Boosting Information Fusion

Costin Barbu[1]
MIT Lincoln Laboratory
Lexington, MA 02420
barbu@ll.mit.edu

Jing Peng
Computer Science Department
Montclair State University
pengj@mail.montclair.edu

Guna Seetharaman[1,2]
Information Directorate, AFMC
AFRL/RITB, Rome, New York
Gunasekaran.Seetharaman@rl.af.mil

Abstract – *Ensemble methods provide a principled framework for building high performance classifiers and representing many types of data. As a result, these methods can be useful for making inferences in many domains such as classification and multi-modal biometrics. We introduce a novel ensemble method for combining multiple representations (or views). The method is a multiple view generalization of AdaBoost. Similar to AdaBoost, base classifiers are independently built from each representation. Unlike AdaBoost, however, all data types share the same sampling distribution as the view whose weighted training error is the smallest among all the views. As a result, the most consistent data type dominates over time, thereby significantly reducing sensitivity to noise. In addition, our proposal is provably better than AdaBoost trained on any single type of data. The proposed method is applied to the problems of facial and gender prediction based on biometric traits as well as of protein classification. Experimental results show that our method outperforms several competing techniques including kernel-based data fusion.*

Keywords: AdaBoost, data fusion, stacking, semi-definite programming.

1 Introduction

Classifiers employed in real world scenarios must deal with various adversities such as noise in sensors, intra-class variations, restricted degrees of freedom and in some cases spoof attacks. It is often helpful to develop classifiers that rely on data from various sources for classification. Such algorithms, known in the literature as multimodal classification algorithms require a clever way of fusing the various sources of information. A robust data fusion strategy compensates for any errors in the feature extraction process due to the adversities faced by a classifier.

We are given a set of training examples: $X = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_n, y_n)\}$, and M disjoint features for each example $x_i = \{x_i^1, x_i^2 ..... x_i^M\}$, where $x_i^j \in \Re^{q_j}$, and $y_i \in \mathcal{Y} = \{-1, +1\}$. Each member x_i^j in the set x_i is known as a *view* of example x_i. In this case, it is the jth view of example x_i. For instance, when three sensors such as radar (Radr), infrared (IR) and visible (Vis) are used to capture an event, each example x_i can be thought of as a set of three views, each consisting of three features obtained from the intensities of radar, infrared and visible components. In this case, the number of views will be three and we can represent the three views of the example x_i as $\{x_i^{Radr}, x_i^{IR}, x_i^{Vis}\}$. 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 $q = \sum_{j=1}^{M} q_j$.

In this paper, we present a novel method for fusing multiple representations of data with boosting. Our method is a multiple view generalization of AdaBoost [1]. Similar to AdaBoost, base classifiers are independently built from each view. Furthermore, all views share the same sampling distribution as the view whose weighted training error is minimum among all the views. This allows the most consistent data type (view) to dominate over time, thereby significantly reducing sensitivity to noise. In addition, since the final ensemble contains classifiers that are trained to focus on different views of the data, better generalization performance can be expected. We support this argument with empirical evaluation performed on FERET facial images [2] and CYGD genomic data [3].

2 Related Work

Considerable research in the pattern recognition field is focused on fusion rules that aggregate the outputs of the first level experts and make a final decision. Various techniques for fusion of expert observations such as linear weighted voting, the Naive Bayes classifiers, the kernel function approach, potential functions, decision trees or multilayer perceptrons have been proposed in recent years [4, 5]. Other approaches are based on bagging, boosting, and arching clas-

[1]The research described here and the conclusions are that of the authors. The document does not in any way represent the policies of the MIT Lincoln Laboratory or US Air Force Research Laboratory.

[2]This research paper has been reviewed and cleared for public release, distribution A, 88ABW-2010-2807.

sifiers [1, 6, 7]. Comprehensive surveys of various classifier fusion studies and approaches can be found in [8, 9, 10].

In [11] Wolpert proposes stacked generalization that is defined as any scheme that feeds data from one set of classifiers to another before making a final decision. The data that feeds up the net of the classifiers is provided by multiple partitionings into two subsets of the original learning set. These pairs of subsets are further employed to gather information about the bias of the original classifier(s) with respect to the learning set. The bias of the constituent classifiers with respect to the learning set is estimated and corrected by the stacked generalization. In information fusion, it is equivalent to forming a linear combination of the classification results of the constituent classifiers.

Lanckriet et al. [12] introduce a kernel-based data fusion 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 semidefinite programming. That is, given a set of kernel matrices $\mathcal{K} = \{K_1, K_2, ..., K_m\}$, the optimal combination $K = \sum_{i=1}^{m} \mu_i K_i$ can be obtained by optimizing coefficients μ_i through semi-definite programming.

