

On Asymmetric Classifier Training for Detector Cascades

Timothy F. Gee

Oak Ridge National Laboratory*

Abstract. This paper examines the Asymmetric AdaBoost algorithm introduced by Viola and Jones for cascaded face detection. The Viola and Jones face detector uses cascaded classifiers to successively filter, or reject, non-faces. In this approach most non-faces are easily rejected by the earlier classifiers in the cascade, thus reducing the overall number of computations. This requires earlier cascade classifiers to very seldomly reject true instances of faces. To reflect this training goal, Viola and Jones introduce a weighting parameter for AdaBoost iterations and show it enforces a desirable bound. During their implementation, a modification to the proposed weighting was introduced, while enforcing the same bound. The goal of this paper is to examine their asymmetric weighting by putting AdaBoost in the form of Additive Regression as was done by Friedman, Hastie, and Tibshirani. The author believes this helps to explain the approach and adds another connection between AdaBoost and Additive Regression.

1 Introduction

The Viola and Jones face detector [1] was a very significant change in approach to face detection. Based on this method, several researchers have developed systems to detect faces of different pose and other objects [2–4]. There is a considerable computational load on face detectors that rely on only the cues obtained from still, gray images. For such an algorithm to reliably detect faces of different scales, it must perform many scans through each image, parsing the image by small regions, using a small spatial increment. Viola and Jones report their algorithm having to classify 50,000 regions for a typical image. Considering that the number of regions that contain a face is likely to be on the order of 1-10, then it becomes obvious that it is important to perform rejections on most image regions as quickly as possible, while taking great care to avoid carelessly rejecting face regions. The cascaded face detector is very fast because of its use of Haar-wavelet-like classifiers that are efficiently calculated from integral images and the novel method of placing classifiers in a cascade, such that non-faces are usually rejected quickly.

In the original work, Viola and Jones leave the face and non-face samples equally weighted, and the number of training samples are on the same order of

* Document prepared by Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

magnitude. They feed the system with 4,916 positive face samples, the greatest majority of which should pass completely through the cascade. At each layer they incorporate 10,000 non-face samples, so the number of positive samples is only a factor of two less than the number of negative samples. At run-time, the detector will scale the size of the features to find faces of various sizes, but the base size used for training in such systems is typically about 24x24.

In the design of the cascade, the designer has in mind a particular goal for the overall system's detection rate, D , and false acceptance rate, A . These overall goals can be used to set goals for the detection and false accept rates of individual stages. This is because the overall performance rates are obtained from the product of the stage performance rates. That is

$$D = \prod_{c=1}^C d_c \quad (1)$$

and

$$A = \prod_{c=1}^C a_c, \quad (2)$$

where d_c is the detection rate for classifier c , and a_c is the false accept rate for classifier c in the cascade. With this in mind, Viola and Jones state that there is maximum acceptable a_c and minimum acceptable d_c for each cascade stage c , and they continue to add features to be evaluated in a given stage until an acceptable false accept rate a_c is achieved. While this is being performed, at each iteration, the discriminant threshold θ_c for the stage is adjusted to ensure that the detection rate d_c is high enough.

Viola and Jones use the steps described in Algorithm 1 to develop each stage classifier in the detector cascade. This is the Discrete AdaBoost algorithm with the introduced θ threshold added to the discriminate function. In their case, each weak classifier $f_m(x)$ is a test applied to a single Haar-wavelet-like feature that has been evaluated on the given training region.

2 Asymmetric Bias Weighting

As discussed above, the original form of the Viola-Jones face detector moved to a different point on the ROC curve by modifying the threshold, θ_c . However, this is clearly non-optimal, since the threshold is determined after weak classifiers have been chosen. In the later paper [5] by Viola and Jones, they intend to handle this by multiplying the weight of face and non-face training samples by a bias term. Initially they advocate multiplying the weight of each sample n by

$$\exp\left(y_n \ln \sqrt{k}\right), \quad (3)$$

where y_n is the desired or true classification as used in 1. For this application, y_n is 1 for faces and -1 for non-faces. k is a weighting that is used to increase

