King Saud University
Journal of King Saud University – Science
www.ksu.edu.sa
www.sciencedirect.com

ORIGINAL ARTICLE

Double committee adaboost

Loris Nanni [a,*], **Sheryl Brahnam** [b], **Alessandra Lumini** [c]

[a] *DIE, University of Padua, Via Gradenigo 6, 35131 Padova, Italy*
[b] *Missouri State University, 901 S. National, Springfield, MO 65804, USA*
[c] *DEIS, Università di Bologna, Via Venezia 52, 47521 Cesena, Italy*

Received 16 January 2012; accepted 20 February 2012
Available online 3 March 2012

KEYWORDS
adaboost;
Random subspace;
Editing approaches;
Multiclassifier systems;
Pattern classification

Abstract In this paper we make an extensive study of different combinations of ensemble techniques for improving the performance of adaboost considering the following strategies: reducing the correlation problem among the features, reducing the effect of the outliers in adaboost training, and proposing an efficient way for selecting/weighing the weak learners. First, we show that random subspace works well coupled with several adaboost techniques. Second, we show that an ensemble based on training perturbation using editing methods (to reduce the importance of the outliers) further improves performance. We examine the robustness of the new approach by applying it to a number of benchmark datasets representing a range of different problems. We find that compared with other state-of-the-art classifiers our proposed method performs consistently well across all the tested datasets. One useful finding is that this approach obtains a performance similar to support vector machine (SVM), using the well-known LibSVM implementation, even when both kernel selection and various parameters of SVM are carefully tuned for each dataset. The main drawback of the proposed approach is the computation time, which is high as a result of combining the different ensemble techniques. We have also tested the fusion between our selected committee of adaboost with SVM (again using the widely tested LibSVM tool) where the parameters of SVM are tuned for each dataset. We find that the fusion between SVM and a committee of adaboost (i.e., a heterogeneous ensemble) statistically outperforms the most used SVM tool with parameters tuned for each dataset. The MATLAB code of our best approach is available at bias.csr.unibo.it/nanni/ADA.rar.

* Corresponding author.
E-mail addresses: loris.nanni@unipd.it (L. Nanni), sbrahnam@missouristate.edu (S. Brahnam), alessandra.lumini@unibo.it (A. Lumini).

1018-3647 © 2012 King Saud University. Production and hosting by Elsevier B.V. All rights reserved.

Peer review under responsibility of King Saud University.
doi:10.1016/j.jksus.2012.02.001

Production and hosting by Elsevier

1. Introduction

A generic machine learning system takes raw data from some input source, preprocesses and transforms the input to reduce noise and to enhance correlation in the data, and then extracts relevant features. System parameters are then continuously fine-tuned until it optimally learns from a training set of data to assign predefined labels to unknown samples in a testing set. Most problems, such as face recognition and finger print identification, use well-known benchmark datasets to test and compare the merits of novel systems. Until recently, these databases contained relatively small and simple sets of data: photographs of faces and images of finger prints, for instance. Today, however, data are far more complex, mainly because data collection and storage have become increasingly cheaper. Scientists are interested in exploring complex relationships between multiple sources of information that define more problems. In medicine, for example, ultrasound images, patient demographic information, and readings from a variety of laboratory tests may all contain vital information regarding particular diseases and outcomes. Increasingly, practitioners are urging researchers to develop computational tools that are capable of handling the current data-rich environments.

To handle complex aggregates of information there is a growing need to develop general-purpose classification methods. Unfortunately, the majority of research in machine intelligence has concentrated on developing methods that work optimally only on well-defined and very specific problems. To handle the complexity of modern data, researchers need to refocus the research agenda to include systems designed to handle a broad spectrum of problems and data types. Ideally, these general-purpose systems would require little parameter tuning and would compete well with less flexible, state-of-the-art methods that have been designed for specific problems. Some recent research along this line includes the work of (Bologna and Appel, 2002; Liu and Huang, 2008; Nanni and Lumini, 2008b, 2006; Güvenir et al., 1998).