There is a close relationship between our technique and that of Viola and Jones [13]. If we have a single view and base classifiers are allowed to include features as well, then both techniques reduce to standard AdaBoost. When noise exists, however, the two techniques diverge. In the case of Viola and Jones, it 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. By restricting noisy, thus "difficult," examples to individual views, the mass of sampling probability on these examples will be restricted in our technique. This is possible because probability mass will be determined by those views having less noise.

3 Boosting Using Shared Sampling Distribution

AdaBoost has been shown to improve the prediction accuracy of weak classifiers using an iterative weight update process [1]. The technique combines weak classifiers (classifiers having classification accuracy slightly better than chance) in a weighted vote fashion giving an overall strong classifier. Detailed explanation of the AdaBoost algorithm is skipped here for brevity, interested readers may refer to [14, 15, 16] for more on AdaBoost.

One of the ways boosting may be used for classifier fusion would be to run boosting separately on each view, obtain separate ensembles for each view, and then take a majority vote among the ensembles when presented with a test example. In this case, separate training of classifiers is needed for each view and the sampling distributions of the data points are also independent.

We propose a different yet simple approach. Our approach performs separate training for each view. However, the weight distribution of training examples is shared among all the views at each boosting round. The main steps of the proposed algorithm are shown in Algorithm 1.

Algorithm 1: Boosting With Shared Sampling Distribution (BSSD)

1. Input: $z_0^j = \{x_i^j, y_i)\}_{i=1}^n, j = 1, \cdots, M$.
2. Initialization: $W_1 = \{w_1(i) = \frac{1}{n}\}_{i=1}^n$.
3. For $k = 1$ to k_{max}
   (a) Sample z_k^j from z_0^j using the distribution W_k.
   (b) Compute hypothesis h_k^j from z_k^j for each view j.
   (c) Calculate error ϵ_k^j: $\epsilon_k^j = P_{i \sim W_k}[h_k^j(x_i^j) \neq y_i]$
   (d) If for each view: $\{\epsilon_k^j\}_{j=1}^M \leq 0.5$, select h_k^* corresponding to $\epsilon_k^* = \min_j\{\epsilon_k^j\}$.
   (e) Calculate $\alpha_k^* = \frac{1}{2} ln(\frac{1-\epsilon_k^*}{\epsilon_k^*})$.
   (f) Update $w_{k+1}(i) = \frac{w_k(i)}{Z_k^*} \times e^{-h_k^*(x_i^*) y_i \alpha_k^*}$, where Z_k^* is a normalizing factor.
4. Output: $F(x) = \sum_{k=1}^{k_{max}} \alpha_k^* h_k^*(x^*)$.
5. Final hypothesis: $H(x) = sign(F(x))$.

Input to the algorithm are the M views of n training examples. The algorithm produces as output a classifier that fuses data from all the views. In the initialization step, all the views for a given training example are initialized with the same weight. Notice that our algorithm works in multimodal fusion where data types might not be compatible or of fixed size vectors. We only require that the number of training examples from each modality be the same.

3.1 Simple Illustration

We use a simple two class example in four dimensions to help explain how BSSD works and highlight the difference between AdaBoost and BSSD. This example is taken from the Iris data [17] with two classes (Versicolour and Virginica). We randomly select 10 examples from each class. Thus, there are total 20 examples, where examples from 1 to 10 are in class Versicolour and examples from 11 to 20 are in class Virginica. For BSSD, there are two views: **sepal** and **petal**. Each view has two attributes. For the sepal view, we have sepal length and sepal width, while for the petal view, we have petal length and petal width. Figure 1 shows the two views.

To make the problem more interesting, we randomly added 30% noise to each view independently by "flipping" the label from one class to another. For the first view, the

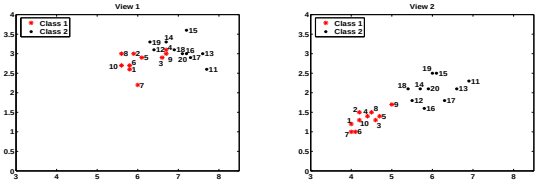

Figure 1: Two views of the Iris data.

noisy examples are: 3, 4, 9, 12, 14 and 19. For the second view, they are: 5, 8, 9, 12, 16 and 18. For the illustration purpose, linear classifiers with weighted least squares fit are used as base classifiers.

