

Online Learning Asymmetric Boosted Classifiers for Object Detection

Minh-Tri Pham Tat-Jen Cham

School of Computer Engineering
Nanyang Technological University
Singapore

{mtpham, astjcham}@ntu.edu.sg

Abstract

We present an integrated framework for learning asymmetric boosted classifiers and online learning to address the problem of online learning asymmetric boosted classifiers, which is applicable to object detection problems. In particular, our method seeks to balance the skewness of the labels presented to the weak classifiers, allowing them to be trained more equally. In online learning, we introduce an extra constraint when propagating the weights of the data points from one weak classifier to another, allowing the algorithm to converge faster. In compared with the Online Boosting algorithm recently applied to object detection problems, we observed about 0-10% increase in accuracy, and about 5-30% gain in learning speed.

1. Introduction

AdaBoost [3, 14] has been successfully used in many machine learning and pattern recognition tasks in computer vision. Its underlying idea is to combine an ensemble of M weak learners to produce the final classifier with very high accuracy. In the tasks of object detection and recognition, especially for face detection, impressive results [8, 10, 15, 16] have been reported when cascading or hierarchically organizing such boosted classifiers from coarse to fine, achieving a final classifier with high detection rates and fast detection speed.

Recently, there has been considerable interest in applying boosting techniques on computer vision problems that require online learning. Examples are: online selection of discriminative features of Grabner and Bischof [5], online conservative learning of Roth *et al.* [13], and online co-training of Javed *et al.* [9]. These methods use the same underlying Online Boosting algorithm proposed by Oza [12] to learn online. The algorithm minimizes the classification error while updating the weak classifiers online.

However, in the tasks of object detection and recognition (e.g., face detection or database retrieval), one often faces with a binary classification problem where the probability of observing the positive examples (*i.e.* the target object) is much lower than the probability of observing the negative examples (e.g. background, non-target object). In such domains, an asymmetric classifier that can avoid the danger of missing positive patterns is often more desirable than the one minimizing the classification error. And then, high detection rates can be achieved by cascading the classifiers (e.g. [15, 16]) or hierarchically organizing them (e.g. [8, 10]) from coarse to fine.

There have been works addressing the problem of learning asymmetric classifiers [1, 7, 11, 16]. Though being differed in details, the common approach of these methods is to setup an asymmetric expected loss where false negatives are penalized more than false positives. Viola and Jones [16] introduced an asymmetric loss: false negatives costs k times more than false positives. They applied an asymmetric re-weighting technique before training each weak classifier so that the result is equivalent to minimizing their asymmetric loss. Ma and Ding [11] applied cost-sensitive learning technique [1] to face detection with re-weighting scheme similar to that of Viola and Jones [16]. However, the weight-updating rule [1] involves a parameter c which is application-dependent and needs extensive trials for best performance. Hou *et al.* [7] proposed an interesting adaptive technique avoiding the need for application-dependent parameters, but their method requires re-training the weak classifiers multiple times to fine-tune their asymmetric parameter a_t . To our best knowledge, their method could not be done online.

In this paper, we introduce a novel boosting algorithm which addresses the problem of learning *online* an *asymmetric* boosted classifier. The idea of learning *online* is similar to the Online Boosting algorithm [12] having been recently used [5, 9, 13], but with a faster convergence speed. On the aspect of learning *asymmetrically*, our method is

similar to that of Viola and Jones [16]. Besides, our method also seeks to balance the *skewness* of labels presented to each weak classifiers (to be discussed in the paper), so that they are trained more equally.

The remaining parts of the paper are organized as follows. In Section 2, we introduce our algorithm for online learning a boosted asymmetric classifier, providing analysis and drawing relations to the offline case. Section 3 gives the experimental results when comparing our method with other methods. We present conclusions in Section 4.

2. Online Asymmetric Discrete AdaBoost

Suppose data come in online, and at iteration N , we observe a new training data point $\mathbf{x}_N \in \mathbb{R}^d$ and its corresponding label $y_N \in \{-1, 1\}$. Because the distribution of labels is highly skewed, we assume $P(y = 1) \ll P(y = -1)$. We wish to learn online a model $C(\mathbf{x})$ to predict the label of an unknown point \mathbf{x} . The model of interest is an ensemble of M weak classifiers $f_m(\mathbf{x}) \in \{-1, 1\}$ related by:

$$C(\mathbf{x}) = \text{sign}(F_M(\mathbf{x})) \quad (1)$$

$$F_M(\mathbf{x}) = \sum_{m=1}^M c_m f_m(\mathbf{x}), \quad (2)$$

where c_m is the voting weight associated with the m -th weak classifier $f_m(\mathbf{x})$ [3].

Our goal is to learn *online* $(c_m, f_m(\mathbf{x}))$ for all $m = 1, 2, \dots, M$ so as to minimize an expected loss $\epsilon(C)$ of wrongly predicting the labels. In this paper, we consider the loss proposed by Viola and Jones in [16] for asymmetric classifiers:

$$\epsilon(C) \stackrel{\text{def}}{=} E[ALoss(C(\mathbf{x}), y)], \quad (3)$$

where

$$ALoss(C(\mathbf{x}), y) \stackrel{\text{def}}{=} \begin{cases} \sqrt{k} & \text{if } y = 1 \text{ and } C(\mathbf{x}) = -1 \\ 1/\sqrt{k} & \text{if } y = -1 \text{ and } C(\mathbf{x}) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

and k is a parameter which tells how many times we penalize false negatives more than false positives.