A promising technique for handling complex datasets is to build multiclassifier systems, or classifier ensembles (Kuncheva and Whitaker, 2003). The basic idea behind this technique is to average the hypotheses of a diverse group of classifiers in order to produce a better approximation to a true hypothesis (Kittler, 1998). A basic method for building an ensemble is (1) to generate K new training sets starting from the original training set; (2) to train a different classifier for each of the K new training sets; and (3) to combine the K classifiers using a decision rule. As discussed more completely in Section 2, many methods are available for aggregating the decisions of the classifiers.

In this paper our aim is to investigate several ensemble approaches for improving adaboost. A well-known problem with adaboost stems from the fact that it is a sequential forward search procedure that uses the greedy selection strategy; thus, redundancy of the weak learners cannot be avoided. To handle this problem, Dezhen and Kai (2008) have proposed a post optimization procedure that removes redundant classifiers using a GA.

Another problem with adaboost is that it overfits very noisy data (Ratsch et al., 2001; Servedio, 2003). At each iteration in the training process, adaboost tends to focus on classifying the misclassified patterns, too often fitting the noise during training. To avoid overfitting, a technique known as BrownBoost (Freund, 1999) was developed. It gives smaller weights to misclassified patterns that are far from the margin. A "soft margin" method that does not give preference to hypotheses extracted from few patterns with large weights due to continuous misclassification is proposed by (Ratsch et al., 2001). Another method is to weigh each pattern by considering the initial probability, as in MadaBoost proposed by (Domingo and Watanabe, 2000). Two other methods, based on the upper bound so that no one pattern can have too much weight, are SmoothBoost (Servedio, 2003) and NadaBoost (Nakamura et al., 2002). In (Bylander and Tate, 2006) a different approach is used where half the training set is removed to form the validation set. adaboost is applied to the validation set creating a modified set of weights. The training and validation sets are then switched, and a second pass is performed. The final classifier votes using both sets of weights. The basic idea here is to reduce overfitting using a validation set extracted from the training set. Another interesting approach is boosting at start (BAS) (Milidiú and Duarte, 2009), it is an adaboost generalization method that selects any initial weight distribution for the patterns. This process is repeated over N iterations, and a subset of the best BAS members are selected to form a committee for improving the performance. Yet another approach is to use editing techniques to cut the outliers from the training set. In (Nanni and Franco, 2011), for example, an ensemble of adaboost is proposed where each adaboost classifier is trained using a different training set extracted by editing techniques.

In this paper we propose a double committee (i.e., an ensemble of ensembles) for improving the performance of adaboost-based approaches. Our first set of experiments compares several adaboost approaches and their combinations using random subspace. We show that random subspace ensembles are very useful when coupled with several adaboost methods, as well as with RotBoost. Moreover, coupling adaboost with an editing approach ensemble, which reduces the importance of outliers, results in further performance improvement.

The remainder of this paper is outlined as follows. In Section 2 we present our proposed ensemble method. In Section 3 we apply our ensemble method to a diverse set of benchmark datasets to examine its flexibility and accuracy. Finally, in Section 4, we summarize our results and make suggestions for further research.

2. Background on classifier ensembles

There are many methods for creating ensembles of classifiers. One of the most common methods is to use some form of pattern perturbation. In pattern perturbation, new training sets are created by changing the patterns in the original training set, usually via an iterative process. Some common methods for accomplishing this goal include bagging (Breiman, 1996), arcing (Bologna and Appel, 2002), class switching (Martínez-Muñoz and Suárez, 2005) and decorate (Melville and Mooney, 2005). In bagging (Breiman, 1996), the new training sets, S1,..., SK, are subsets of the original training set. In arcing (Bologna and Appel, 2002), each new training set is calculated based on the misclassified patterns in a previous iteration. In class switching (Martínez-Muñoz and Suárez, 2005), K training sets are created by randomly changing the labels of a subset of the training set. In decorate (Melville and Mooney, 2005), the training sets are obtained by adding patterns that the combined decision of the ensemble misclassifies. In boosting/adaboost (Freund and Schapire, 1997), each training pattern is given a weight that increases for patterns that are more difficult to classify.