Algorithm 1 Steps for Discrete AdaBoost with threshold θ

Assume training samples $\{(x_n, y_n)\}_{n=1}^N$
 with $x \in \mathcal{R}^k$, and $y \in \{-1, 1\}$

Perform the following steps

- 1) Initialize observation weights
 $w_n = 1/N$ for $n = 1, 2, \dots, N$.
 - 2) For $m = 1$ to M :
 - a) Fit the classifier $f_m(x) \in \{-1, 1\}$
 to the training data weighted according to w_n .
 - b) Compute the error as $\epsilon_m = \frac{\sum_{n=1}^N w_n I(y_n \neq f_m(x_n))}{\sum_{n=1}^N w_n}$.
 - c) Compute $\alpha_m = \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$.
 (Note: must have $\epsilon_m < 0.5$)
 - d) Set $w_i \leftarrow w_i \exp[\alpha_m I(y_n \neq f_m(x_n))]$
 for $n = 1, 2, \dots, N$.
 - 3) Final classifier is $g(x) = \text{sgn}\left[\sum_{m=1}^M \alpha_m f_m(x) > \theta\right]$.
-

the detection rate for the resulting strong classifier. The motivation is to make the false rejections cost k times more than false detections. That is

$$cost(n) = \begin{cases} \sqrt{k} & \text{if } y_n = 1 \text{ and } g(x_n) = -1 \\ \sqrt{\frac{1}{k}} & \text{if } y_n = -1 \text{ and } g(x_n) = 1 \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where again, y_n is the true classification, and $g(x_n)$ is the estimated classification.

In the Viola and Jones paper, it is shown that the solution's training error bound achieves the desired cost ratio by weighting the original importance of the training samples using k . However, through empirical testing, it became evident that the weighting only affects the weak classifier in the first AdaBoost iteration. Then the authors decide to factor the term into parts and apply the M th root of the weighting k during each of M rounds of AdaBoost training. This enforces the same bound, the asymmetric weighting was found to take effect throughout the training of the ensemble, and improved results were obtained in an illustrative two-dimensional classification example. Also, beyond this two-dimensional case, Viola and Jones train their face detector cascade using asymmetric AdaBoost and compare it to that obtained using symmetric AdaBoost and found that the new ROC curve points indicated higher detection rates for given false acceptance

Algorithm 2 Forward Stage-wise Additive Modeling [8]

Assume training samples $\{(x_i, y_i)\}_{i=1}^N$ with $x \in \mathcal{R}^k$, and $y \in \{-1, 1\}$

Perform the following steps

- 1) Initialize $f_0(x) = 0$.
 - 2) For $m = 1$ to M :
 - a) Compute $(\beta_m, \gamma_m) = \arg \min_{\beta, \gamma} \sum_{i=1}^N L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma))$,
 where $b(\cdot)$ is a parameterized basis function,
 and $L(\cdot)$ is a norm.
 - b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.
 - 3) Final classifier is $g(x) = \text{sgn}[f_M(x)]$.
-

rates. It is worth noting that the Asymmetric AdaBoost approach has been repeated by others for face detection and other applications [6].

3 Comparison to Additive Regression

Based on the work of Friedman, Hastie, and Tibshirani [7] AdaBoost iterations can be shown to be equivalent to Additive Regression when a specific loss function is used. The steps for achieving a classifier from Additive Regression are shown in Algorithm 2.

It is shown in [7] that by using the loss function

$$L(y, f(x)) = \exp(-yf(x)), \quad (5)$$

with Additive Regression, it is equivalent to Discrete AdaBoost. Here it is shown that the Asymmetric AdaBoost presented by Viola and Jones is equivalent to creating a classifier from additive regression using a loss function of:

$$L_m(y, f(x)) = \exp(-yf(x)) \exp\left(y \sum_{j=1}^m \frac{1}{M} \ln \sqrt{k}\right), \quad (6)$$

where m is the number of the current additive iteration. The proof is as follows. At each iteration m , the parameters for additive regression are chosen by

$$(\beta_m, b_m) = \arg \min_{\beta, b} \sum_{i=1}^N \exp[-y_i (f_{m-1}(x_i) + \beta b(x_i))] \exp\left(y_i \sum_{j=1}^m \frac{1}{M} \ln \sqrt{k}\right). \quad (7)$$

