ShareBoost: Boosting for Multi-view Learning with Performance Guarantees*

Jing Peng, Costin Barbu, Guna Seetharaman, Wei Fan, Xian Wu, and Kannappan Palaniappan

Montclair State University; MIT Lincoln Lab; AFRL/RITB; IBM T.J. Watson Research; IBM China Research; University of Missouri-Columbia
jing.peng@montclair.edu, barbu@ll.mit.edu, Gunasekaran.Seetharaman@rl.af.mil, weifan@us.ibm.com, wuxian@cn.ibm.com, palaniappank@missouri.edu

Abstract. Algorithms combining multi-view information are known to exponentially quicken classification, and have been applied to many fields. However, they lack the ability to mine most discriminant information sources (or data types) for making predictions. In this paper, we propose an algorithm based on boosting to address these problems. The proposed algorithm builds base classifiers independently from each data type (view) that provides a partial view about an object of interest. Different from AdaBoost, where each view has its own re-sampling weight, our algorithm uses a single re-sampling distribution for all views at each boosting round. This distribution is determined by the view whose training error is minimal. This shared sampling mechanism restricts noise to individual views, thereby reducing sensitivity to noise. Furthermore, in order to establish performance guarantees, we introduce a randomized version of the algorithm, where a winning view is chosen probabilistically. As a result, it can be cast within a multi-armed bandit framework, which allows us to show that with high probability the algorithm seeks out most discriminant views of data for making predictions. We provide experimental results that show its performance against noise and competing techniques.

Keywords: Data fusion, boosting, convergence, multi-view learning.

1 Introduction

Classifiers employed in real world scenarios must deal with various adversities such as noise in sensors, intra-class variations, and restricted degrees of freedom [18]. It is often helpful to develop classifiers that rely on data from various sources (views) for classification. Such classifiers require an effective way of fusing the

* A part of the research described in this material is based upon work funded by AFRL, under AFRL Contract No. FA8750-09-2-0155. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MIT Lincoln Lab, or AFRL.

various sources of information. Resulting fused classifiers can offer a number of advantages, such as increased confidence in decision-making, resulting from fused complementary data, and robust performance against noise. Multi-view learning finds its applications in many domains such as defense, medicine, and sciences [13].

While algorithms that combine multi-view information are known to exponentially quicken object identification and classification, they lack the ability to seek out relevant information to augment a decision process. In this paper, we present a novel shared sampling approach to boosting for learning from multiple representations of data that addresses these problems. Here a representation (view) corresponds to a specific type of feature or attribute. For example, an image can be represented by (1) texture features, (2) edge features, and/or (3) shape features. Thus, each of these views provides a partial view about an object of interest, i.e., revealing a particular aspect of data. Our method provides a mechanism to exploit all of the data available, and as such, the method can be very useful for making inferences about potential objects of interest characterized with multiple views.

The proposed technique is a novel application of AdaBoost [10]. Similar to AdaBoost, our technique builds base classifiers independently from each view. Unlike AdaBoost, however, all views share the same sampling distribution as the view whose weighted training error is the minimum among all the views. This allows the most consistent data type to dominate over time, thereby significantly reducing sensitivity to noise. In addition, since the final strong classifier contains classifiers that are trained to focus on different views of the data, better generalization performance can be expected.

We note that each base classifier in the proposed algorithm is selected from one of the representations or views. We thus show that if we model base classifier selection as a sequential decision process, we can cast this scenario within a multi-armed bandit framework [3], where each view of data is modeled as the arm of a slot machine. The resulting algorithm can be viewed as a randomized version of the shared sampling algorithm in that a winning view is chosen probabilistically instead of in a greedy fashion. Furthermore, this casting allows us to show that with high probability the algorithm seeks out decision relevant views for making predictions. We also provide experimental results that corroborate our theoretical analysis.

2 Related Work

In multi-view learning, a co-training procedure for classification problems was developed [4]. The idea is that better classifiers can be learned at the individual view level, rather than constructed directly on all the available views. Co-training has been extensively investigated in the context of semi-supervised learning [22,23,8]. In this work, we are mainly interested in creating classifiers that fuse information from multiple views for better generalization.

In many ways, multi-view learning and data fusion address the same set of problems. From the viewpoint of data fusion, comprehensive surveys of various classifier fusion studies and approaches can be found in [11,12]. More recently, Lanckriet et al. [13] introduce a kernel-based data fusion (multi-view learning) approach to protein function prediction in yeast. The method combines multiple kernel representations in an optimal fashion by formulating the problem as a convex optimization problem that can be solved using semi-definite programming.

In [24] stacked generalization from multiple views was proposed. It is a general technique for construction of multi-level learning systems. In the context of multi-view learning, it yields unbiased, full-size training sets for the trainable combiner. In some cases stacked generalization is equivalent to cross-validation, in other cases it is equivalent to forming a linear combination of the classification results of the constituent classifiers. In [25], a local learning technique was proposed that combines multi-view information for better classification.