Feature perturbation is another method that generates new training sets. Some of the more common feature perturbation techniques include random subspace (Ho, 1998), and input decimated ensemble (Tumer and Oza, 2003). In random subspace (Ho, 1998), K new training sets are generated from subsets of the feature set. In input decimated ensemble (Tumer and Oza, 2003), the new training set S_i is generated via the principal component analysis (PCA) transform. PCA is calculated on the training patterns that belong to class i. However, a disadvantage in input decimated ensemble is that the size of the ensemble is bounded by the number of classes. This limitation can be avoided, as in (Nanni and Lumini, 2008a,b), where PCA is performed on training patterns that have been partitioned into clusters. Additional feature perturbation methods include (Ranawana and Palade, 2005), where neural networks are trained on different encoding models for identifying *Escherichia coli* promoter sequences in strings of DNA. Similarly, in (Guo and Lin, 2006) training sets are built using various combinations of features specific to the problem.

Classifier perturbation is yet another way to build ensembles. In this case, ensembles are composed either by using different types of classifiers or by using the same type but with different parameter settings. In both cases, the ensembles are trained on the same training set and the decisions are combined. Some examples include (Lan et al., 2007), where the decisions of five different classifiers (logistic regression, linear discriminant analysis, quadratic discriminant analysis, naive bayes, and K-nearest neighbors) were combined using a weighted-vote decision rule to predict which genes responded to stress, and (Nanni and Lumini, 2007), where three radically different classifiers (a linear support vector machine, a nonlinear radial-basis support vector machine, and a Karhunen-Loeve subspace) were combined to solve a variety of problems.

Finally, ensembles can be composed using a combination of the above methods. Some examples of hybrid methods include random forest (Breiman, 2001), rotation forest (Rodriguez et al., 2006), and RotBoost (Zhang and Zhang, 2008). Random forest (Breiman, 2001) uses a bagging ensemble of decision trees, where a random selection of features are used to split a given node. Rotation forest (Rodriguez et al., 2006) is an ensemble of decision trees, where K new training sets are generated using PCA projections on subsets of the training patterns. Independent component analysis (ICA) has been used as a feature transform for building a rotation forest ensemble in (Nanni and Lumini, 2008a,b; Liu and Huang, 2008). RotBoost (Zhang and Zhang, 2008) ensembles are constructed from decision trees that combine rotation forest and adaboost. RotBoost has been shown to outperform bagging, MultiBoost, rotation forest, and adaboost (Zhang and Zhang, 2008). RotBoost is also one of the first methods that outperformed stand-alone ensemble methods.

As mentioned in the introduction there are several methods for aggregating results: majority voting, sum rule, max rule, min rule, product rule, median rule, and borda count, to name some of the most common. In (Kittler, 1998), the sum rule, or averaging, was shown to outperform most decision rules.

3. Proposed ensemble system

After extensive investigation, we found that the best general-purpose classifier system tested in our experiments is a multi-classifier system that combines the random subspace approach with an editing approach ensemble that reduces the importance of the outliers. Because our intention is to develop a general-purpose classifier, all the parameters in our proposed system had to remain the same, regardless of the dataset. In other words, no ad hoc dataset tuning was allowed. Below we provide a short description, along with an algorithmic outline, of our best systems.

3.1. Random subspace (RS)

RS (Ho, 1998) reduces dimensionality by randomly sampling subsets of features. In our experiments, we use 50% of all features. RS modifies the training data set by generating K ($K = 50$ in our experiments) new training sets and generates classifiers using these modified training sets. The results are combined using the sum rule.

Outline of random subspace: The random subspace ensemble method entails three steps as outlined in (Ho, 1998):

1. Given a d-dimensional data set $D = \{(x_j, t_j)|1 \leqslant j \leqslant m\}$, $x_j \epsilon X R^d$ $t_j C = \{1,...,c\}$ is the label class of x_j, n new projected k-dimensional data sets $D_i = \{(P_i(x_j), t_j)|1 \leqslant j \leqslant m\}$ are generated ($1 \leqslant i \leqslant n$), where P_i is a random projection.
2. Each new data set D_i is given in input to a fixed learning algorithm L which outputs the classifiers h_i.
3. The final classifier h is obtained by aggregating the base classifiers through a given decision rule.