Like other boosting methods, instead of minimizing the expected asymmetric loss $\epsilon(C)$ directly, we seek to minimize an upper bound (see [16] for more details):

$$J(F_M) \stackrel{\text{def}}{=} E[\sqrt{k}y e^{-yF_M(\mathbf{x})}] \geq \epsilon(C). \quad (5)$$

However, unlike other methods, while minimizing this bound, we also seek to balance the *skewness* of the label distribution presented to each of the weak classifier, allowing them to be trained more equally and thereby resulting in a better performance. This can be achieved by carefully controlling the asymmetric weight for each weak classifier.

Algorithm 1 Online Asymmetric Discrete AdaBoost

Require: k

// Initial conditions:

Set $v_m^{tp} = v_m^{tn} = v_m^{fp} = v_m^{fn} = 0, m = 1, 2, \dots, M$.

Set $\gamma_m = 0, m = 1, 2, \dots, M$.

for each new example (\mathbf{x}_N, y_N) **do**

Set k_m according to equation (18) (see section 2.2).

Set $v \leftarrow 1$.

for $m = 1$ to M **do**

// Fit $f_m(\mathbf{x})$:

Set $w \leftarrow v\sqrt{k_m y_N}$.

Learn $f_m(\mathbf{x})$ incrementally using new example (\mathbf{x}_N, y_N) with weight w .

// Propagate the weights (see section 2.3):

Re-run $f_m(\mathbf{x}_N)$ and compare it with y_N .

if true positive **then**

Set $v_m^{tp} \leftarrow v_m^{tp} + v$.

Set $a_m \leftarrow \frac{v_m^{tp} + v_m^{fp}}{v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn}}$.

Set $v \leftarrow v \frac{a_m N / 2}{v_m^{tp}}$

else if false negative **then**

Set $v_m^{fn} \leftarrow v_m^{fn} + v$.

Set $a_m \leftarrow \frac{v_m^{tp} + v_m^{fp}}{v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn}}$.

Set $v \leftarrow v \frac{(1 - a_m) N / 2}{v_m^{fn}}$

else if false positive **then**

Set $v_m^{fp} \leftarrow v_m^{fp} + v$.

Set $a_m \leftarrow \frac{v_m^{tp} + v_m^{fp}}{v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn}}$.

Set $v \leftarrow v \frac{a_m N / 2}{v_m^{fp}}$

else

Set $v_m^{tn} \leftarrow v_m^{tn} + v$.

Set $a_m \leftarrow \frac{v_m^{tp} + v_m^{fp}}{v_m^{tp} + v_m^{tn} + v_m^{fp} + v_m^{fn}}$.

Set $v \leftarrow v \frac{(1 - a_m) N / 2}{v_m^{tn}}$

end if

Set $\gamma_m \leftarrow \log \frac{1 - a_m}{a_m}$.

Set e_m according to equation (25).

Set $c_m \leftarrow \frac{1}{2} \log \frac{1 - e_m}{e_m}$.

end for

end for

Output the classifier: $\text{sign}(\sum_{m=1}^M c_m f_m(\mathbf{x}))$.

2.1. Why equal skewness?

To see why this is the case, let us first define the term “skewness” as follows.

Definition The skewness of a binary distribution $P(y), y \in \{-1, 1\}$ is given by

$$\gamma \stackrel{\text{def}}{=} \log \frac{P(y = -1)}{P(y = 1)}. \quad (6)$$

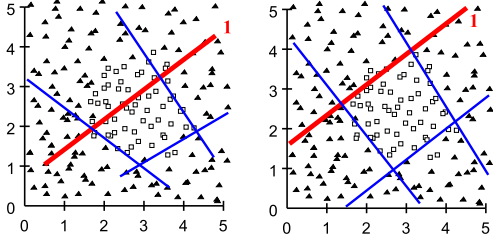


Figure 1. A case where examples are highly skewed $\gamma_1 \gg 1$: Positives are squares, negatives are triangles. Suppose we train with four weak classifiers, which are linear separators. The first classifier is labeled as '1'. On the left is the result trained by Viola and Jones' method: while the subsequent weak classifiers are well-modeled the first classifier is highly affected by γ_1 , resulting in rejecting a significant proportion of positives. On the right is the result trained by our method. Most of the positives are preserved.

Let us consider Viola and Jones' reweighting technique. Suppose the distribution of labels initially have skewness γ_1 . It is easy to verify that, by multiplying the weights with $\sqrt[2M]{k^{y_n}}$, the new skewness is given by (see Proposition A.5 for the proof):

$$\gamma'_1 = \gamma_1 - \frac{1}{M} \log k. \quad (7)$$

However, for subsequent weak classifiers, because of the balanced re-weighting scheme of AdaBoost, the total weights of positive examples and negative examples after being updated *are equal*; which means $\gamma_m = 0$ for all $m > 1$. Therefore, when the updated weights are multiplied with $\sqrt[2M]{k^{y_n}}$, the skewness of the labels becomes

$$\gamma'_m = -\frac{1}{M} \log k. \quad (8)$$

So while subsequent weak classifiers *may* be trained equally, training the first classifier is largely affected by γ_1 . The result is that the first classifier could be trained very wrongly, due to the high skewness of the labels of the examples, as illustrated in (figure 1). By ensuring all weak classifiers to be trained with equal skewness, we expect their decision boundaries after training will have equal effects. Therefore, we seek to balance the *skewness* of the label distribution presented to the weak classifiers.

