a vector of weights $\mathbf{a}_j = [a_{j1}, \cdots, a_{jd}]$ and the j-th column of the $d \times n$ matrix $\mathbf{M}$ is the output of the per-entry kernel operation given $\mathbf{x}$ and the j-th PV $\mathbf{p}_j$ as the input, i.e., $\mathbf{m}_j = [\min(x_1, j), \cdots, \min(x_d, j)]^T$ for $i = 1, \cdots, n$.

For the nd dimensional hyper-plane $\hat{\mathbf{w}} = [\hat{\mathbf{w}}_1 \cdots \hat{\mathbf{w}}_d]^T$ in Eq. (4), its i-th part $\hat{\mathbf{w}}_i = [w_{i1} \cdots w_{in}], i = 1, \cdots, d$, is a vector in R^n. Hence $\hat{\mathbf{w}}_i$ can be represented by a set of n bases in R^n which are defined as $\{\boldsymbol{\gamma}_j | \boldsymbol{\gamma}_j = [\mathbf{1}_{1\times j}, \mathbf{0}_{1\times(n-j)}],\ j = 1, \cdots, n\}$:

$$\hat{\mathbf{w}}_i = \sum_{j=1}^{n} a_{ji}\boldsymbol{\gamma}_j, i = 1, \cdots, d. \tag{6}$$

The coefficient a_{ji} which forms the matrix of $\mathbf{A}$ can be solved through Gaussian elimination. The proofs of Eq. (5) are given as follows:

According to φ in Eq. (3), it can be easily found that $\varphi_i(j) = \boldsymbol{\gamma}_j$ and $\hat{\mathbf{x}} = \varphi(\mathbf{x}) = [\varphi_1(x_1), \cdots, \varphi_n(x_n)]^T$. Substitute Eq. (6) into (4), we have

$$\hat{\mathrm{f}}(\hat{\mathbf{x}}) = \langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle = \sum_{i=1}^{d}\sum_{j=1}^{n} \langle a_{ji}\boldsymbol{\gamma}_j, \varphi_i(x_i) \rangle = \sum_{j=1}^{n}\sum_{i=1}^{d} a_{ji} \langle \boldsymbol{\gamma}_j, \varphi_i(x_i) \rangle. \tag{7}$$

Considering the kernel trick in Eq. (1), Eq. (7) becomes:

$$\hat{\mathrm{f}}(\hat{\mathbf{x}}) = \sum_{j=1}^{n}\sum_{i=1}^{d} a_{ji} \min(x_i, j) = \mathrm{trace}(\mathbf{AM}), \tag{8}$$

Here we verify Eq. (5).

Input: training samples $\{\mathbf{x}^k, \mathrm{y}^k\}_{k=1,\cdots,N}$ with the labels $\mathrm{y}^k \in \{-1, +1\}$
1. Encoding
  1.1 Map all vectors $\mathbf{x}$ to R^{nd} using Eq. (3)
2. Train a linear SVM $\hat{\mathbf{w}}$ in R^{nd}
3. Decoding
  3.1 Calculate the weights matrix $\mathbf{A}$ of the PVs according to Eq. (6)
  3.2 Get the classification function $\mathrm{f}(\mathbf{x})$ using Eq. (5)
  3.3 Build a look-up table for $\mathrm{f}(\mathbf{x})$
Output: PVC in the form of $\mathrm{f}(\mathbf{x}) = \sum_{i=1}^{d} LUT_i(x_i)$

Fig. 2. Algorithm of PVC learning

Eq. (5) transforms the inner product in the implicit space in Eq. (4) to a *nonlinear* function referring to pre-defined PVs $\boldsymbol{\gamma}_j$ in the feature space. More important, since $x_i \in [0, \cdots, n]$, f in Eq. (5) can be further accelerated by building d piecewise constant functions for each entry, i.e, $\mathrm{f}(\mathbf{x}) = \sum_{i=1}^{d} LUT_i(x_i)$, such that the computational complexity in evaluation is reduced to $O(d)$, exactly the same to that of linear classifiers.