3.2. Reduced reward-punishment editing (RRP)

In the reward-punishment editing technique (Nanni & Franco, 2011) both global and local criterion are used to obtain a more reliable result. In the reduced version (RRP), only local criterion is used for selecting the patterns that are to be removed from the training set. This is accomplished by assigning two weights to each pattern, $\mathbf{x}_i$, as follows:

1. $\boldsymbol{WR}(i)$: denotes the number of times pattern $\mathbf{x}_i$ belongs to a "winner hypersphere". That is it counts the number of times when it contributes to the *correct* classification of another pattern.
2. $\boldsymbol{WP}(i)$: denotes the number of times pattern $\mathbf{x}_i$ belongs to a "loser hypersphere". That is it counts the number of times when it contributes to the *wrong* classification of another pattern.

$\boldsymbol{WR}(i)$ and $\boldsymbol{WP}(i)$ are both linearly normalized between 0 and 1. The final weight of $\boldsymbol{WF}(i)$ is calculated as follows: $\boldsymbol{WF}(i) = \alpha \times \boldsymbol{WR}(i) + (1-\alpha)((1-\boldsymbol{WP}(i))$. Only the θ percentage (see below) of the patterns with highest weight is retained.

Pseudo-code for the reduced RP-Editing algorithm is given in Fig. 1. This function has the following input parameters:

- Training set TS.
- Class labels CL of the training patterns.
- Values of the parameters of the RP-Editing algorithm: k, α, and θ.

In the pseudo-code of the reduced RP-Editing algorithm the following procedures are used:

- K-NN($\mathbf{x}$, S, k): classifies the pattern $\mathbf{x}$ using the k-NN classifier built using set S;
- Normalize: linearly normalizes the values of $\boldsymbol{WR}$ and $\boldsymbol{WP}$ between 0 and 1.
- RankAndEdit (TS, WF, θ): sorts the patterns in the training set TS in decreasing order of score (WF). It retains only the first θ percentage patterns in TS.

```
RP-EDITING($TS$, $CL$, $k$, $\alpha$, $\theta$)
begin
    $WR = WP = WPR = 0$
    for each $\mathbf{x}_i \in TS$
        // k-NN classification of the pattern $\mathbf{x}_i$
        $[L, c]$ = K-NN($\mathbf{x}_i$, $TS$, $k$)
        // Is the pattern $\mathbf{x}_i$ correctly classified?
        if $CL(i) = c$ then
            // Reward of the patterns that contributed to the correct classification of $\mathbf{x}_i$
            for $j = 1$ to $k$
                if $CL(L(j)) = c$ then
                    $WR(L(j)) = WR(L(j)) + 1$
                end if
            end for
        else
            // Punishment of the patterns that contributed to the wrong classification of $\mathbf{x}_i$
            for $j = 1$ to $k$
                if $CL(L(j)) = c$ then
                    $WP(L(j)) = WP(L(j)) + 1$
                end if
            end for
        end if
    end for
    NORMALIZE($WP$, $WR$)
    // Computation of the final weight and Editing
    for each $\mathbf{x}_i \in TS$
    $WF(i) = \alpha \times WR(i) + (1 - \alpha) \times (1 - WP(i))$
    RANKANDEDIT($TS$, $WF$, $\theta$)
end
```

Figure 1 Pseudo-code of the reduced RP-Editing algorithm (from (Nanni & Franco, 2011)).

In building the multiclassifier system, all the training subsets are obtained using all the combinations of α0, 0.25, 0.5, 0.75, 1, θ10%, 22.5%, 35%, 47.5% and k1, 3, 5, 7, 9. Also, if a given training set has less than two patterns for each class, it is discarded from the ensemble.