Boosting has been investigated in multi-view learning recently [21]. In particular, there is a close relationship between our technique and that proposed in [21]. If we have a single view and base classifiers are allowed to include features as well, then both techniques reduce to AdaBoost. When noise exists, however, the two techniques diverge. The technique in [21] behaves exactly like AdaBoost. Noise forces the boosting algorithm to focus on noisy examples, thereby distorting the optimal decision boundary. On the other hand, our approach restricts noise to individual views, which has a similar effect to that of placing less mass of sampling probability on these noisy examples. This is the key difference between the two techniques.

Multi-armed bandits have been studied in a number of applications [2,3]. We state that multi-armed bandits described in [5] is stochastic by nature. There are many applications where the stochastic setting can be applied to nonstationary environments, such as performance tuning problems [15] and the SAT problem [14]. Algorithms such as UCB [2] and UCBV [1] work well for making AdaBoost more efficient. Given that AdaBoost is adversarial by nature, it is difficult to use stochastic bandits to derive strong performance guarantees on AdaBoost. Many arguments made in [5] remain heuristic to an extent. However, this has been addressed in [6].

3 Shared Sampling Algorithm

In this section, we first describe the shared sampling (ShareBoost) algorithm. We then present a randomized version of it. We are given a set of training examples: $X = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$, and M disjoint features for each example $x_i = \{x_i^1, \cdots, x_i^M\}$, where $x_i^j \in \Re^{q_j}$, and $y_i \in \mathcal{Y} = \{-1, +1\}$. Each member x_i^j is known as a *view* of example x_i. We assume that examples (x_i, y_i) are drawn randomly and independently according to a fixed but unknown probability distribution D over $\mathcal{X} \times \mathcal{Y}$. Here the input space $\mathcal{X}$ is $\Re^q$, where $q = \sum_{j=1}^{M} q_j$.

The algorithm builds weak classifiers independently from each view (feature source). However, all data types share the same sampling distribution computed from the view having the smallest error rate. The key steps of the algorithm are shown in Algorithm 3, where $I(\cdot)$ is the indicator function.

```
**ShareBoost** $(\{(x_i^j, y_i)\}_{i=1}^n)$

1. **Initialization:** $w_1(i) = \frac{1}{n}$, $i = 1, \cdots, n$.
2. **For** $t = 1$ **to** $T$
   (a) Compute base classifier $h_t^j$ using distribution $w_t$
   (b) Calculate:$\epsilon_t^j = \sum_i w_t(i) I(h_t^j(x_i^j) \neq y_i)$ and $\alpha_t^* = \frac{1}{2}\ln(\frac{1-\epsilon_t^*}{\epsilon_t^*})$, where $\epsilon_t^* = \min_j\{\epsilon_t^j\}$ with corresponding $h_t^*$
   (c) Update $w_{t+1}(i) = \frac{w_t(i)}{Z_t^*} \times \exp(-y_i h_t^*(x_i^*)\alpha_t^*)$, where $Z_t^*$ is a normalization factor.
3. **Output:** $H(x) = sign(\sum_{t=1}^T \alpha_t^* h_t^*(x))$
```

Input to the algorithm is the jth view of n training examples. The algorithm produces as output a classifier that combines data from all the views. In the initialization step, all the views for a given training example are initialized with the same weight. The final decision function $H(x)$ is computed as a weighted sum of base classifiers $h_t^*(x^*)$, selected at each iteration from the views that had the smallest training error or largest α value. In this sense, ShareBoost possesses the ability to decide at each iteration which view to influence its final decision. This ability goes beyond simple subspace selection. It empowers ShareBoost not only to exploit the interplay between subspaces, but also to be more robust against noise.

We note that since we are mainly interested in asymtotatic margins, we are less concerned with distorted class probabilities associated with boosting predictions [9], especially in the two class case.

4 Randomized Shared Sampling Algorithm

The ShareBoost algorithm introduced above is greedy in that resampling weights for all views are determined solely by the winning view. That is, it employs a winner take-all strategy. One of the benefits associated with this algorithm is that noise will be restricted to individual views. In other words, noise will be *compartmentalized*, which has a similar effect to that of placing less mass of sampling probability on noisy examples. This, however, needs not to be the case in approaches such as those described in [21].

In order to provide a convergence analysis, we describe a randomized version of the shared sampling algorithm, where a winning view is chosen probabilistically. Consequently, it can be cast within a multi-armed bandit framework [3]. This in turn allows us to show that with high probability the algorithm chooses a set of *best* (large edges to be detailed later) views for making predictions.

4.1 Adversarial Multi-armed Bandit Approach

In the multi-armed bandit problem [17], a gambler chooses one of M slot machines to play. Formally, a player algorithm pulls one out of M arms at each time

t. Pulling an arm j_t at time t results in a reward $r_(j_t) \in [0,1]$, from a stationary distribution. The goal of the player algorithm is to maximize the expected sum of the rewards over the pulls. More precisely, let $G_A(T) = \sum_{t=1}^{T} r_(j_t)$ be the total reward that algorithm A receives over T pulls. Then the performance of algorithm A can be evaluated in terms of regret with respect to the average return of the optimal strategy (pulling consistently the best arm) $Reg = G_O - G_A(T)$, where $G_O = \sum_{t=1}^{T} \max_{i\in\{1,\cdots,M\}} R_(i_t)$. Here $R(i)$ represents the expected return of the ith arm.