This is equivalent to

$$(\beta_m, b_m) = \arg \min_{\beta, b} \sum_{i=1}^N \omega_i^{(m)} \exp[-y_i \beta b(x_i)], \quad (8)$$

where

$$\omega_i^{(m)} = \exp[-y_i f_{m-1}(x_i)] \exp\left(y_i \sum_{j=1}^m \frac{1}{M} \ln \sqrt{k}\right). \quad (9)$$

The ω_i^m correspond to the sample weights used in the AdaBoost algorithm. This derivation is identical to that shown in [7] with the insertion of the Asymmetric weighting term. As in [7] it can be shown that β and b are chosen identically to that of the AdaBoost algorithm. We can determine the update rule for sample weights by examining

$$\omega_i^{(m+1)} = \exp[-y_i f_m(x)] \exp\left(y_i \sum_{j=1}^{m+1} \frac{1}{M} \ln \sqrt{k}\right) \quad (10)$$

$$= \exp[-y_i (f_{m-1}(x_i) + \beta G(x_i))] \exp\left(y_i \sum_{j=1}^m \frac{1}{M} \ln \sqrt{k}\right) \exp\left(\frac{1}{M} y_i \ln \sqrt{k}\right) \quad (11)$$

$$= \omega_i^{(m)} \exp[-y_i \beta G(x_i)] \exp\left(\frac{1}{M} y_i \ln \sqrt{k}\right). \quad (12)$$

Thus, the sample weights are updated identically to what was proposed in [5].

4 Objective Function

Again, the work of [7] is followed closely in order to determine why it is desirable to use these iterations to find a strong classifier. The components of the final strong classifier are chosen in an attempt to minimize the expected value of the final loss function. Although, the loss function $L_m()$ changes with each iteration, at the final iteration, M , the loss function is

$$L_M(y, f(x)) = \exp[-y f(x)] \exp\left(y \sum_{j=1}^M \frac{1}{M} \ln \sqrt{k}\right) \quad (13)$$

$$= \exp[-y f(x)] \exp\left(y \ln \sqrt{k}\right). \quad (14)$$

The objective function to be minimized is then

$$J(F) = E \left[\exp[-yF(x)] \exp \left(y \ln \sqrt{k} \right) \right] \quad (15)$$

$$\begin{aligned} &= P(y = 1|x) \exp[-F(x)] \exp \left(\ln \sqrt{k} \right) \\ &\quad + P(y = -1|x) \exp[F(x)] \exp \left(-\ln \sqrt{k} \right) . \end{aligned} \quad (16)$$

By taking the derivative of $J(F)$ with respect to F and setting it to 0, we obtain

$$F(x) = \frac{1}{2} \ln \left[\frac{kP(y = 1|x)}{P(y = -1|x)} \right] . \quad (17)$$

AdaBoost uses the sign of this function to determine classification. It can be seen that the probability of choosing class $y = 1$ is multiplied by k .

5 Experimental Results

Although this paper is mainly a mathematical reinforcement of the paper by Viola and Jones, experimental results are given to provide some further quantification of the method. In a contrived example, 1000 samples are randomly obtained from each of two class distributions. Each class is distributed by a two-dimensional Gaussian uncorrelated in x and y and with standard deviations $\sigma_x = 2$ and $\sigma_y = 0.5$. The mean of class 1 is $\mu_1 = (0, -1)$, and the mean of class 2 is $\mu_2 = (0, 1)$. The training samples are shown in Figure 1.

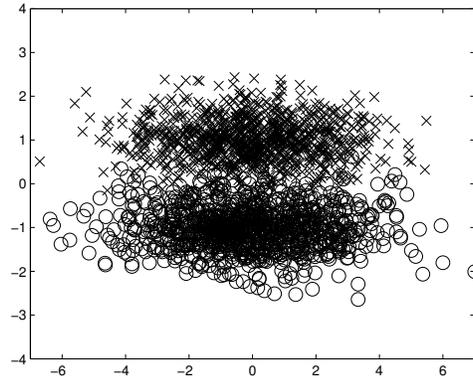


Fig. 1. Training data for two classes in example 1.