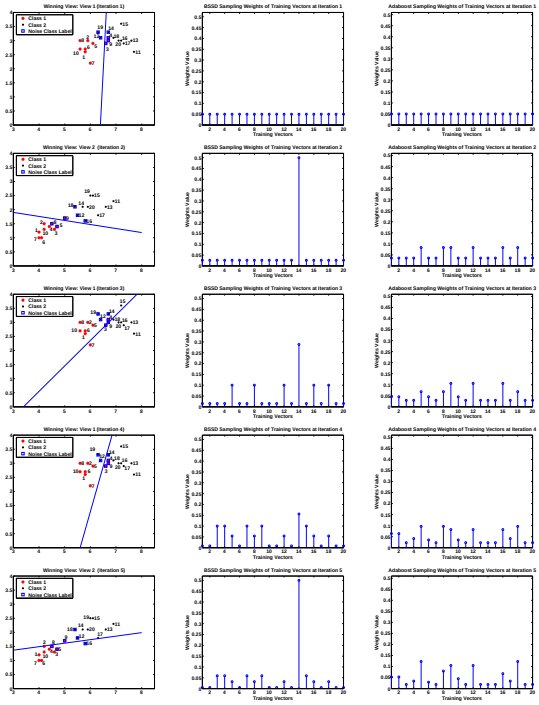

Figure 2: Sampling weights. Left column: Winning views along with decision boundaries. Middle column: Sampling weights computed by BSSD. Right column: Sampling weights computed by AdaBoost.

We performed 50 boosting rounds. The left column in Figure 2 shows the winning views along with decision boundaries computed by BSSD, while the middle column shows the shared sampling weights for the first five iterations. After the first boosting round, the first (sepal) view is the winning view. The base classifier mislabels example 14. Thus, its weight increases, while the weights of the rest decrease. At the next boosting round, the second (petal) view is the winner whose base classifier mislabels examples 5, 8, 12, 16, and 18, but correctly labels example 14. As a result, the sampling weights for examples 5, 8, 12, 16 and 18 are increased, while the weight for example 14 is decreased. Similar observations can be made for the remaining boosting rounds.

What is more interesting to observe is that BSSD does not overemphasize the noisy examples, as evidenced by the shared sampling weights associated with these examples. That is, BSSD places less mass of sampling probability on these difficult examples, with the exception of example 14. The reason is that when a view competes and wins, it contributes its piece of information about an event to the boosting process by forcing losing views to accept its interpretation of the event through shared sampling probability. And as such, the winning view potentially makes "corrections" to sampling probability resulting from overcommitment by the losing views.

For example, when Petal (second) view misclassifies example 5 (one of its noisy examples), it increases its weight. However, when Sepal (first) view competes and wins, it reduces the sampling weight for example 5, because example 5 is not "difficult" as far as Sepal is concerned. This can be seen from boosting round 3 to round 4. As long as the views do not share the same noisy examples (this assumption is reasonable in practice, because information for each view is obtained from independent sources), this mechanism of alternating winning views plays the role in "softening" re-sampling weights for difficult or noisy examples, thereby making BSSD more robust against noise.

We can state this formally in the following lemma.

Lemma 1 *Let j represent the view among M views that has the least amount of noise (e.g., it has the fewest noisy examples) or the best representation in terms of class separability. When the view j wins in the learning process of BSSD, the sampling weights for noisy examples other than those associated with the jth view will be decreased.*

Proof. Let the margin of the training example $z_i = (x_i, y_i)$ be $\theta_i = y_i \sum_{t=1}^{k} c_t^* h_t^*(x_i^*)$, where $c_t^* = \frac{\alpha_t^*}{\sum_{l=1}^{k} \alpha_l^*}$. $w_{k+1}(z_i)$ can be written as [18]

$$w_{k+1}(z_i) = \frac{exp(-\frac{1}{2}\theta_i)^{|\alpha|}}{\sum_{i=1}^{n} exp(-\frac{1}{2}\theta_i)^{|\alpha|}}, \tag{1}$$

where $|\alpha| = \sum_{t=1}^{k} |\alpha_t^*|$. As α increases, examples with smaller margin (e.g., difficult or noisy examples) will receive larger sampling weights. Since $|\alpha|$ increases at least linearly with the number of boosting rounds [18], the noisy examples will receive larger sampling weights. When the jth view wins, it will increase the sampling weights for its noisy or difficult examples, while decreasing the sampling weights for the smaller margin examples that are not part of the jth view. The jth view winning will happen during the learning process because the jth view has the fewest noisy examples, thus an overall large margin or smaller average error, where the rate of error for z_i is given by [18], $err(z_i) = \sum_{t=1}^{k} c_t^* I(y_i \neq h_t^*(x_i^*)) = \frac{1}{2}(1 - \theta_i)$. Here $I(\cdot)$ denotes the indicator function. ■

While the lemma appears to set stronger conditions, BSSD performs much better in practice than the conditions required by the lemma. For example, the above Iris data

experiment shows that even when the views have the same percentage of noise, BSSD will outperform AdaBoost, as long as the views do not share the exact noisy examples.