In this work, we cast the proposed fusion algorithm within the adversarial bandit framework [2,3]. In this setup, no statistical assumptions are made about the generation of rewards. This can be viewed as having a second, non-random player that generates a reward sequence $\mathbf{r}_1, \mathbf{r}_2 \cdots$ of vectors

$$\mathbf{r}_t = (r_t(1), \cdots, r_t(M)),$$

where $r_t(j) \in [0,1]$. There is no restriction on reward vectors $\mathbf{r}(t)$ generated, which can be influenced by the player algorithm's previous actions. Only the reward $r_(j_t)$ of the chosen arm j_t is revealed to the player algorithm. Since the rewards are not drawn from a stationary distribution, any kind of regret can only be defined with respect to a particular sequence of actions. One such regret is the worst case regret $G_{(j_1,\cdots,j_T)} - G_A(T)$, where $G_{(j_1,\cdots,j_T)} = \sum_{j=1}^{T} r_t(j_t)$. Thus, the worst case regret measures how much the player algorithm lost (or gained) by following algorithm A instead of choosing actions $(i_i, \cdots, i_T)$.

A special case of this is the regret of strategy A for the best single action

$$Reg_A(T) = G_{max}(T) - G_A(T) \tag{1}$$

where $G_{max}(T) = \max_i \sum_{t=1}^{T} r_t(i)$. That is, strategy A is compared to the best fixed arm, retrospectively. Notice that when player algorithm A that achieves $\lim_{T\to\infty} \frac{Reg_A(T)}{T} \leq 0$ is called a no-regret algorithm.

4.2 Exp3.P: Exponential-Weight Algorithm for Exploration and Exploitation

The adversarial multi-armed bandit problem can be treated within the class of Exponentially Weighted Average Forecaster algorithms [7]. Typically these algorithms maintain a probability distribution over the arms and draws a random arm from this distribution at each step. The probability for pulling an arm increases exponentially with the average of past rewards the arm receives. In particular, we chose the Exp3.P (*E*xponential-weight algorithm for *E*xploration and *E*xploitation) algorithm [3], because the particular form of the probability bound on the weak regret (1) allows us to derive a strong result for the proposed fusion algorithm.

In the Exp3.P algorithm, the probability distribution (line 2(a)) for choosing arms is a mixture (weighted by γ) of the uniform distribution and a distribution that allocates a probability mass exponential in the estimated cumulative reward to each arm. This mixture ensures that the algorithm tries out all M arms. When

```
Exp3.P ($\alpha > 0$,$\gamma \in (0,1]$)
  1. Initialization: $i = 1, \cdots, M$.
     (a) $d_1(i) = exp(\frac{\alpha\gamma}{3}\sqrt{\frac{T}{M}})$
  2. For $t = 1, 2, \cdots, T$
     (a) $p_t(i) = (1-\gamma)\frac{d_t(i)}{\sum_{j=1}^{M} d_t(j)} + \frac{\gamma}{M}$, $i = 1, \cdots, M$
     (b) Choose $i_t$ randomly according to $p_t(i)$
     (c) Receive reward $r_t(i_t) \in [0,1]$
     (d) For $j = 1, \cdots, M$ set
         (1) $\hat{r}_t(j) = \begin{cases} r_t(j)/p_t(j) & \text{if } j = i_t; \\ 0 & \text{otherwise.} \end{cases}$
         (2) $d_{t+1}(j) = d_t(j)\exp(\frac{\gamma}{3M}(\hat{r}_t(j) + \frac{\alpha}{p_t(j)\sqrt{MT}}))$
```

arm i_t is selected (line 2(d)(1)), the estimated reward $\hat{r}_(i_t)$ for the arm is set to $r_t(i)/p_t(i)$. This choice compensates the reward of arms that are unlikely to be chosen. For the purpose of our analysis, we state the following theorem (Theorem 6.3 in [3]).

Theorem 1. *For any fixed* $T > 0$*, for all* $M \geq 2$ *and for all* $\delta > 0$*, if* $\gamma = \min\{3/5, 2\sqrt{(3M \log M)/(5T)}\}$ *and* $\alpha = 2\sqrt{\log(MT/\delta)}$*, then*

$$G_{max} - G_{Exp3.P} \leq 4\sqrt{MT \log \frac{MT}{\delta}} + 4\sqrt{\frac{5}{3}MT \log M} + 8 \log \frac{MT}{\delta} \quad (2)$$

holds for any assignment of rewards with probability at least $1 - \delta$*.*

It can be seen that α and γ are "smoothing" parameters: the larger they are, the more uniform the probability distribution for choosing arms $\mathbf{p}_t$. In addition, Exp3.P is a no-regret algorithm with probability 1 [3].