2.4 Summary

Fig. 2 summarizes the learning of PVC. Benefit from *encoding*, nonlinear classification in the feature space can be achieved by the efficient *linear* classification in the implicit space. The hyper-plane learnt in the implicit space ensures the discriminative power of PVCs. Finally, *decoding* contributes to the efficiency, accelerating the detection speed of PVCs to the level of linear classifiers.

In specific, the computational complexity of *encoding* is $O(Nd)$, linear in the number of training samples N and the dimensionality of vectors d. Besides, the learning of hyper-plane in implicit space can be done using the extremely fast SVM toolkit LIBLINEAR [22]. Likewise, the computational complexity of evaluation of an input vector is $O(d)$, exactly the same as linear classifiers. Therefore, both the learning and detection of PVCs are efficient.

PVCs are a kind of weak classifiers independent of any boosting algorithm [7-13]. As a result, this paper simply uses the RealBoost as an example in the following experiments to evaluation the proposed PVCs.

3 Experiments

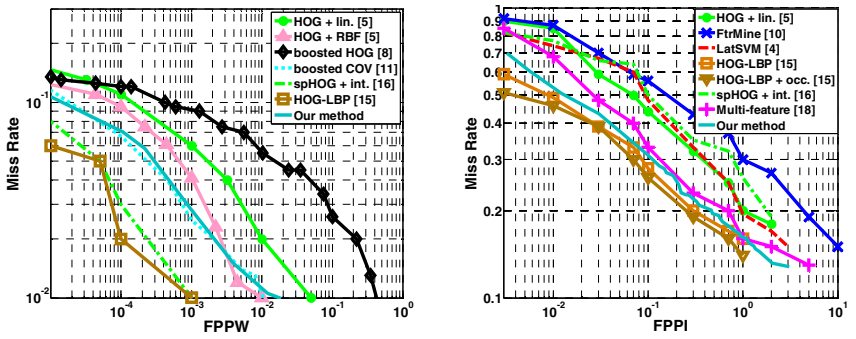
This section describes experiments in detecting multiple objects, including pedestrians, cars, motorbikes, dogs, and cows to evaluate the proposed PVCs. All experiments are evaluated on desktops with Intel 3.0GHz CPU and 4GB memory.

This paper utilizes the multi-size HOG features [5, 8, 9] as an examples to train a cascade classifier considering its great success in the literature. Totally 4496 36d HOGs of block sizes from 16×16 to 48×96 are generated as the candidates feature pool. 60 out of the 4,496 features are randomly selected and evaluated in each round. More details of RealBoost can be found in the papers [12, 13].

3.1 Pedestrian Detection on the INRIA Dataset

We evaluate PVCs in pedestrian detection using the commonly used INRIA dataset [5] and follow the same experimental settings in [8].

Firstly, we carry out experiments to evaluate our method with respect to two criteria, i.e., the miss rate tradeoff False Positive rate Per Window (FPPW) and the detection rate versus False Positive rate Per Image (FPPI) respectively.



(a) (b)

Fig. 3. Evaluation on INRIA dataset with respect to (a) FPPW and (b) FPPI criteria

Fig. 3 (a) illustrates the results with respect to FPPW. Using the same HOG features, our method reaches a similar accuracy as the RBF kernel SVM [5], and reduces the miss rates by 4 percentages at 10e-4 FPPW compared to the linear SVM [5] and the boosted linear ones [8]. Our result is inferior to [15] and [16] since 1) [15] utilizes composite and thus more powerful descriptor combining HOG and other features; 2) boundary effects were reported on the implementation of [16] and its exact performance is actually similar to the one achieved by RBF kernel SVM [1]. Fig. 3 (b) shows the results under the FPPI testing. Our method achieves competitive results compared with most of the algorithms. Using the HOG features, our result achieves by at least 4 percentages improvement on the detection rate compared to the linear SVM [5] and IKSVM [16].

Fig. 3 shows that using the proposed PVC as weak classifiers in boosting framework is effective. Several detection examples in difficult images including clutter backgrounds and the pedestrians with variant appearances are shown in Fig. 4.