When the resulting boosted classifier is applied to the original data with noise removed, 80% accuracy is obtained. For comparison, we ran AdaBoost for 50 iterations on the same data, where the noisy examples are: 5, 8, 9, 12, 16 and 18 (e.g., they are identical to those in the second view for BSSD). AdaBoost achieved an accuracy of 70%. The right column in Figure 2 shows Adaboost re-sampling weights for the first five iterations. As expected, AdaBoost consistently placed more mass of sampling probability on the noisy examples, resulting in less accurate performance.

4 Error Bounds

4.1 Tighter Bound on Training Error

Freund and Schapire [14] define the margin of the training example (x_i, y_i) as

$$\theta_i = \frac{y_i F(x_i)}{\sum_{k=1}^{k_{max}} \alpha_k} \Rightarrow y_i F(x_i) = \theta_i \sum_{k=1}^{k_{max}} \alpha_k.$$

Lemma 2 *Given the weighted training errors at iteration k for hypotheses h_k^j corresponding to M views $\epsilon_k^1, \cdots, \epsilon_k^M$, denoting $\epsilon_k^* = min_j\{\epsilon_k^j\}$ and θ_i the margins of the training examples (x_i, y_i). Then for an ensemble of classifiers that fuses M distinct views, the bound on the training error is given by*

$$err_{train}(H) \leq \Pi_{k=1}^{k_{max}} [2\sqrt{\epsilon_k^*(1-\epsilon_k^*)}] - \frac{1}{n} \sum_{i:H(x_i)=y_i} e^{-\theta_i \sum_{k=1}^{k_{max}} \alpha_k^*}.$$

Notice that $\epsilon_k(1-\epsilon_k)$ decreases with ϵ_k for $\epsilon_k \in (0, 0.5]$. Therefore, the lemma states that if we always choose the base classifier at each boosting round having the smallest error rate among all views in the final combined classifier, we can reduce training error faster. This potentially produces a final combined classifier that is less complex due to fewer base classifiers. This can lead to better generalization, especially in noise free data.

4.2 Generalization Error Bound

The generalization error is defined as the probability of misclassifying a new example [14]. Since the final classifier computed by our algorithm is

$$H(x) = sign(F(x)) = sign(\sum_{k=1}^{k_{max}} \alpha_k^* h_k^*(x^*)),$$

the final classifier's output will not be affected by the division of F(x) by a positive quantity, namely $\sum_{k=1}^{k_{max}} \alpha_k^*$. Let us define $f(x) = F(x)/\sum_{k=1}^{k_{max}} \alpha_k^*$.

The generalization error bound for our algorithm is a generalization of the AdaBoost error bound for multiple views. The proof of our bound follows the lines of that introduced by Schapire et al [15]. Given $\mathcal{H}$ the space where the base classifiers are chosen, the function $f(x)$, as defined above, clearly belongs to the convex hull $\mathcal{C}$ of $\mathcal{H}$. $\mathcal{C}$ is the set of mappings that can be generated by taking a weighted average of hypotheses from $\mathcal{H}$: $\mathcal{C}= \{f : x \rightarrow \sum_h a_h h(x) | a_h \geq 0; \sum_h a_h = 1\}$

Throughout the rest of the paper the notation $P_{(x,y)\sim W}[A]$ will mean the probability of the event A when the example (x,y) is sampled according to W, and $P_{(x,y)\sim S}[A]$ will mean the probability with respect to sampling uniformly at random an example from the training set. Their abreviation will be $P_W[A]$ and $P_S[A]$. The expected value will be denoted $E_W[A]$ and $E_S[A]$.

Theorem 3 *([15]) Let S be a sample of n examples chosen independently at random according to W. Assume that the base hypothesis space $\mathcal{H}$ has the VC-dimension d and let $\delta > 0$. Then with probability at least $1-\delta$ over the random choice of the training set S, every weighted average function $f \in \mathcal{C}$ satisfies the following bound for all $\theta > 0$*

$$P_W[yf(x) \leq 0] \leq P_S[yf(x) \leq \theta] + O(\frac{1}{\sqrt{n}} \sqrt{\frac{dlog^2(\frac{n}{d})}{\theta^2} + log(\frac{1}{\delta})}).$$

The empirical error bound for the shared sampling distribution based algorithm for fusion of base classifiers from M views is provided by Theorem 4.

Theorem 4 *Given the weighted training errors at iteration k for hypotheses corresponding to the M views $\epsilon_k^1, \cdots, \epsilon_k^M$ and denoting $\epsilon_k^* = min_j\{\epsilon_k^j\}$. Then for any θ, we have that*

$$P_S[yf(x) \leq \theta] \leq \Pi_{k=1}^{k_{max}} [2\sqrt{\epsilon_k^{*1-\theta}(1-\epsilon_k^*)^{1+\theta}}].$$

More recent results [19] give the following error bound

$$P_W[yf(x) \leq 0] \leq \inf_{\theta \in (0,1]} \{P_S[yf(x) \leq \theta] + \frac{C}{\theta}\sqrt{\frac{d}{n}} + \sqrt{\frac{\log\log_2(2\theta^{-1})}{n}}\} + \frac{\sqrt{\frac{1}{2}\log\frac{2}{\delta}} + 2}{\sqrt{n}}$$

where C is a constant. This bound slightly improves the bound established in Theorem 3, and will be used later in our convergence analysis. Notice that $\epsilon_k^{1-\theta}(1-\epsilon_k)^{1+\theta}$ decreases with decreasing ϵ_k, when $\epsilon_k \in (0, 0.5]$ for any given $\theta > 0$. Thus for a fixed sample size n, BSSD achieves better generalization performance.