4.3 Randomized ShareBoost: Combining ShareBoost and Exp3.P

We are now in a position to combine ShareBoost and Exp3.P to establish a performance guarantee for the proposed algorithm. To do so, we must first specify a reward function for each information source. First, we define the training error

$$Err = \frac{1}{n}|\{i : H(x_i) \neq y_i\}|. \quad (3)$$

If we write

$$E_H(H, W_1) = \sum_{i=1}^{n} w_1(i) exp(-H(x_i)y_i), \quad (4)$$

then $E_H(H, W_1)$ upper bounds Err [20]. Furthermore, let

$$E_h(h, W_t) = \sum_{i=1}^{n} w_t(i) exp(-h(x_i)y_i).$$

It can be shown that

$$E_H(H, W_1) = \prod_{t=1}^{T} E_h(h_t, W_t). \quad (5)$$

It is shown that at each boosting round, the base learner tries to find a weak classifier h_t that minimizes $E_h(h, W_t) = \sum_{i=1}^{n} w_t(i) exp(-h(x_i)y_i)$ (Algorithm 1). Thus, minimizing $E_h(h, W_t)$ at each boosting round minimizes the training error Eq. (3) in an iterative greedy fashion.

Now let

$$\beta_t = \sum_i w_i(t) y_i h_t(x_i) = E_{i \sim W(t)}[y_i h_t(x_i)], \quad (6)$$

be the *edge* [19] of the base hypothesis h_t chosen by the base learner at time step t. Here the edge helps define reward functions in the proposed algorithm.

One can show that [20]

$$E_h(h, W) = \sqrt{1 - \beta_t^2}. \quad (7)$$

This implies that the training error of the final classifier is at most $\prod_{t=1}^{T} \sqrt{1 - \beta_t^2}$. This upper bound suggests several possible reward functions. For example, we can define the reward function as

$$r_t(j) = 1 - \sqrt{1 - \beta_t^2(V_j)}, \quad (8)$$

where $\beta_t(V_j)$ is the edge (6) of the classifier chosen by the base learner from source V_j at the tth boosting round. Since $\beta_t^2(V_j) \in [0, 1]$, this reward is between 0 and 1.

It is important to notice that a reward function in logarithm is proposed in [6] that restricts values the edge (6) can take. In contrast, our reward function does not have such a restriction. Also, (8) allows us to establish a sharper bound than the one in [6], as we will see later.

The combined algorithm, called Randomized ShareBoost or rShareBoost, is shown in Algorithm 4. Here, w_t denotes the distribution for sampling examples that is shared by all sensors, while d_t represents the weight for determining the distribution for sampling views.

rShareBoost seems to have departed significantly from the ShareBoost sampling algorithm. In ShareBoost, boosting is executed in parallel by all the information sources. In contrast, rShareBoost performs boosting along the view chosen by the bandit algorithm only. From the viewpoint of computation, rShareBoost is much more efficient, i.e., it is a fraction $(1/M)$ of the time required for

```
rShareBoost ($\alpha > 0$, $\gamma \in (0,1]$, $\{(x_i, y_i)\}_{i=1}^n$)
1. $w_1(i) = \frac{1}{n}$, $i = 1, \cdots, n$. $d_1(j) = exp(\frac{\alpha\gamma}{3}\sqrt{T/M})$, $j = 1, \cdots, M$.
2. For $t = 1$ to $T$
   (a) $p_t(j) = (1-\gamma)\frac{d_t(j)}{\sum_{k=1}^M d_t(k)} + \frac{\gamma}{M}$
   (b) Let $j$ be the view chosen using $p_t$
   (c) Obtain base classifier $h_t^j$ using distribution $w_t$.
   (d) Calculate: $\epsilon_t^j = \sum_i w_t(i) I(h_t^j(x_i^j) \neq y_i)$. and $r_t(j) \in [0,1]$ (8).
   (e) For $k = 1, \cdots, M$ set
       i. $\hat{r}_t(k) = r_t(k)/p_t(k)$ $(k = j)$, and $0$ $(k \neq j)$
       ii. $d_{t+1}(k) = d_t(k) e^{\frac{\gamma}{3M}(\hat{r}_t(k) + \frac{\alpha}{p_t(k)\sqrt{MT}})}$
   (f) Let $\alpha_t^* = \frac{1}{2}\ln(\frac{1-\epsilon_t^*}{\epsilon_t^*})$, where $\epsilon_t^* = \epsilon_t^j$, $h_t^* = h_t^j$
   (g) Update $w_{t+1}(i) = \frac{w_t(i)}{Z_t} \times \exp(-y_i h_t^*(x_i^*)\alpha_t^*)$, where $Z_t$ is a normalization factor.
3. Output: $H(x) = sign(\sum_{t=1}^T \alpha_t^* h_t^*(x))$
```

ShareBoost. In addition, ShareBoost is greedy, while rShareBoost is not. That is, the probability distribution for sampling training examples for all views is determined solely by the winning source. In rShareBoost, however, the information source selected by Exp3.P may not be the winning view. This provides rShareBoost with an opportunity to examine potential information sources that may prove to be useful.

5 Convergence Analysis of Randomized ShareBoost

We prove a convergence result for the rShareBoost algorithm described in 4.3 in the following theorem. Since rShareBoost is a randomized version of ShareBoost, the result also provides insight into the behavior of the ShareBoost algorithm.