4. Experimental results

For comparing our general-purpose system with other state-of-the-art methods, we report results obtained on the following benchmark datasets, most of which are available in the UCI Repository.[1] First, we test the following UCI datasets (a detailed description of these databases is available on the UCI machine learning website at http://archive.ics.uci.edu/ml/):

1. The breast cancer dataset (BREAST)
2. The heart disease dataset (HEART)
3. The Pima Indians dataset (PIMA)

[1] http://archive.ics.uci.edu/ml/

4. The Wisconsin breast dataset (WDBC)
5. The Ionosphere dataset (IONO)
6. The vehicle silhouettes dataset (VEI)
7. The vowel dataset (VOW)
8. The German credit (CreditG)
9. The wine dataset (WINE)
10. The sonar dataset (SONAR)

We also test our approach using the HIV dataset[2] (HIV) and five medical image classification problems[3]: 1) 2D HeLa dataset (HE) (Boland and Murphy, 2001); 2) locate endogenous (LE) (Fink et al., 2006); 3) locate transfected (LT) (Fink et al., 2006); 4) CHO dataset (CH) (Shamir et al., 2008); and 5) RNAi (RN) (Shamir et al., 2008).

Table 1 summarizes the main characteristics of the datasets in terms of the number of attributes (A), patterns (E), and classes (C).

We use a standard evaluation protocol in our experiments. As is the custom in many classification experiments, the features are linearly normalized between 0 and 1. Results for each dataset are averaged over ten experiments. We randomly resample the learning and the testing sets (containing respectively half of the patterns) while maintaining the distribution of the patterns in the classes for each experiment. This resampling is done ten times as well. The results are reported as the area under the ROC curve (AUC). AUC is a scalar performance indicator that can be interpreted as the probability that the classifier will assign a higher score to a randomly picked positive pattern rather than to a randomly picked negative pattern. For the multiclass datasets we use the one versus all approach for calculating the area under the ROC curve.

In the first set of experiments, we compare several approaches for building adaboost classifiers. Each cell of the table contains two values. The first is the performance obtained using the standard approach, and the second is the performance obtained using a random subspace (RS) of 50 adaboost classifiers (i.e., 50×50 weak classifiers, if each adaboost combines 50 weak classifiers).

In Table 2 we report the performance obtained by the following systems (for each adaboost method 50 weak classifiers are combined):

- RotB (RotationBoosting): the method proposed in (Zhang and Zhang, 2008).
- Real (RealAdaboost): as implemented in GML adaboost MATLAB Toolbox, using the decision tree as classifier (Schapire and Singer, 1999).
- Gentle (GentleAdaboost): as implemented in GML adaboost MATLAB Toolbox, using the decision tree as classifier (Friedman et al., 2000).

[2] Dataset used in T. Rögnvaldsson and L. You. "Why neural networks should not be used for HIV-1 protease cleavage site prediction". Bioinformatics, 20, pp. 1702–1709 (2004) after the orthonormal encoding the data are projected by principal component analysis in a 50-dimensional space.

[3] Each image is described by rotation invariant uniform bins extracted by local ternary patterns (Tan and Triggs, 2010), let us define P as the number of pixels in the neighborhood, R as the radius and τ the threshold used for extracting the ternary coding. The feature vector that describe an image is obtained concatenating the descriptors obtained with ($P = 8$, $R = 1$) and ($P = 16$, $R = 2$), both with threshold $\tau = 2$.

Table 1 Characteristics of the datasets used in the experimentation: A is the number of attributes, E is the number of patterns, and C is the number of classes.

Dataset	A	E	C
BREAST	9	699	2
HEART	13	303	2
WDBC	30	569	2
PIMA	8	768	2
VEI	18	946	4
IONO	34	351	2
VOW	10	528	11
CreditG	20	1000	2
WINE	13	178	2
HIV	50	362	2
SONAR	60	208	2
He	56	862	10
LE	56	502	10
LT	56	553	10
CH	56	327	5
RN	56	200	10

- Modest (ModestAdaboost): as implemented in GML adaboost MATLAB Toolbox, using the decision tree as classifier (Vezhnevets and Vezhnevets, 2005).
- A (adaboost.M2): using a neural network as classifier.
- RA-s: the ensemble of modified RealAdaboost proposed in (Gómez-Verdejo et al., 2010), in this approach a neural network is used as the classifier.

The column AV reported in Table 2 reports the average performance of a given method in the set of tested datasets. The column DIFF is the improvement of the AUC between the stand-alone version classifier and its random subspace version.