2.2. Minimizing the upper bound while balancing label skewness presented to weak classifiers

For easy explanations, let us assume for the moment that we are learning offline. The offline version of our algorithm is sketched in (algorithm 2). The algorithm itself is similar to Viola and Jones's Asymmetric AdaBoost [16]. However, real-valued weak classifiers [14] are replaced by discrete-valued weak classifiers [3]. And, rather than multiplying the weights presented to the m -th weak classifier

Algorithm 2 Offline Asymmetric Discrete AdaBoost

Start with weights $v_n \leftarrow 1/N, n = 1, 2, \dots, N$.

for $m = 1$ to M **do**

 Choose k_m .

 Set $w_n \leftarrow v_n \sqrt[k_m]{k^{y_n}}, n = 1, 2, \dots, N$, and normalize w_n so that $\sum_{n=1}^N w_n = 1$.

 Fit weak classifier $f_m(\mathbf{x}) \in \{-1, 1\}$ using data (\mathbf{x}_n, y_n) with weights w_n .

 Set $c_m \leftarrow \frac{1}{2} \log \frac{1-e_m}{e_m}$, where e_m is the weighted error given by $e_m \leftarrow \sum_{n=1}^N w_n 1_{[y_n \neq f_m(\mathbf{x}_n)]}$.

 Set $v_n \leftarrow w_n e^{-y_n c_m f_m(\mathbf{x}_n)}, n = 1, 2, \dots, N$.

end for

Output the classifier: $\text{sign}(\sum_{m=1}^M c_m f_m(\mathbf{x}))$.

with $\sqrt[2M]{k^{y_n}}$, they are multiplied with a more general $\sqrt[k_m]{k^{y_n}}$ (e.g. k_m can be set to $\sqrt[M]{k}$). Our goal is to design parameters k_1, k_2, \dots, k_M so that the algorithm:

1. Minimizes $J(F_M)$ in (5).
2. Ensures equal label skewness presented to weak classifiers.

Let us denote, by $v_{m,n} = v_m(\mathbf{x}_n, y_n)$ and by $w_{m,n} = w_m(\mathbf{x}_n, y_n)$, the weight of the n -th example *before* and *after* being multiplied with $\sqrt[k_m]{k^{y_n}}$ respectively. According to the algorithm, they are propagated by the following rules:

$$v_{m+1}(\mathbf{x}, y) = w_m(\mathbf{x}, y) e^{-y c_m f_m(\mathbf{x})} \quad (9)$$

$$w_m(\mathbf{x}, y) = v_m(\mathbf{x}, y) \sqrt[k_m]{k^y} / Z_m \quad (10)$$

where Z_m is a normalization factor to make $w_m(\mathbf{x}, y)$ a distribution.

In Proposition A.1, we prove that at the m -th weak classifier, the algorithm fits $(c_m, f_m(\mathbf{x}))$ by minimizing $J_m(c, f) = E[(\prod_{i=1}^m \sqrt[k_i]{k^{y_i}}) e^{-y(F_{m-1}(\mathbf{x}) + cf(\mathbf{x}))}]$. Therefore, to minimize $J(F_M)$ in (5), the following condition must be true:

$$\prod_{m=1}^M k_m = k \quad (11)$$

Note that AdaBoost itself is an M -stage greedy process where at each stage, a weak classifier is fit based on previous weak classifiers. Hence, to minimize the expected loss of the ensemble classifier, the last weak classifier must be chosen to minimize this loss. Clearly, our method does not principally minimize $J(F_M)$ from the beginning (and neither do other methods [7, 11, 16]). But, as pointed out by Viola and Jones, by doing so the first weak classifier will absorb all the asymmetric weights, leaving only symmetric weights to subsequent weak classifiers, which is not a good idea. It is better to distribute the asymmetric weights among the weak classifiers equally, so that each of them

only absorb the same amount of asymmetric weights, and give similar results.

Next, denote by γ_m the skewness of the labels weighted by $v_m(\mathbf{x}, y)$:

$$\gamma_m \stackrel{\text{def}}{=} \log \frac{P_{v_m}(y = -1)}{P_{v_m}(y = 1)}, \quad (12)$$

and by γ'_m the skewness of the labels weighted by $w_m(\mathbf{x}, y)$:

$$\gamma'_m \stackrel{\text{def}}{=} \log \frac{P_{w_m}(y = -1)}{P_{w_m}(y = 1)}. \quad (13)$$

The relationship of γ_m and γ'_m is given by (Proposition A.5):

$$\gamma'_m = \gamma_m - \log k_m. \quad (14)$$

For the first weak classifier, γ_1 is the original unweighted label distribution of examples. For subsequent weak classifiers, if $f_{m-1}(\mathbf{x})$ is at the optimal (*i.e.* $f_{m-1}(\mathbf{x}) = \arg \min_f J_{m-1}(c, f)$), then based on Corollary A.4, the total weights of positive and negative examples weighted by $v_m(\mathbf{x}, y)$ are equal, implying that $\gamma_m = 0$.