Theorem 2. *Let $V = \{V_1, \cdots, V_M\}$ be a set of M information sources. Suppose that there exists an information source $V_i \in V$ and a constant $0 < \rho <= 1$ such that for any distribution over the training data set S, the base learner returns a base classifier from V_{i^+} with an edge $\beta_{V_{i^+}} \geq \rho$. Then, with probability at least $1-\delta$, the training error (3) of rShareBoost will become 0 after at most in time polynomial in (besides other parameters).*

$$(\log(M/\delta), 1/\rho, \log n),$$

where input parameters to rShareBoost are set to

$$\gamma = \min\{3/5, 2\sqrt{(3M\log M)/(5T)}\}$$

and

$$\alpha = 2\sqrt{\log(MT/\delta)},$$

as required by Theorem 3. More precisely,

$$T = \max\left(\log^2 \frac{M}{\delta}, \left(\frac{2C}{\rho-2}\right)^4, \frac{2\log n}{\rho}\right). \tag{9}$$

where $C = \sqrt{32M} + \sqrt{27M\log M} + 16$.

Proof. The arguments are along the line of the proof of Theorem 1 in [6]. Once the number of sensor sources is given, M becomes constant. Let

$$r^* = \max_i \sum_{t=1}^{T} r_t(i) \tag{10}$$

be the reward of the optimal arm, retrospectively. We can denote this arm by $i^* = \arg\max_i \sum_{t=1}^{T} r_t(i)$. Thus r^* can be written as $r^* = \sum_{t=1}^{T} r_t(i^*)$. Notice that i^* may not be the same as arm i^+ returned by the weak learner. Since i^* is the best arm, it follows that

$$r^* \geq \sum_{t=1}^{T} r_t(i^+). \tag{11}$$

We first upper bound the logarithm of the exponential margin loss (4). From (4) and (5), we have

$$\log(E_H(H, W_1)) = \log\left(\prod_{t=1}^{T} E_h(h_t, W_t)\right) = \sum_{t=1}^{T} \log(E_h(h_t, W_t)).$$

¿From (7), (8) and $\log x \leq x - 1$, we can show that

$$\sum_{t=1}^{T} \log(E_h(h_t, W_t)) \leq \sum_{t=1}^{T} -r_t(j_t). \tag{12}$$

Let $Z = 4\sqrt{MT\log\frac{MT}{\delta}} + 4\sqrt{\frac{5}{3}MT\log M} + 8\log\frac{MT}{\delta}$. From Theorem 1 and (10), we can bound (12) as follows $\sum_{t=1}^{T} -r_t(j_t) \leq -Tr^* + Z \leq -\sum_{t=1}^{T} r_t(i^+) + Z = \sum_{t=1}^{T}(\sqrt{1-\beta_t^2(V_{j_t})} - 1) + Z$, where the second inequality follows from (11), and the equality follows from (8). We therefore have

$$\sum_{t=1}^{T} -r_t(j_t) \leq \sum_{t=1}^{T}(\sqrt{1-\rho^2} - 1) + Z \tag{13}$$

where (13) follows from the assumption that $\beta_t(j^+) \geq \rho$. If we write

$$\sqrt{1-\rho^2} = \sqrt{(1-\rho)(1+\rho)} = \sqrt{(1-\rho) + (1-\rho)\rho},$$

we can show that

$$\sqrt{1-\rho^2} \leq 2 - \rho, \tag{14}$$

where we use the following facts: $\sqrt{x+y} \le \sqrt{x}+\sqrt{y}$ for any $x, y \ge 0$, $\sqrt{(1-x)x} \le \sqrt{1-x}$ for $0 \le x \le 1$, and $2xy \le x^2 + y^2$ for any x, y. Combining (13) and (14), we have

$$\sum_{t=1}^{T} \log(E_h(h_t, W_t)) \le \sum_{t=1}^{T}(1-\rho) + Z. \tag{15}$$

Furthermore,

$$Z \le 4\sqrt{MT(\log\frac{M}{\delta} + \sqrt{T})} + 4\sqrt{\frac{5}{3}MT\log M} + 8\log\frac{M}{\delta} + 8\sqrt{T} \tag{16}$$

$$\le 4\sqrt{MT(\sqrt{T}+\sqrt{T})} + 4\sqrt{\frac{5}{3}MT\log M} + 8\sqrt{T} + 8\sqrt{T} \tag{17}$$

$$= T^{3/4}\sqrt{32M} + T^{1/2}\left(\sqrt{\frac{80}{3}M\log M} + 16\right) \tag{18}$$

$$\le T^{3/4}\left(\sqrt{32M} + \sqrt{27M\log M} + 16\right), \tag{19}$$

where the first inequality follows from $\sqrt{T} > \log T$; the second inequality follows from $\sqrt{T} > \log T$ and $T > \log^2 \frac{M}{\delta}$; and the last inequality follows from the fact that $T > 1$ and $\frac{80}{3} < 27$. Now let $C = \sqrt{32M} + \sqrt{27M\log M} + 16$. Thus, $\sum_{t=1}^{T} \log(E_h(h_t, W_t)) \le T(1 - \rho + T^{-1/4}C) \le -\frac{1}{2}T\rho$, where the last inequality follows from $T > (\frac{2C}{\rho-2})^4$ (9).

In [20], it is shown that $E_H(H, W_1)$ (4) upper bounds Err (3). That is $Err \le E_H(H, W_1)$. Thus, we have $Err \le \exp(-\frac{1}{2}T\rho)$. If we set Err to be less than $\frac{1}{n}$, it must be zero. Therefore, if we let

$$\exp(-\frac{1}{2}T\rho) \le \frac{1}{n},$$

and combine with (9), we obtain the time bound stated in the theorem.