Analyzing the results reported in Table 2, we can draw the following conclusions:

- The best performing ensemble method in the UCI datasets is an RS of *A*.
- None of the tested classifiers generalizes better than any of the others, i.e., none outperforms any of the others across all the datasets (no free lunch theorem).
- RS ensembles prove quite useful except in RA-s.
- RS-RotB obtains the highest AUC in the image datasets; in these datasets, the most used indicator in the literature is accuracy (notice that these are multiclass problems). The average accuracy of RS-RotB in these datasets is 90.24%, while the average accuracy of RS-A is 92.29%.

We ran the Wilcoxon signed-rank test (Demsar, 2006) to compare the results of different methods, as this method was shown in (Demsar, 2006) to be the best approach for comparing classifiers. The null hypothesis is that there is no difference between the accuracies of couples of classifiers. We reject the null hypothesis (level of significance 0.10) and accept that both a random subspace of *A* and a random subspace of RS-RotB are the best approaches.

For the next tests, *A* is selected as the classifier because of its good performance in all the datasets (considering as well its accuracy in the image datasets) and because is less computational power with respect to RotB (see Table 7). In Table 3, we report the performance obtained by the following systems:

Table 2 Experimental results of methods on different datasets.

	HEART	SONAR	PIMA	IONO	BREAST	VEI	VOW	WDBC	Credit
RotB	0.9140	0.9156	0.8094	0.9812	0.9911	0.9394	0.9947	0.9968	0.7875
	0.9206	0.9331	0.8170	**0.9847**	**0.9919**	**0.9396**	**0.9951**	0.9961	0.7982
Real	0.8734	0.8987	0.7701	0.9747	0.9888	0.9020	0.9877	0.9938	0.7315
	0.8910	0.9156	0.8010	0.9804	0.9908	0.9271	0.9909	0.9950	0.7330
Gentle	0.8827	0.9078	0.7707	0.9708	0.9857	0.9165	0.9891	0.9947	0.7449
	0.8957	0.9250	0.7920	0.9776	0.9898	0.9246	0.9907	0.9955	0.7598
Modest	0.8741	0.8979	0.7919	0.9687	0.9877	0.7415	0.4358	0.9936	0.6895
	0.8971	0.9169	0.8018	0.9758	0.9906	0.8172	0.4447	0.9959	0.7017
A	0.8893	0.9066	0.7848	0.9601	0.9865	0.9303	0.9890	0.9935	0.7167
	0.9206	**0.9351**	**0.8189**	0.9777	0.9907	0.9369	0.9827	0.9959	0.7832
RA-s	0.9120	0.8857	0.8159	0.9476	0.9908	0.9220	0.9254	**0.9989**	0.8038
	0.9189	0.8794	0.8101	0.9674	0.9914	0.9040	0.9149	0.9963	**0.8121**

	WINE	HIV	HE	LE	LT	CH	RN	AV	DIFF (%)
RotB	0.9970	0.9515	0.9825	0.9953	0.9930	0.9989	0.9540	0.9501	0.4
	0.9992	0.9573	**0.9851**	**0.9957**	**0.9940**	0.9993	**0.9707**	**0.9548**	
Real	0.9825	0.9363	0.9725	0.7868	0.9840	0.9864	0.6215	0.8994	0.76
	0.9950	0.9348	0.9750	0.7879	0.9335	0.9899	0.6671	0.9067	
Gentle	0.9885	0.9331	0.9650	0.9826	0.9833	0.9736	0.8366	0.9266	0.98
	0.9963	0.9344	0.9780	0.9862	0.9873	0.9845	0.8590	0.9360	
Modest	0.9877	0.9337	0.9450	0.6175	0.5442	0.8659	0.3018	0.7860	3.2
	0.9955	0.9426	0.9725	0.6818	0.6038	0.9677	0.3643	0.8169	
A	0.9810	0.9543	0.9815	0.9921	0.9920	0.9992	0.9245	0.9363	1.47
	0.9992	0.9568	0.9840	0.9928	0.9923	**0.9995**	0.9379	0.9503	
RA-s	**0.9992**	**0.9665**	0.9801	0.9912	0.9875	0.9992	0.9125	0.9399	−0.04
	0.9992	0.9651	0.9800	0.9910	0.9888	0.9992	0.9135	0.9395	

The bold values are the highest performance in each dataset (i.e. each column of the tables).