Therefore, by assuming that all classifiers $f_m(\mathbf{x})$ minimize their goal $f_m(\mathbf{x}) \rightarrow \arg \min_f J_m(c, f)$, the second condition can be established:

$$\gamma_1 - \log k_1 = -\log k_m \quad \forall m \in \{2..M\}. \quad (15)$$

Solving the system of (11) and (15) analytically, we get:

$$k_1 = e^{\frac{1}{M} \log k + \frac{M-1}{M} \gamma_1} \quad (16)$$

$$k_m = e^{\frac{1}{M} (\log k - \gamma_1)} \quad \forall m \in \{2..M\}. \quad (17)$$

However, in practice $f_{m-1}(\mathbf{x})$ are not always at the optimal, because we use crude approximations to conditional expectation, such as decision trees or other constrained models, and the training size may not be large enough. Therefore, γ_m may not be 0 for $m > 1$.

We estimate k_m based on γ_m and the remaining amount of k to be distributed among the remaining weak classifiers. Suppose up to $m - 1$ weak classifiers have been trained (and hence the values of k_1, k_2, \dots, k_{m-1} are obtained). The value of k_m is chosen by constraining it with condition (11), assuming that all remaining weak classifiers minimize their goal ($f_m(\mathbf{x}) \rightarrow \arg \min_f J_m(c, f)$), and by seeking to balance the skewness of label distributions among the remaining weak classifiers. Similarly to (16), once we solve the system of equations, we get:

$$k_m = e^{\frac{1}{M-m+1} (\log k - \sum_{i=1}^{m-1} \log k_i) + \frac{M-m}{M-m+1} \gamma_m}. \quad (18)$$

2.3. Online learning

We wish to convert algorithm 2 to online learning mode. Recently, some variants of boosting methods adapted to online learning were proposed in the literature [2, 12]. But

they are mainly designed for symmetric cases. In this paper, we are interested in ideas proposed by Oza and Russell [12] which have been recently applied to computer vision problems [5, 9, 13]. Although their work was not designed for asymmetric classifiers, some of their ideas are propagated to our method.

2.3.1 Propagating the weights

In [12], Oza and Russel propagated the weights rather than by using the normal updating rule, which is sensitive to wrong predictions at early iterations, but by seeking to achieve two conditions which eventually the normal updating rule will guarantee. The first condition is, the total weights of examples presented to a weak classifier is N . It is reasonable considering that the weight of each new example can be set to 1 initially, and that the weight-updating rule is just a way of re-distributing the weights among the examples.

The second condition comes from the fact that, after c_m is optimally chosen (given $f_m(\mathbf{x})$ is fixed), the total weights presented to the next weak classifier, of correctly classified examples and wrongly classified examples, *are equal* (see Corollary 2 in [4]). By asymptotically tuning the weights presented to the next classifiers to have balance between total correctly classified weights and total wrongly classified weights, the weights will be more similar to what they are in offline AdaBoost, thereby convergence could occur more rapidly. To achieve both conditions asymptotically, they scale the weights as follows:

$$\lambda_{m+1,n} = \begin{cases} \lambda_{m,n} \frac{N/2}{\lambda_m^{sc}} & \text{if } y_n = f_m(\mathbf{x}_n) \\ \lambda_{m,n} \frac{N/2}{\lambda_m^{sw}} & \text{if } y_n \neq f_m(\mathbf{x}_n) \end{cases} \quad (19)$$

where λ_m^{sc} and λ_m^{sw} are the total weights of examples, correctly classified and wrongly classified by $f_m(\mathbf{x})$ respectively, after observing N examples; and $\lambda_{m,n}$ is the weight of example (\mathbf{x}_n, y_n) presented to the m -th weak classifier. The idea is, when N is large, the total weights presented to the next weak classifier of those correctly classified and wrongly classified should both converge to $N/2$.

The property of balance between total correctly classified weights and total wrongly classified weights also holds for weights $v_{m,n}$ (equations (9) and (10)) in our method (see Corollary A.3 for the proof). Besides, we expect that each weak classifier $f_m(\mathbf{x})$ will eventually be at the optimal when $N \rightarrow \infty$. Based on Corollary A.4, it means the total weights of positive examples and the total weights of negative examples will eventually be *equal*.

Therefore, in our method, the weights $v_{m,n}$ are propagated to asymptotically achieve three conditions rather than two, expecting a faster convergence rate. The first two conditions are the same as those of Oza and Russel, but applied to weights $v_{m,n}$. The last condition is to ensure the total

weights $v_{m,n}$ of positive examples and negative examples to be equal.