Notice that because of (8), our bound is $O(\frac{1}{\rho^4})$ in terms of ρ in the worst case, which is sharper than the time complexity of $O(\frac{1}{\rho^6})$ established in [6].

There are a number of reasons why the above bound makes sense. As noted in [6], the most significant is that the regret bound for Exp3.P does not depend on n. Another observation is that rShareBoost is a boosting algorithm. Thus, it should output a strong classifier H with zero training error after $T = O(\log n)$ iterations, when its base learner is capable of returning a base classifier having edge (6) $\beta_t \ge \rho$ for given $\rho > 0$. Several boosting algorithms meet this condition [20]. Notice that rShareBoost is PAC learnable because its time complexity is polynomial in $\log \frac{1}{\delta}$.

Both the number of information sources M and the quality that each source can provide in terms of ρ are involved in the analysis. When M is large, a large number of trials must be taken to gather information so that the best information source can be properly identified. On the other hand, when sensor sources are unable to provide reliable information, we must lower ρ, hence a reduced edge

Fig. 1. FERET Sample images

(recall large edge values imply large asymptotic margins [19]). Thus, a large number of iterations are required to build enough base classifiers to create a strong final classifier. In practice, a trade-off between the competing goals has to be made.

It is important to note that the theorem states that when views can provide useful information (a large edge, thus a large ρ), rShareBoost can quickly identify those views. That is, the probability of choosing those views to build base classifiers increases exponentially, which can be particularly useful in environments with noise, thereby avoiding unproductive computations. It addresses one of the important issues facing fusion: Can an algorithm automatically switch between subsets of information sources that are most decision-relevant? The above analysis shows that rShareBoost can.

6 Experiments

We have carried out empirical study evaluating the performance of the proposed algorithm. As comparison, the following methods are evaluated. (1) ShareBoost (Algorithm 1), and (2) rShareBoost. (Algorithm 4), (3) iBoost (The boosting with independent sampling distribution). In iBoost, re-sampling weights of training examples are independent for each view. Similar to ShareBoost, a base classifier from the view having the largest α value is selected at each boosting round. All these three algorithms employ a Naive Bayes learner to build base classifiers, and the Gaussian distribution is used for marginal's. The number of base classifiers is 150. (4) The semi-definite programming (SDP) algorithm [13], where kernel functions are Gaussian. (5) The AdaBoost-MV algorithm, where AdaBoost is applied to each view independently and final classification is determined by majority vote. (6) The AdaBoost-Ca algorithm, where AdaBoost is applied to a concatenation of all views. (7) The majority vote (SVM-MV) algorithm, where SVMs are used as component classifier. (8) The stacked generalization (Stacking) algorithm [24]. Stacking is very similar to SVM-MV. The only difference is that, instead of majority vote, the final combiner is another SVM.

Table 1. Average accuracy: Noise free

Data	*Face*	*Gender*	*Glass*	*Gene*
ShareBoost	0.76	0.87	0.74	0.67
rShareBoost	0.76	0.87	0.73	0.66
iBoost	0.75	0.85	0.72	0.60
SDP	0.70	0.45	0.48	0.60
AdaBoost-MV	0.74	0.77	0.71	0.62
AdaBoost-Ca	0.74	0.84	0.70	0.61
SVM-MV	0.70	0.58	0.58	0.64
Stacking	0.70	0.58	0.58	0.63

Table 2. Average accuracy: 30% noise

Data	*Face*	*Gender*	*Glass*	*Gene*
ShareBoost	0.72	0.63	0.63	0.56
rShareBoost	0.73	0.63	0.64	0.56
iBoost	0.65	0.53	0.56	0.54
SDP	0.70	0.46	0.52	0.54
AdaBoost-MV	0.68	0.51	0.58	0.53
AdaBoost-Ca	0.65	0.51	0.58	0.52
SVM-MV	0.69	0.57	0.59	0.53
Stacking	0.70	0.58	0.58	0.57

Three FERET image data sets and one gene data set are used here. The problems are (1) Face detection, (2) Gender classification, and (3) detection of Glasses on faces. Sample images are shown in Fig. 1.

For the face and gender data, each image is represented by three poses in terms of eigenfaces extracted from three head orientations: 1) frontal, 2) half left, and 3) half right profiles. The non-face images are blacked out faces. In the glass detection experiment, each image is represented by three types of features extracted from only one pose of an individual, namely (1) eigenfaces, (2) Canny edges, and (3) wavelet coefficients. Each dataset has 101 samples and each view has 101 dimensions after applying PCA.

The gene data set is from the Yeast Database (CYGD) [16]. The task is to combine different sources to determine membrane vs non-membrane proteins. Three sources are derived from BLAST and Smith-Waterman genomic methods, and from gene expression measurement. The dataset has 100 examples and the number of dimensions after applying PCA is 76, 74 and 64, respectively. These dimensions explain 90% variance in the data.