Secondly, we compare the learning and evaluation speeds[3] of PVCs with other *linear* weak classifiers including linear SVM, linear least square, and the *nonlinear* fast IKSVM [17]. Both PVCs and linear SVMs are trained using the **LIBLINEAR** toolkit [22]. Table 1 shows that both the training and detection speeds of PVCs are fasted among all weak classifiers compared. The efficiency improvement brought by PVCs is analyzed as follows.

Regardless the time consumption of the bootstrapping, the computational costs of a boosted classifier mainly depend on two factors: the training time of a single weak classifier and the weak classifier number. We observe that using the efficient LIBLINEAR package, the difference between the training time of a single PVC and a single conventional *linear* weak classifier is trivial. We also realized that the boosted PVCs as well as the boosted fast IKSVMs uses only one quarter number of weak classifiers (about 450) of that used by the boosted *linear* ones (about 1700) to achieve a false positive rate of 10^{-4} on the training data. Therefore, training PVCs is more efficient than other *linear* weak classifiers.

[3] Generally, boosted classifiers are more efficient than SVM classifiers in detection due to the cascade framework. Hence we only compare our method with the boosted classifiers in the speed evaluation here.

Table 1. Speed comparison between PVC and other commonly used weak classifiers

Weak classifier	Training time until 10e-4 FP	Patches processed per second
Linear SVM	>24 hours	~90,000
Linear Least Square	~24 hours	~90,000
fast IKSVM [17]	~10 hours	~20,000
PVC	**~6 hours**	**~100,000**

Table 2. Comparisons on PASCAL VOC 2007 dataset with respect to Average Precision (AP)

	Car	Motorbike	Bicycle	Dog	Cow
Oxford [23]	**0.432**	0.375	0.409	---	---
UoCTTI [[23]	0.346	0.276	0.369	0.023	0.140
Our method	0.418	**0.415**	**0.488**	**0.191**	**0.189**

The evaluation time of a boosted classifier depends on two factors as well: the average number of passed weak classifiers for one candidate sample and the computational cost of a single weak classifier. We calculate the average number of weak classifiers used to make a decision on two millions samples for FPPW evaluation on INRIA dataset. The boosted classifiers based on PVCs and fast IKSVMs use about 21 weak classifiers in average, while the linear versions use about 27. Because of this, although the computational cost of single PVC is the same as that of linear weak classifier, the evaluation speed of the boosted PVCs is still faster than the *linear* ones.

It is also worth to mention that our method still achieves better training and detection speed than the boosted fast IKSVMs [17], since PVC uses dense encoding and avoid additional costs in sparse encoding as in [17].

3.2 Car, Dog, and Cow Detection on PASCAL VOC 2007 Dataset

In this section, we evaluate the PVC on five categories of PASCAL VOC 2007 dataset, *car*, *bicycle*, *motorbike*, *dog* and *cow*. We follow the training and detection protocols [23]. The positive training samples with 'hard' label, strong occlusion (more than 50% of the car is occluded) or weak illumination are dropped. Table 2 compares the average precision with two baseline algorithms [23], including the object detector proposed by Chum, *et al*., and Felzenszwalb, *et al*., as abbreviated as 'Oxford' and 'UoCTTI' in Table 2. Both of them use HOG-like features and SVM-like *nonlinear* classifiers. We can observe that our method achieves the best results in the competition on *motorbike*, *bicycle*, *dog* and *cow*; and the second best result in *car* detection. So we can also safely conclude that the boosted PVCs is effective for object detection. Several detection examples are illustrated in Fig. 4.

Fig. 4. Illustration of detection examples. The first three columns, the fourth and fifth columns, the first two rows of the last column, and the last row of the last column show the detection results of pedestrian detection on the INRIA dataset, and the results of car, bike, dog, and cow detection on the PASCAL VOC 2007 dataset respectively.

4 Conclusion

This paper shows that *nonlinear* classifiers can be efficiently applied in the boosting framework by proposing a novel *nonlinear* weak classifier named PVC. PVC, in the form of a look-up table is efficiently built by learning a classification hyper-plane in the implicit space and further feeds the hyper-plane back to the feature space. PVC has high efficiency as the linear classifiers and good discriminative power as the non-linear classifiers. The experimental results on the INRIA pedestrian dataset and the PASCAL VOC 2007 dataset show that boosted PVCs significantly accelerates the learning and detection speeds of the boosted *nonlinear* classifiers, while maintaining their accuracy.