Our weight-updating rule is as follows:

$$v_{m+1,n} = \begin{cases} v_{m,n} \frac{a_m N/2}{v_m^{tp}} & \text{if } y = 1 \text{ and } f_m(\mathbf{x}_n) = 1 \\ v_{m,n} \frac{(1-a_m)N/2}{v_m^{fn}} & \text{if } y = 1 \text{ and } f_m(\mathbf{x}_n) = -1 \\ v_{m,n} \frac{a_m N/2}{v_m^{fp}} & \text{if } y = -1 \text{ and } f_m(\mathbf{x}_n) = 1 \\ v_{m,n} \frac{(1-a_m)N/2}{v_m^{tn}} & \text{if } y = -1 \text{ and } f_m(\mathbf{x}_n) = -1 \end{cases} \quad (20)$$

where v_m^{tp} , v_m^{fn} , v_m^{fp} , and v_m^{tn} are the total weights of, true positive examples, false negative examples, false positive examples, and true negative examples respectively, after observing N examples; and a_m is an estimate of the weighted probability that $f_m(\mathbf{x})$ predicts a positive label, given by:

$$a_m \stackrel{\text{def}}{=} \frac{v_m^{tp} + v_m^{fp}}{v_m^{tp} + v_m^{fp} + v_m^{tn} + v_m^{fn}}. \quad (21)$$

Proposition A.6 verifies that our weight-updating rule asymptotically achieves the three conditions above. Note that γ_m can be estimated from a_m by:

$$\gamma_m = \log \frac{P_{v_m}(y = -1)}{P_{v_m}(y = 1)} \approx \log \frac{1 - a_m}{a_m}. \quad (22)$$

2.3.2 Estimating c_m

Once $f_m(\mathbf{x})$ is updated using the using the weight-updating rule in (20), we need to estimate c_m . Let w_m^{tp} , w_m^{fn} , w_m^{fp} , and w_m^{tn} be the total weights $w_{m,n}$ of, true positive examples, false negative examples, false positive examples, and true negative examples respectively, after observing N examples. Based on Proposition A.1, we get:

$$c_m = \frac{1}{2} \log \frac{1 - e_m}{e_m}, \quad (23)$$

where $e_m = \epsilon_{w_m}(f_m)$ is the weighted expected error of $f_m(\mathbf{x})$. From (10), we can estimate e_m by:

$$e_m \approx \frac{w_m^{fn} + w_m^{fp}}{w_m^{tp} + w_m^{fn} + w_m^{fp} + w_m^{tn}} \quad (24)$$

$$= \frac{v_m^{fn} \sqrt{k_m} + v_m^{fp} / \sqrt{k_m}}{(v_m^{tp} + v_m^{fn}) \sqrt{k_m} + (v_m^{fp} + v_m^{tn}) / \sqrt{k_m}}. \quad (25)$$

The final online algorithm is presented in (algorithm 1). Note that, since we seek to scale the sum of $v_{m,n}$ to N , we expect Z_m to be a constant factor. Therefore, we do not normalize $w_{m,n}$.

3. Experimental Results

We performed three experiments to test the performance of our method, and to see how other online boosting methods for object detection problems [5, 9, 13] benefited from

Dataset	AdaBoost	Online Boosting	Our method
Promoters	0.8455	0.7136	0.7429+
Breast Cancer	0.9445	0.9573	0.9552
German Credit	0.735	0.6879	0.7013+
Chess	0.9517	0.9476	0.9501
Mushroom	0.9966	0.9987	0.9988
Census Income	0.9365	0.9398	0.9372
Synthetic 5-dim	0.9382	0.9049	0.9251
Synthetic 50-dim	0.8923	0.7972	0.8404+

Table 1. Online Boosting vs. Our method

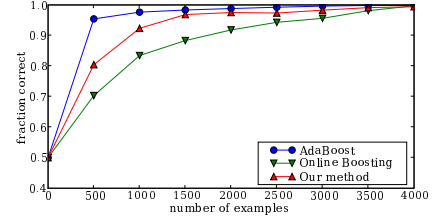


Figure 2. Learning curves for Mushroom dataset

our method. All our experiments shown below classified objects in real time on a standard PC (Intel Pentium IV 2.00 GHz with 512 MB RAM).

3.1. Online Boosting

We compared our method with the Online Boosting algorithm of Oza and Rusell [12] on several two-class datasets (table 1) from the UCI KDD [6] presented in the work. The last two datasets are our synthetic datasets. We used Naïve Bayes classifiers as weak classifiers. Because the datasets are symmetric (*i.e.* positive examples and negative examples are equal), we set $k = 1$. The testing environments were set up similar to that of Oza's, with 10 runs of 5-fold cross validation.

We noticed that our method performs better than Online Boosting on under-complete datasets (those with a '+' in table 1), and performs similarly on over-complete datasets. This is expected as our method converges faster than Online Boosting in early iterations, but eventually both methods converge to that of offline AdaBoost (*e.g.* see figure 2) as more examples are observed.

3.2. Online co-training for moving objects detection

In [9], Javed *et al.* proposed an online co-training framework for object detection. In their work, a classifier is first trained offline on a generic scenario; and then updated using Online Boosting, allowing the boosted classifier to adapt to the current scenario.

We implemented their method on the problem of pedestrian detection. Similar to their work, we used a background model to subtract moving regions, followed by scaling the



Figure 3. Some classification results from PETS2001 dataset1. White box: region predicted as pedestrian. Black box: region predicted as non-pedestrian.

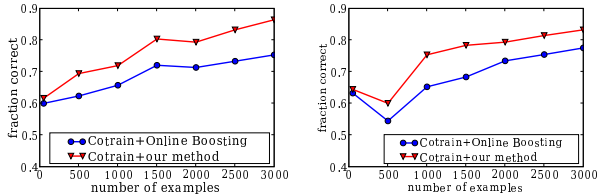


Figure 4. Performance on the PETS2001 datasets. Each example corresponds to a detected moving region. Left: PETS2001 dataset 1. Right: PETS2001 dataset 2.