In this hypothetical case, which is indicative of performance in near-ideal situations, $M = 12$ iterations were performed, with the asymmetric weighting

$k = 10$. The weak classifiers are obtained by simply choosing a feature and threshold that minimizes the current weighted error. The obtained results are shown in Table 1.

Table 1. Example 1 performance in percent for strong classifier of 12 iterations, $k = 10$

	True Detections	False Detections	True Rejections	False Rejections
Training	99.6	7.4	92.6	0.4
Testing	99.5	6.0	94.0	0.5

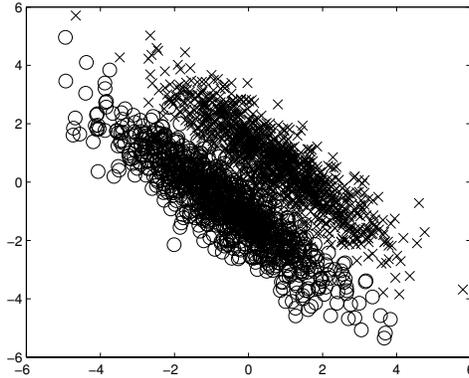


Fig. 2. Training data for two classes in example 2.

In the second example, the same class distributions and class separation were used, but the Gaussians were rotated 45 degrees with respect to the feature axes. The mean of class 1 is $\mu_1 = (-1/\sqrt{2}, -1/\sqrt{2})$, and the mean of class 2 is $\mu_2 = (1/\sqrt{2}, 1/\sqrt{2})$. The training samples are shown in Figure 2. This example is indicative of lesser performance using the same method. $M = 12$ iterations were performed, with the asymmetric weighting $k = 10$. The obtained results are shown in Table 2.

6 Conclusions

This paper has shown how Asymmetric AdaBoost can be represented as a form of Additive Regression, given the correct loss function. This provides clarification

Table 2. Example 2 performance in percent for strong classifier of 12 iterations, $k = 10$

	True Detections	False Detections	True Rejections	False Rejections
Training Data	98.6	34.0	66.0	1.4
Testing Data	97.7	37.0	63.0	2.3

for how this approach works, as well as increasing the general understanding of AdaBoost. Examples with simulated classes were used to show how the approach influences the training goal. The ability of the weak classifiers to precisely partition the sample space can distort the effects of the bias weighting. However, it can be easily seen from the examples that a significant decision bias is achieved.

References

1. Viola, P., Jones, M.: Robust real-time object detection. In: ICCV Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling, IEEE (2001)
2. Li, S.Z., Zhu, L., Zhang, Z., Zhang, H.: Learning to detect multi-view faces in real-time. In: Proceedings of the IEEE International Conference on Development and Learning. (2002)
3. Lienhart, R., Maydt, J.: An extended set of haar-like features for rapid object detection. In: Proceedings of the IEEE International Conference on Image Processing. Volume 1. (2002) 900–903
4. Wu, J., Mullin, M.D., Rehg, J.M.: Linear asymmetric classifier for cascade detectors. In: ICML '05: Proceedings of the 22nd international conference on Machine learning, New York, NY, USA, ACM Press (2005) 988–995
5. Viola, P., Jones, M.: Fast and robust classification using asymmetric adaboost and a detector cascade. NIPS 14 (2002)
6. Healy, M., Ravindran, S., Anderson, D.: Effects of varying parameters in asymmetric adaboost on the accuracy of a cascade audio classifier. In: SoutheastCon, 2004. Proceedings. IEEE. (2004) 169–172
7. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting (revised with discussions). *The Annals of Statistics* **28** (2000) 337–407
8. Hastie, T., Tibshirani, R., Friedman, J.: *The Elements of Statistical Learning*. Springer-Verlag (2001)