5 Experimental Evaluation

We have carried out experimental studies evaluating the performance of the proposed data fusion algorithm on a number of data sets. The following competing methods are compared.

- The BSSD algorithm with the Naive Bayes classifiers [20] as base classifiers for boosting. The Gaussian model is used for the marginal distributions in the Naive Bayes classifier.
- The boosting with independent sampling distribution (BISD) with Naive Bayes classifiers. This algorithm is a variant of BSSD, where re-sampling weights of training examples are independent for each view.

- The stacking (Stacking) algorithm [11] with SVMs as a back-end generalizer.
- The semidefinite programming (SDP) algorithm [12], where kernel functions along each view can be Gaussian: $k(x,y) = exp(-\|x-y\|^2/\sigma^2)$, polynomial: $k(x,y) = (x \cdot y + 1)^2$, or linear: $k(x,y) = x \cdot y$. Here $\cdot$ denotes dot product. We wish to thank Lanckriet for providing us with Matlab code for SDP.
- The majority vote (MV) algorithm, where SVMs are used as component classifier along each view.

Ten-fold cross-validation was used for model selection (or choosing procedural parameters). For SVMs, two procedural parameters: σ and C , the soft margin parameter, take values in $[10^{-2}, 10^2]$ and C in $[10^{-2}, 10^2]$, respectively. All results reported here are averaged over 20 runs, where each run splits data into 60% training and 40% testing.

Figure 3: Sample images from FERET facial images database.

5.1 Experimental Data

Two real examples are used to evaluate the proposed technique and its competitors.

5.1.1 FERET Facial Image Data

Three binary class data sets have been generated from the FERET database of facial images [2]. The three classification problems are (1) Face classification, (2) Gender classification, and (3) detection of Glasses (spectacles) on faces. Sample images from the FERET database are shown in Figure 3.

For the face and gender classification experiments, each image is represented by three views in terms of eigenfaces extracted from three head orientations (poses): (1) frontal, (2) half left profile and (3) half right profile. The non-face images in the face classification data set 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 filter detected edges [21], and (3) wavelet coefficients [22]. Each dataset has 101 pictures and the number of dimensions after applying principal component analysis is 101 for each view.

5.1.2 CYGD Genomic Data

The third data set is the yeast genomic data that can be obtained from the MIPS Comprehensive Yeast Genome Database (CYGD) [3]. The task consists of combining different data sources for gene classification (membrane vs non-membrane proteins). There are three sources of data–considered "views" in our framework–that are derived from BLAST and Smith-Waterman genomic methods, and from gene expression measurements. The dataset has 100 examples and the number of dimensions after applying principal component analysis to each of the three views is 76, 74 and 64, respectively. These dimensions explain 90% variance in the data.

Table 1: Results for face classification

Method	A_{V_1}	A_{V_2}	A_{V_3}	A_{fus}	Sig.
MV	0.709	0.709	0.700	0.700	yes
Stack	0.709	0.709	0.700	0.717	yes
SDP	*Gauss*	*Gauss*	*Gauss*	0.698	yes
SDP	*Poly*	*Gauss*	*Gauss*	0.698	yes
SDP	*Poly*	*Linear*	*Gauss*	0.698	yes
BISD	0.62	0.60	0.58	0.750	no
BSSD	0.623	0.615	0.564	0.763	
Results for gender classification					
MV	0.555	0.490	0.549	0.539	yes
Stack	0.555	0.490	0.549	0.578	yes
SDP	*Gauss*	*Gauss*	*Gauss*	0.444	yes
SDP	*Poly*	*Gauss*	*Gauss*	0.450	yes
SDP	*Poly*	*Linear*	*Gauss*	0.442	yes
BISD	0.64	0.53	0.56	0.859	no
BSSD	0.633	0.538	0.578	0.865	
Results for glass detection					
MV	0.560	0.613	0.528	0.576	yes
Stack	0.560	0.613	0.528	0.672	yes
SDP	*Gauss*	*Gauss*	*Gauss*	0.457	yes
SDP	*Poly*	*Gauss*	*Gauss*	0.439	yes
SDP	*Poly*	*Linear*	*Gauss*	0.477	yes
BISD	0.57	0.56	0.56	0.720	no
BSSD	0.591	0.594	0.546	0.740	

5.2 Experimental Results

Tables 1 and 2 show the average results registered by the competing methods on the tasks. The average accuracy of individual classifiers from each view before fusion is shown in the columns A_{V_1}, A_{V_2} and A_{V_3} for the facial data (Table 1) and in the columns A_B, A_{SW} and A_G for the genomic data (Table 2). Also, the paired *t*–test with a 95% confidence level was performed to determine if the difference in performance between BSSD and the competing techniques (Stacking, MV, SDP and BISD) is statistically significant, and is shown in the last column.