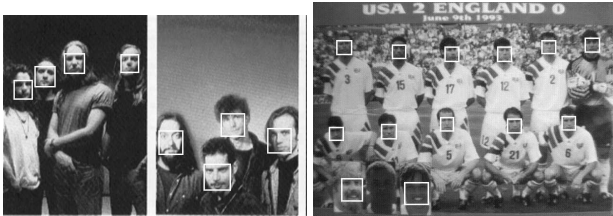


Figure 5. Some detection results of our Online Asymmetric Discrete AdaBoost on the MIT+CMU test set.

regions down to 30x30 pixels, and then applying PCA on the gradient magnitudes to obtain the global features. The classifier (pedestrian detector) was first trained offline using 150 manually labeled images, and then updated online. We compared the performance of their method when using Online Boosting, and when using our method. We used the PETS2001 datasets¹ for training and testing the results. The performance (in figure 3 and figure 4) shows that co-training benefited from our method.

3.3. Online learning for face detection

We analyzed the performance of five algorithms on the problem of face detection: Viola and Jones' Offline Asymmetric (Discrete) AdaBoost (DAB.VJ) [16], our Offline Asymmetric Discrete AdaBoost (DAB.A), our Online Asymmetric Discrete AdaBoost (DAB.OA), Online AdaBoost for feature selection using Online Boosting (DAB.OFS) [5], and Online AdaBoost for feature selection using our Online Asymmetric Discrete AdaBoost (DAB.OAFS). Here, the feature selection technique in the

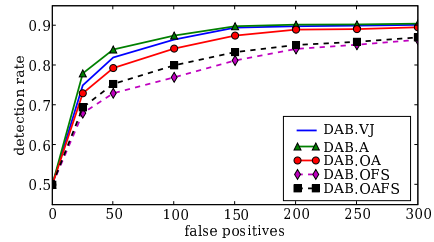


Figure 6. ROC curves of the algorithms on the MIT+CMU set.

last two algorithms is the technique presented in [5, 13] that use Oza's Online Boosting.

The face detectors followed the same flowchart as in [15]. A total of 1500 frontal face images and 5000 non-face images were collected. Face and non-face images are scaled to a size 24x24 pixels. Haar-like features are extracted from the sub-windows, forming a feature pool of 5000 features. For all algorithms, a cascade of 25 boosted classifiers, from coarse to fine, was set up. The first classifier had one weak classifier, while the final classifiers had 200 weak classifiers.

We tested the algorithms on the MIT+CMU frontal face test set² (see figure 6 and figure 7). Note that for the purpose of testing, we used fewer boosted classifiers and had a smaller feature pool than existing offline-trained face detectors. As the result, the obtained detection rates were lower than theirs.

For algorithms DAB.VJ, DAB.A, and DAB.OA, a weak classifier selects the best feature from the big feature pool. As a result, the training processes took very long. For algorithms that use online feature selection (FS), *i.e.* DAB.OFS and DAB.OAFS, the size of the pool was much smaller (we set the size to 250, the same as proposed in [5]), and hence the training processes were much shorter (about 20 times faster), but at the cost of reduced detection rates. We observed that online classifiers were often not so stable at first, and only converged after a fair number of iterations had passed. When the best feature of the big pool was somehow swapped into the small pool, there was a good chance it would be rejected some time later, because of 1) being poorly trained, and 2) being compared with those not as good but well trained in the small pool. As a result, the best features selected by the additional FS process were, though pretty good, not so good as those selected from the big feature pool.

Careful examination of the ROC curves shows a few points: 1) our skewness balancing scheme resulted in a small performance gain over that of Viola and Jones, 2) our Online Asymmetric Discrete AdaBoost converged asymptotically to the first two offline algorithms as more examples were trained, and 3) the online feature selection technique

¹<http://ftp.pets.rdg.ac.uk/PETS2001/>

²http://vasc.ri.cmu.edu/idb/html/face/frontal_images/

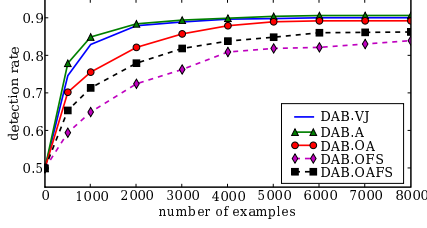


Figure 7. Detection rate over number of examples observed of a 10-feature boosted classifier. k is set to 20 for all algorithms.

also benefited from our Online Asymmetric Discrete AdaBoost algorithm.

4. Conclusions

In this paper, we presented an integrated framework for learning asymmetric boosted classifiers and online learning to address the problem of online learning asymmetric boosted classifiers, which is applicable to object detection problems. In addition, our method seeks to balance the skewness of the labels presented to the weak classifiers, allowing them to be trained more equally, and hence resulting in a small performance gain. In terms of online learning, we enforced three conditions to the weights of the examples presented to weak classifiers, resulting in a faster convergence speed when compared to the Online Boosting algorithm [12] used in online boosting methods for object detection problems. Our experiments showed that, by replacing Online Boosting with our algorithm, about 0-10% increase in accuracy and about 5-30% gain in learning speed were observed, depending on the problem the algorithms were applied to.