Ten-fold cross-validation was used for model selection. For rShareBoost, we set α to 0.15 and γ to 0.3, respectively, as suggested in [6]. The results are averaged over 30 runs (60% training and 40% testing). Table 1 shows the average accuracy (noise free). In terms of paired t–test with a 95% confidence level, for the Face data, ShareBoost and rShareBoost are significantly better than SDP, SVM-MV and Stacking. For the Gender data, ShareBoost and rShareBoost are better than

SDP, AdaBoost-MV, SVM-MV and Stacking. For the Glass data, ShareBoost and rShareBoost significantly outperform SDP, AdaBoost-Ca, SVM-MV and Stacking. For the Gene data, ShareBoost and rShareBoost outperform the rest, except SVM-MV and Stacking.

To examine sensitivity to noise, we randomly added 30% noise to the class label of the training data for all three views by *flipping* a label from one class to another. Flipping labels to generate noise produces similar effect to that produced by poor views in terms of overlapping classes. Table 2 shows the average classification accuracy over 30 runs in the noisy case. For *Face*, ShareBoost and rShareBoost ourperform iBoost, AdaBoost-MV, and AdaBoost-Ca. For Gender and Glass, ShareBoost and rShareBoost are significantly better than the rest. The results show that ShareBoost and rShareBoost are more than robust against noise than the competing methods. Overall, rShareBoost is similar to ShareBoost in performance. However, rShareBoost is much more efficient computationally than ShareBoost.

Note that we also carried out experiments using feature noise (white noise). Every method performed better. However, their relative performances remained the same as in the label noise case. Label noise seems to create harder problems. It most likely creates problems with overlapping classes, while feature noise (white noise) may not.

Notice that the number of base classifiers for both ShareBoost and rShare-Boost is 150. However, for each base classifier ShareBoost requires the amount

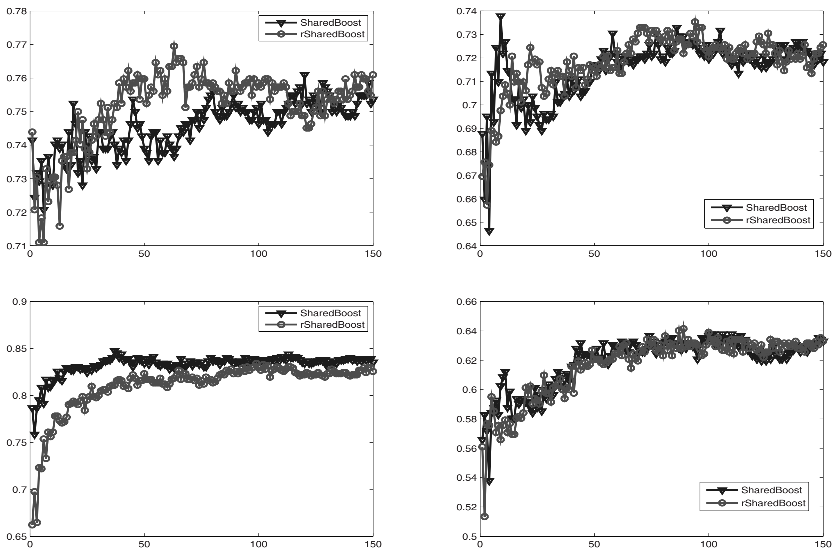

Fig. 2. Average accuracy as a function of the number of base classifiers achieved by ShareBoost and rShareBoost over 30 runs on the Face and Gender data. Left column: the noise-free case. Right column: the 30% noisy case.

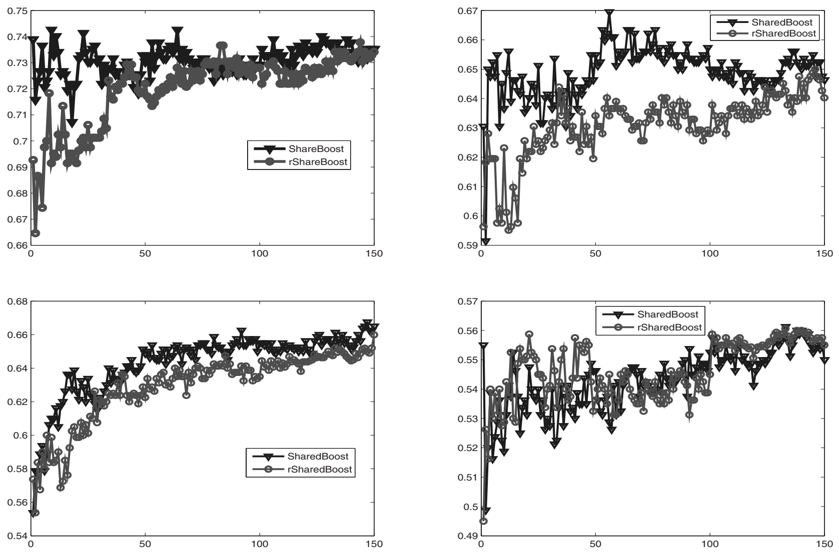

Fig. 3. Average accuracy as a function of the number of base classifiers achieved by ShareBoost and rShareBoost over 30 runs on the Glass and Gene data. Left column: the noise-free case. Right column: the 30% noisy case

of computation that is three times that required for rShareBoost. The fact that ShareBoost and rShareBoost registered similar performance on the problems examined shows that rShareBoost is more efficient computationally than ShareBoost. Figure 2 shows the average accuracy as a function of the number of base classifiers registered by ShareBoost and rShareBoost over 30 on the Face and Gender data, while Figure 3 shows the average accuracy by the two methods on the Glass and Gene data. In the figures, the left column shows the noise-free case, and the right column shows the 30% noisy case. The results show clearly that choosing a winning view probabilistically as in rShareBoost can be as effective as the winner-take-all strategy employed ShareBoost.