These results demonstrate that our proposed fusion algorithm significantly outperforms the competing techniques (except BISD, where no significant difference is observed) on these noise free problems that we have experimented with. We note however that BSSD does outperform BISD on the Yeast genomic dataset. In particular, our simple technique registered superior performance over mathematically sophisticated techniques such as SDP. We will explore later mathematical arguments behind this.

Table 2: Results for Genomic Data

Methods	A_B	A_SW	A_G	A_{fus}	Sig.
Stack	0.62	0.58	0.59	0.63	no
MV	0.62	0.58	0.59	0.638	no
SDP	$Gauss$	$Gauss$	$Gauss$	0.60	yes
BISD	0.50	0.54	0.52	0.60	yes
BSSD	0.52	0.55	0.54	0.65	

Notice that in some examples, the average combined accuracy is worse than that obtained from a single view (i.e., majority vote and stacking). In the case of the majority vote, if the majority makes a wrong decision, so does the combined decision, resulting in a decrease in accuracy. Similar observations were made in [23, 15]. For stacking, the average fusion accuracy is obtained by the (stacked) classifier whose input is class labels generated by the component classifiers along each view. If the component classifiers produce poor results, the input to the stacked classifier will be noisy or poorly represented. This fact helps explain why the stacked accuracy is worse than the classification accuracy of some of the component classifiers.

5.3 Robustness against Noise

Robustness against noise is a key feature for any data fusion algorithm that must operate across the full range of conditions and scenarios the system is anticipated to encounter. In Tables 3, 4 and 5 we compared the robustness of BSSD and the competing techniques against noise (average fusion accuracies over 20 runs). We randomly added noise to the class label of the training data sets on two or all three views at various levels: 10%, 20% and 30% by "flipping" the label from one class to another. Flipping labels to generate noise produces similar effect to that produced by poor representations (features).

In the case of noisy views, noise is encoded in the base kernel matrices in SDP, which in turn severely degrades its performance. On the other hand, our approach simply relies more on other available data sources if one of them is not reliable. In fact, our technique prefers views that are better represented in terms of class separability. Recall that at each iteration, the resampling and weight update in BSSD are performed using a shared distribution. That is, the weights for all views of a given training example are updated according to the opinion of the winning classifier (having the smallest average training error). This winning classifier is unlikely to come from a view that either is poorly represented or provides little information about the two classes. As a result, our algorithm relies heavily on data sources that best separate the two classes, which is important for building the optimal linear combination of base classifiers.

Table 3: Robustness of BSSD vs MV

$Data$ Set	$Noise$ 2 Views	MV A_{fus}	BSSD A_{fus}	$Stat.$ Signif
Face	10%	0.70	0.75	yes
	20%	0.71	0.75	yes
	30%	0.70	0.74	yes
Gender	10%	0.57	0.75	yes
	20%	0.58	0.69	yes
	30%	0.58	0.66	yes
Glass	10%	0.59	0.73	yes
	20%	0.58	0.70	yes
	30%	0.58	0.67	yes
$Data$ Set	$Noise$ 3 Views	MV A_{fus}	BSSD A_{fus}	$Stat.$ Signif
Face	10%	0.70	0.74	yes
	20%	0.72	0.73	no
	30%	0.69	0.70	no
Gender	10%	0.55	0.71	yes
	20%	0.57	0.61	yes
	30%	0.57	0.54	no
Glass	10%	0.60	0.73	yes
	20%	0.60	0.67	yes
	30%	0.59	0.61	no

We state that, similar to [12], cross-validation was not used to choose kernel parameters for SDP in our experiments. The idea is that SDP learns the optimal coefficients, μ of kernel matrices that determine how much each view contributes to the final decision and to encourage diversity. It is highly likely that cross-validated kernel parameters will help increase accuracy in classification. We suspect that this may be one of the causes for the sub-optimal performance of SDP on 2 out of 3 facial tasks. We will have more to say later regarding SDP in the "Discussions" section.