A. Appendix

Proposition A.1. Consider the Offline Asymmetric Discrete AdaBoost algorithm presented in (algorithm 2), the algorithm sequentially fits weak classifier $(c_m, f_m(\mathbf{x}))$ by minimizing $E[(\prod_{i=1}^m \sqrt{k_i^y}) e^{-y(F_{m-1}(\mathbf{x}) + cf(\mathbf{x}))}]$ w.r.t. $(c, f(\mathbf{x}))$.

Proof. The proof is analogous to Proposition 1 in Friedman et al. [4], except that here we have an extra term $\prod_{i=1}^m \sqrt{k_i^y}$. In addition, the proof in [4] requires an approximation of $f(\mathbf{x})$ up to the second order. The approximation is avoided in our proof. Let $w_m(\mathbf{x}, y)$ be the weighting function of the examples when presenting to the m -th weak classifier. By induction, we have

$$w_m(\mathbf{x}, y) = \left(\prod_{i=1}^m \sqrt{k_i^y} \right) e^{-yF_{m-1}(\mathbf{x})} / Z_m, \quad (26)$$

where Z_m is the normalization factor.

The goal is rewritten as:

$$J_m(c, f) = E\left[\left(\prod_{i=1}^m \sqrt{k_i^y}\right) e^{-y(F_{m-1}(\mathbf{x}) + cf(\mathbf{x}))}\right] \quad (27)$$

$$= E[Z_m w_m(\mathbf{x}, y) e^{-ycf(\mathbf{x})}] \quad (28)$$

$$= Z_m E[w_m(\mathbf{x}, y)] E_{w_m}[e^{-ycf(\mathbf{x})}], \quad (29)$$

where $E_w[\cdot]$ is the *weighted expectation* defined by

$$E_w[g(\mathbf{x}, y)] \stackrel{\text{def}}{=} \frac{E[w(\mathbf{x}, y)g(\mathbf{x}, y)]}{E[w(\mathbf{x}, y)]}. \quad (30)$$

Now consider the two choices of $f(\mathbf{x})$:

$$\begin{aligned} & E_{w_m}[e^{-ycf(\mathbf{x})}] \\ &= E_{w_m}[e^{-c} 1_{yf(\mathbf{x})=1} + e^c 1_{yf(\mathbf{x}) \neq 1}] \end{aligned} \quad (31)$$

$$= e^{-c} E_{w_m}[1_{yf(\mathbf{x})=1}] + e^c E_{w_m}[1_{yf(\mathbf{x}) \neq 1}]. \quad (32)$$

Denote by $\epsilon_{w_m}(f) = E_{w_m}[1_{yf(\mathbf{x}) \neq 1}]$ the *weighted expected error* of weak classifier $f(\mathbf{x})$, then:

$$\begin{aligned} E_{w_m}[e^{-ycf(\mathbf{x})}] &= e^{-c}(1 - \epsilon_{w_m}(f)) + e^c \epsilon_{w_m}(f) \\ &= e^{-c} + (e^c - e^{-c}) \epsilon_{w_m}(f). \end{aligned} \quad (33)$$

Therefore, minimizing (27) w.r.t. $f(\mathbf{x})$ is equivalent to minimizing $\epsilon_{w_m}(f)$ w.r.t. $f(\mathbf{x})$.

Next, suppose $\hat{f}(\mathbf{x})$ is the weak classifier trained by minimizing $\epsilon_{w_m}(f)$. Setting the derivative of $J_m(c, \hat{f})$ w.r.t. c to 0, we get:

$$\hat{c} = \arg \min_c E_{w_m}[e^{-yc\hat{f}(\mathbf{x})}] = \frac{1}{2} \log \frac{1 - \epsilon_{w_m}(\hat{f})}{\epsilon_{w_m}(\hat{f})}. \quad (34)$$

□

Corollary A.2. Consider the Asymmetric AdaBoost algorithm of Viola and Jones [16] where Discrete AdaBoost is used instead of Real AdaBoost. The algorithm sequentially fits weak classifier $(c_m, f_m(\mathbf{x}))$ by minimizing $E[{}^2\sqrt{k^y} e^{-y(F_{m-1}(\mathbf{x}) + cf(\mathbf{x}))}]$ w.r.t. $(c, f(\mathbf{x}))$.

Proof. The results follows from Proposition A.1 considering $k_1 = k_2 = \dots = k_M = \sqrt[k]{k}$. □

Corollary A.3. Consider the Offline Asymmetric Discrete AdaBoost algorithm presented in (algorithm 2), after the m -th weak classifier is trained:

$$E[yf_m(\mathbf{x})v_{m+1}(\mathbf{x}, y)] = 0. \quad (35)$$

Proof. The result is similar to Corollary 2 in Friedman et al. [4], and follows from considering the partial derivative of $J_m(c, f)$ w.r.t. c at point $(c_m, f_m(\mathbf{x}))$ is 0. □

Corollary A.4. Consider the Offline Asymmetric Discrete AdaBoost algorithm presented in (algorithm 2), at the optimal $f_m(\mathbf{x}) = \arg \min_f J_m(c, f)$:

$$E[yv_{m+1}(\mathbf{x}, y)] = 0. \quad (36)$$

Proof. Setting the point-wise derivative of $J_m(c, f)$ w.r.t. $f(\mathbf{x})$ to 0, and then marginalizing it over \mathbf{x} , we get the result. \square

Proposition A.5. Suppose the weighted marginal distribution of y presented to the m -th weak classifier has skewness γ_m , if we multiply the weight of each example (\mathbf{x}_n, y_n) by $\sqrt{k_m^{y_n}}$, the newly weighted marginal distribution has skewness $\gamma'_m = \gamma_m - \log k_m$.