7 Summary

We have developed the ShareBoost algorithm for boosting for multi-view learning. The ShareBoost algorithm has been shown to be robust against noise by limiting noise to individual views. We have also developed a randomized version of the algorithm, rShareBoost, that can be cast within a multi-armed bandit framework. This formulation allows us to state its convergence and show that with high probability the rShareBoost algorithm judiciously seeks out decision relevant views for making predictions. rShareBoost has achieved a performance similar to ShareBoost at a fraction $(1/M)$ of time required for ShareBoost. We have provided the experimental results that validate our theoretical analysis.

We plan on exploring other base classifiers such as decision stumps in ShareBoost to examine its performance, and investigating its mechanism for exploiting interplays between multiple views for learning.

References

1. Audibert, J.-Y., Munos, R., Szepesvari, C.: Exploration-exploitation tradeoff using variance estimates in multi-armed bandits. Theor. Comput. Sci. 410(19), 1876–1902 (2009)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine Learning 47, 235–256 (2002)
3. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.: The non-stochastic multi-armed bandit problem. SIAM Journal on Computing 32(1), 48–77 (2002)
4. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: In Proceedings of the Eleventh Annual Conference in Computational Learning Theory
5. Busa-Fekete, R., Kegl, B.: Accelerating adaboost using ucb. In: KDDCup (JMLR W&CP), pp. 111–122 (2009)
6. Busa-Fekete, R., Kegl, B.: Fast boosting using adversarial bandits. In: Proceedings of International Conference on Machine Learning (2010)
7. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, Cambridge (2006)
8. Culp, M., Michailidis, G., Johnson, K.: Tri-training: Exploiting unlabeled data using three classifiers. IEEE Transactions on Knowledge and Data Engineering 17, 1529–1541 (2005)
9. Fawcett, T., Niculescu-mizil, A.: Technical note: Pav and the roc convex hull. Machine Learning 68, 97–106 (2007)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and Systems Science 55, 119–139 (1997)
11. Kittler, J.: Combining classifiers: A theoretical framework. Pattern Analysis and Applications 1, 18–27 (1998)
12. Kuncheva, L.I., Bezdek, J.C., Duin, R.P.W.: Decision templates for multiple classifier fusion: An experimental comparison. Pattern Recognition 34, 299–314 (2001)
13. Lanckriet, G.R.G., Deng, M.H., Cristianini, N., Jordan, M.I., Noble, W.S.: Kernel-based data fusion and its application to protein function prediction in yeast. In: Proceedings of the Pacific Symposium on Biocomputing, vol. 9, pp. 300–311 (2004)
14. Maturana, J., Fialho, A., Saubion, F., Schoenauer, M., Sebag, M.: Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In: Proceedings of IEEE ICEC, pp. 365–372 (2009)
15. Mesmay, F.D., Rimmel, A., Voronenko, Y., Puschel, M.: Bandit-based optimization on graphs with application to library performance tuning. In: Proceedings of International Conference on Machine Learning, pp. 729–736 (2009)
16. Mewes, H.W., Frishman, D., Gruber, C., Geier, B., Haase, D., Kaps, A., Lemcke, K., Mannhaupt, G., Pfeiffer, F., Schüller, C., Stocker, S., Weil, B.: Mips: a database for genomes and protein sequences. Nucleic Acids Research 28, 37–40 (2000)
17. Robbins, H.: Some aspects of the sequential design of experiments. Bulletin American Mathematical Society 55, 527–535 (1952)

18. Ross, A., Jain, A.K.: Multimodal biometrics: an overview. In: Proceedings of 12th European Signal Processing Conference, pp. 1221–1224 (2004)
19. Rudin, C., Schapire, R., Daubechies, I.: Precise statements of convergence for adaboost and arc-gv. Contemporary Mathematics 443 (2007)
20. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence rated predictions. Machine Learning 3(37), 297–336 (1999)
21. Viola, P., Jones, M.: Fast and robust classification using asymmetric adaboost and a detector cascade. In: Advances in Neural Information Processing Systems, vol. 14 (2002)
22. Wang, W., Hua Zhou, Z.: On multi-view active learning and the combination with semi-supervised learning. In: Proceedings of the 25th International Conference on Machine Learning (2008)
23. Wang, W., Hua Zhou, Z.: A new analysis of co-training. In: Proceedings of the 25th International Conference on Machine Learning (2010)
24. Wolpert, D.H.: Stacked generalization. Neural Networks 5, 241–259 (1992)
25. Zhang, D., Wang, F., Zhang, C., Li, T.: Multi-view local learning. In: Proceedings of the 23rd National Conference on Artificial Intelligence (AAAI), pp. 752–757 (2008)