5.4 Boosting Majority Vote and Feature Concatenation

Most of the experiments previously described are concerned with the majority vote algorithm having SVMs along each view as component classifier (expert). An alternative is to have AdaBoost as expert along each representation. It may well be the case where the ensemble mechanism can boost the performance of majority vote. In this experiment, we use the facial, texture and genomic data to examine majority vote with AdaBoost as expert (AdaBoost-MV). As a reference, we also run AdaBoost in a concatenated feature space, where the features from each view form a single,

Table 4: Robustness of BSSD vs Stacking

Data Set	*Noise* 2 Views	Stacking A_{fus}	BSSD A_{fus}	*Stat.* Signif
Face	10%	0.70	0.75	yes
	20%	0.70	0.75	yes
	30%	0.68	0.74	yes
Gender	10%	0.52	0.75	yes
	20%	0.53	0.69	yes
	30%	0.53	0.66	yes
Glass	10%	0.60	0.73	yes
	20%	0.60	0.70	yes
	30%	0.57	0.67	yes
Data Set	*Noise* 3 Views	Stacking A_{fus}	BSSD A_{fus}	*Stat.* Signif
Face	10%	0.71	0.74	yes
	20%	0.70	0.73	no
	30%	0.69	0.70	no
Gender	10%	0.51	0.71	yes
	20%	0.53	0.61	yes
	30%	0.52	0.54	no
Glass	10%	0.59	0.73	yes
	20%	0.60	0.67	yes
	30%	0.57	0.61	no

Table 5: Robustness of BSSD vs SDP.

Data Set	*Noise* 2 Views	SDP A_{fus}	BSSD A_{fus}	*Stat.* Sig
Face	10%	0.70	0.75	yes
	20%	0.70	0.75	yes
	30%	0.70	0.74	yes
Gender	10%	0.46	0.75	yes
	20%	0.49	0.69	yes
	30%	0.46	0.66	yes
Glass	10%	0.56	0.73	yes
	20%	0.52	0.70	yes
	30%	0.52	0.67	yes
Data Set	*Noise* 3 Views	SDP A_{fus}	BSSD A_{fus}	*Stat.* Sig
Face	10%	0.70	0.74	yes
	20%	0.70	0.73	yes
	30%	0.70	0.70	no
Gender	10%	0.46	0.71	yes
	20%	0.49	0.61	yes
	30%	0.46	0.54	yes
Glass	10%	0.56	0.73	yes
	20%	0.52	0.67	yes
	30%	0.52	0.61	yes

concatenated feature space (AdaBoost-Concatenated). All the three methods, AdaBoost-MV, BSSD, and AdaBoost-Concatenated, use Naive Bayes [20] as base classifiers for boosting.

While AdaBoost-Concatenated performs the worst, due mainly to the curse of dimensionality, BSSD stays on top, especially in noisy environments. It is clear from the performance of both BISD and AdaBoost-MV that the shared sampling mechanism employed by BSSD once again demonstrates distinct advantage in dealing with noisy data.

6 Discussions

Our experiments show that a simple method like BSSD consistently outperforms mathematically sophisticated ones such as SDP. In the case of noisy views, noise is encoded in the base kernel matrices, which in turn severely degrades its performance. Robustness against noise however is a feature that is essential for any data fusion system due to the very nature of different scenarios where such systems are employed. For instance, in a multimodal biometric system it may not be always possible to get the perfect fingerprint of an individual due to dust being accumulated on the sensor or a scar on the finger.

In the noise free case, we try to explore the convergence rate to explain the apparent performance difference. The SDP generalization error bound is given by [12]

$$\frac{1}{n}\sum_{i=1}^{n}\max\{1-y_i f(x_i),0\}+\frac{1}{\sqrt{n}}(4+\sqrt{2\log(\frac{1}{\delta})}+\sqrt{\frac{\mathcal{C}(\mathcal{K})}{n\gamma^2}}), \quad (2)$$

where $\mathcal{C}(\mathcal{K}) = E_\sigma \max_{K\in\mathcal{K}} \sigma^T K\sigma$ with $\sigma^T \in \{\pm 1\}^{2n}$ chosen uniformly randomly, and γ is the margin parameter. Note that the first term is the empirical error, while the second term represents the complexity. Asymptotically (i.e., as n goes to infinity), we are interested in the complexity terms in both (2) and (2). From (2), we have $\tilde{O}(\sqrt{\frac{d}{n\theta^2}})$. For SDP, we have $\tilde{O}(\frac{\sqrt{\mathcal{C}(\mathcal{K})}}{n\gamma})$. Here $\tilde{O}$ is used to hide all logorithmic and constant factors. In [12], it is shown that $\mathcal{C}(\mathcal{K}) \leq 2n^2$. The following lemma shows that it is exactly equal to $2n^2$. Given a fixed set $\{K_1, ..., K_m\}$ of kernel matrices, define $\tilde{K} = \{K = \sum_{j=1}^m \mu_j K_j | K_j \geq 0, \mu_j \in \Re\}$.