The algorithm proposed in this paper is promising to be further studied. We will generalize it to other image descriptors such as the covariance matrix [11] and extend it to other computer vision applications.

Acknowledgement. The authors appreciate the supports of the Academy of Finland, Infotech Oulu and the FiDiPro program of Tekes. Dr. Xilin Chen is also partially supported by the Natural Science Foundation of China under contract No. 61025010.

References

1. Dollár, P., Wojek, C., Schiele, B., Perona, P.: Pedestrian Detection: An Evaluation of the State of the Art. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(4), 743–761 (2012)
2. Enzweiler, M., Gavrila, D.: Monocular pedestrian detection: Survey and experiments. IEEE Transactions on Patten Analysis and Machine Intelligence 31(12), 2179–2195 (2009)
3. Gall, J., Yao, A., Razavi, N., Van, G.L., Lempitsky, V.: Hough Forests for Object Detection, Tracking, and Action Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence 33(11), 2188–2202 (2011)

4. Felzenszwalb, P., Girshick, R., McAllester, D., Ramanan, D.: Object Detection with Discriminatively Trained Part Based Models. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(9), 1627–1645 (2010)
5. Dalal, N., Triggs, B.: Histograms of Oriented Gradients for Human Detection. In: CVPR (2005)
6. Vedaldi, A., Gulshan, V., Varma, M., Zisserman, A.: Multiple Kernels for Object Detection. In: ICCV (2009)
7. Viola, P., Jones, M.: Rapid Object Detection Using a Boosted Cascade of Simple Features. In: CVPR (2001)
8. Zhu, Q., Avidan, S., Yeh, M., Cheng, K.: Fast Human Detection Using a Cascade of Histograms of Oriented Gradients. In: CVPR (2006)
9. Laptev, I.: Improvements of Object Detection Using Boosted Histograms. In: BMVC (2006)
10. Dollár, P., Tu, Z., Tao, H., Belongie, S.: Feature Mining for Image Classification. In: CVPR (2007)
11. Tuzel, O., Porikli, F., Meer, P.: Pedestrian Detection via Classification on Riemannian Manifolds. In: PAMI, vol. 30, pp. 1713–1727 (2008)
12. Wu, B., Nevatia, R.: Cluster Boosted Tree Classifier for Multi-View, Multi-Pose Object Detection. In: ICCV (2007)
13. Wu, B., Nevatia, R.: Optimizing Discrimination- Efficiency Tradeoff in Integrating Heterogeneous Local Features for Object Detection. In: CVPR (2008)
14. Zhang, J., Huang, K., Yu, Y., Tan, T.: Boosted local structured HOG-LBP for object localization. In: CVPR (2011)
15. Wang, X., Han, X., Yan, S.: An HOG-LBP Human Detector with Partial Occlusion Handling. In: ICCV (2009)
16. Maji, S., Berg, A., Malik, J.: Classification Using Intersection Kernel Support Vector Machines is Efficient. In: CVPR (2008)
17. Maji, S., Berg, A.: Max-Margin Additive Classifiers for Detection. In: ICCV (2009)
18. Wojek, C., Schiele, B.: A Performance Evaluation of Single and Multi-feature People Detection. In: Rigoll, G. (ed.) DAGM 2008. LNCS, vol. 5096, pp. 82–91. Springer, Heidelberg (2008)
19. Wojek, C., Walk, S., Schiele, B.: Multi-Cue Onboard Pedestrian Detection. In: CVPR (2009)
20. Yang, M.-H., Roth, D., Ahuja, N.: A Tale of Two Classifiers: Snow vs. SVM in Visual Recognition. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 685–699. Springer, Heidelberg (2002)
21. Schapire, R., Singer, Y.: Improved Boosting Algorithms Using Confidence-rated Predictions. In: Machine Learning, vol. 37, pp. 297–336 (1999)
22. Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9, 1871–1874 (2008)
23. http://pascallin.ecs.soton.ac.uk/challenges/VOC/voc2007/