Proof. Let $v_m(\mathbf{x}, y)$ and $w_m(\mathbf{x}, y)$ be the weighting function before and after being multiplied respectively. We have

$$w_m(\mathbf{x}, y) = v_m(\mathbf{x}, y)\sqrt{k_m^y}. \quad (37)$$

Let w_m^p and w_m^n be the total weights of positive and negative examples respectively *after* being multiplied, and v_m^p and v_m^n be the total weights of positive and negative examples respectively *before* being multiplied. By definition:

$$\begin{aligned} \gamma'_m &= \log \frac{P_{w_m}(y = -1)}{P_{w_m}(y = 1)} = \log \frac{w_m^n}{w_m^p} = \log \frac{v_m^n/\sqrt{k_m}}{v_m^p\sqrt{k_m}} \\ &= \log \frac{v_m^n}{v_m^p k_m} = \log \frac{P_{v_m}(y = -1)}{P_{v_m}(y = 1)} - \log k_m \end{aligned} \quad (38)$$

$$= \gamma_m - \log k_m. \quad (39)$$

\square

Proposition A.6. Consider the weight-updating rule proposed in (20), when $N \rightarrow \infty$:

$$v_m^{tp} + v_m^{fn} + v_m^{fp} + v_m^{tn} \rightarrow N \quad (40)$$

$$v_m^{tp} + v_m^{fn} - v_m^{fp} - v_m^{tn} \rightarrow 0 \quad (41)$$

$$v_m^{tp} - v_m^{fn} - v_m^{fp} + v_m^{tn} \rightarrow 0 \quad (42)$$

Proof. We prove by induction. These conditions are true for $m = 1$. Suppose they are true up to $m = i - 1$ for some i . We consider v_i^{tp} , v_i^{fn} , v_i^{fp} , and v_i^{tn} as the total weight estimates of true positives, false negatives, false positives, and true negatives respectively of the $(i - 1)$ -th classifier (multiplied by factor N). By definition, as $N \rightarrow \infty$:

$$v_i^{tp} \rightarrow a_{i-1}N/2 \quad (43)$$

$$v_i^{fn} \rightarrow (1 - a_{i-1})N/2 \quad (44)$$

$$v_i^{fp} \rightarrow a_{i-1}N/2 \quad (45)$$

$$v_i^{tn} \rightarrow (1 - a_{i-1})N/2 \quad (46)$$

The results follow. \square

Acknowledgements

This work was done at Centre for Multimedia and Networking (CeMNet), School of Computer Engineering, Nanyang Technological University, Singapore. We would like to thank Xi Chen, Peng Song, Minhtuan Pham, ThanhGiang T. Vo, and ThanhVu H. Nguyen for their support on this work.

References

- [1] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan. Adacost: Misclassification cost-sensitive boosting. In *Int. Conf. on Machine Learning*, pages 97–105, 1999.
- [2] A. Fern and R. Givan. Online ensemble learning: An empirical study. *Machine Learning*, 53(1-2):71–109, 2003.
- [3] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *Int. Conf. on Machine Learning*, pages 148–156, 1996.
- [4] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2):337–374, 2000.
- [5] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. CVPR*, volume 1, pages 260–267, 2006.
- [6] S. Hettich and S. D. Bay. The uci kdd archive. University of California, Irvine, Dept. of Information and Computer Sciences, 1999.
- [7] X. Hou, C.-L. Liu, and T. Tan. Learning boosted asymmetric classifiers for object detection. In *Proc. CVPR*, volume 1, pages 330–338, 2006.
- [8] C. Huang, H. Ai, Y. Li, and S. Lao. Vector boosting for rotation invariant multi-view face detection. In *Int. Conf. on Computer Vision*, volume 1, pages 446–453, 2005.
- [9] O. Javed, S. Ali, and M. Shah. Online detection and classification of moving objects using progressively improving detectors. In *Proc. CVPR*, volume 1, pages 696–701, 2005.
- [10] S. Z. Li, L. Zhu, Z. Zhang, A. Blake, H. Zhang, and H. Shum. Statistical learning of multi-view face detection. In *Proc. ECCV*, pages 67–81, 2002.
- [11] Y. Ma and X. Ding. Robust real-time face detection based on cost-sensitive adaboost method. In *Proc. ICME*, volume 2, pages 465–472, 2003.
- [12] N. Oza. Online bagging and boosting. In *Int. Conf. on Systems, Man and Cybernetics*, volume 3, pages 2340–2345, 2005.
- [13] P. Roth, H. Grabner, D. Skocaj, H. Bischof, and A. Leonardis. On-line conservative learning for person detection. In *Int. Workshop on VSPETS*, pages 223–230, 2005.
- [14] R. E. Schapire and Y. Singer. Improved boosting using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999.
- [15] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proc. CVPR*, 2001.
- [16] P. Viola and M. Jones. Fast and robust classification using asymmetric adaboost and a detector cascade. In *NIPS*. MIT Press, Cambridge, MA, 2002.