Lemma 5 *Let $\mathcal{K} = \tilde{\mathcal{K}} \cup \{yy^T | y \in \{\pm 1\}^{2n}$. We have the following $E_\sigma \max_{K\in\mathcal{K}} \sigma^T K\sigma = 2n^2$.*

Proof. Let $K = yy^T$ and $\sigma = y$. The result follows immediately. ■

Thus the complexity term for SDP becomes $\tilde{O}(\frac{1}{\gamma})$. That is, it does not decrease with n. A similar conclusion is also observed in [24]. On the other hand, $\tilde{O}(\sqrt{\frac{d}{n\theta^2}})$ approaches 0 when n goes to infinity, given everything else being fixed. That is, BSSD has a faster convergence rate than SDP, at least in the way in which the complexity of SDP is measured. This may explain why BSSD has superior performance over SDP on the data sets we have experimented with.

7 Summary

We have presented a novel data fusion technique for multimodal learning. We have provided theoretical analysis for our proposal and shown empirically that our technique significantly outperforms several competing techniques on a number of classification problems and is very robust against noise, which is essential for any data fusion system that must operate across the full range of scenarios the system is expected to encounter. While we have shown experimentally that shared sampling plays a key role in robustness against noise, we have yet to provide a precise mathematical statement. In our future research, we plan on applying advanced concentration inequalities to provide performance bounds for our shared sampling technique with high confidence.

References

[1] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and Systems Science*, vol. 55, pp. 119–139, 1997.

[2] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The feret evaluation methodology for face recognition algorithms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, pp. 1090–1104, 2000.

[3] H. W. Mewes, D. Frishman, C. Gruber, B. Geier, D. Haase, A. Kaps, K. Lemcke, G. Mannhaupt, F. Pfeiffer, C. Schller, S. Stocker, and B. Weil, "Mips: a database for genomes and protein sequences," *Nucleic Acids Research*, vol. 28, no. 1, pp. 37–40, 2000.

[4] S. Hashem, "Optimal linear combination of neural networks," *Neural Networks*, vol. 19, pp. 599–614, 1997.

[5] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas, "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, pp. 226–239, 1998.

[6] L. Breiman, "Bagging predictors," *Machine Learning Journal*, vol. 24, pp. 123–140, 1996.

[7] ——, "Arching classifiers," *Annals of Statistics*, vol. 26, pp. 801–849, 1998.

[8] J. Kittler, "Combining classifiers: A theoretical framework," *Pattern Analysis and Applications*, vol. 1, pp. 18–27, 1998.

[9] ——, "A framework for classifier fusion: Is still needed?" *Lecture Notes in Computer Science*, vol. 1876, pp. 45–56, 2000.

[10] L. I. Kuncheva, "A theoretical study on six classifier fusion strategies," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 2, pp. 281–286, 1997.

[11] D. H. Wolpert, "Stacked generalization," *Neural Networks*, vol. 5, pp. 241–259, 1992.

[12] G. R. G. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, "Learning the kernel matrix with semi-definite programming," *Journal of Machine learning Research*, vol. 5, pp. 27–72, 2004.

[13] P. Viola and M. Jones, "Fast and robust classification using asymmetric adaboost and a detector cascade," *Advances in Neural Information Processing Systems*, vol. 14, 2002.

[14] Y. Freund and R. E. Schapire, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 5, no. 14, pp. 771–780, 1999.

[15] R. E. Schapire, Y. Freund, P. Bartlett, and W. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *Proceedings of the Fourteenth International Conference on Machine learning*, pp. 322–330, 1997.

[16] R. E. Schapire and Y. Singer, "Improved boosting algorithms using confidence rated predictions," *Machine Learning*, vol. 3, no. 37, pp. 297–336, 1999.

[17] R. A. Fisher, "Iris data set, UCI machine learning repository," 1936. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html

[18] G. Rätsch, T. Onoda, and K.-R. Müller, "Soft margins for adaboost," *Machine Learning*, vol. 42, no. 3, pp. 287–320, 2001.

[19] V. Koltchinskii, D. Panchenko, and F. Lozano, "Some new bounds on the generalization error of combined classifiers," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds. MIT Press, 2001, pp. 245–251.

[20] T. Mitchell, *Machine Learning*. McGraw-Hill, 1997.

[21] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 679–714, 1986.

[22] P. P. Vaidyanathan, *Multirate Systems and Filter Banks*. Prentice Hall, 1993.

[23] J. Friedman, "On bias, variance, 0/1-loss, and the curse-of-dimensionality," *http://www-stat.stanford.edu/ jhf*, 1996.

[24] O. Bousquet and D. Herrmann, "On the complexity of learning the kernel matrix," *Advances in Neural Information Processing Systems*, vol. 15, pp. 415–422, 2003.