Table 3 Experimental results of ensemble of random subspace of adaboost.

	HEART	SONAR	PIMA	IONO	BREAST	VEI	VOW	WDBC	Credit
Rs-A	0.9206	**0.9351**	**0.8189**	0.9777	**0.9907**	**0.9368**	**0.9825**	0.9959	0.7832
Out-A	0.9155	0.9134	0.8160	0.9708	0.9898	0.9350	0.9805	0.9957	0.7992
ED-A	**0.9220**	0.9244	0.8160	**0.9801**	0.9901	0.9330	0.9781	**0.9971**	**0.8014**

	WINE	HIV	HE	LE	LT	CH	RN
Rs-A	0.9992	0.9568	**0.9840**	**0.9928**	0.9923	**0.9995**	**0.9379**
Out-A	0.9987	**0.9588**	0.9789	0.9918	0.9918	0.9991	0.9355
ED-A	**0.9993**	0.9511	0.9769	0.9918	**0.9925**	0.9984	0.9344

The bold values are the highest performance in each dataset (i.e. each column of the tables).

- Rs-X: a RS of 50 X.
- OUT-X: the RRP ensemble is combined with an RS of 50 X; in this method the training patterns cut by RRP are used for training adaboost, but their weights are not changed inside the adaboost algorithm.
- ED-X: the RRP ensemble is combined with an RS of 50 X; this is the standard approach based on RRP. The training patterns cut by RRP are NOT used for training adaboost.

The performance of all the methods reported in Table 3 is very similar (no statistical differences using the Wilcoxon rank test). From Table 3 we see that the best choice is to couple RS with A, because Rs-A compared with the other methods has the lowest computational demands. Notice that ED-A is an interesting method since it is based on an editing approach ensemble that reduces the importance of the outliers.

The next experiment, reported in Table 4, compares some methods for selecting and weighting the weak learners. The input consists of the scores obtained by each weak learner weighed by the weight it obtained in that dataset. In this experiment, we test the following methods (coupled with Rs-A):

- Sparse: the classifier selection method proposed in (Zhang and Zhoum, 2011), we use the well-known Platt's method (Platt, 1999) for obtaining the probabilities from the weak learners scores.
- Sel_AUC: a genetic algorithm for weighing each weak learner between 0 and 1. The fitness function is the AUC obtained by the ensemble in the training set.
- Sel_SFFS: sequential forward floating is used for selecting the weak learner classifiers. The fitness function is 1/AU + (1−W')/W, where AU is AUC obtained by the

Table 4 Experimental results of weak learner selection/weighting.

	HEART	SONAR	PIMA	IONO	BREAST	WDBC	VEI	VOW	Credit
Sparse	0.9210	0.9088	**0.8201**	**0.9823**	0.9910	0.9966	0.9350	0.9815	**0.8088**
Sel_AUC	0.9190	0.8940	0.8165	0.9795	**0.9916**	0.9958	0.9315	0.9800	0.8080
Sel_SFFS	0.9120	0.8870	0.8125	0.9762	0.9910	0.9950	0.9330	0.9805	0.8055
Rs-A	0.9206	**0.9351**	0.8189	0.9777	0.9907	0.9959	**0.9368**	**0.9825**	0.7832

	WINE	HIV	HE	LE	LT	CH	RN
Sparse	**0.9996**	**0.9584**	**0.9800**	**0.9940**	0.9935	0.9989	**0.9380**
Sel_AUC	**0.9996**	0.9548	0.9825	0.9915	0.9950	0.9990	0.9350
Sel_SFFS	**0.9996**	0.9551	0.9836	0.9925	0.9925	0.9989	0.9365
Rs-A	0.9992	0.9568	0.9840	0.9928	0.9923	**0.9995**	0.9379

The bold values are the highest performance in each dataset (i.e. each column of the tables).

Table 5 Experimental results of KNORA,ORACLE and fusions with SVM.

	HEART	SONAR	PIMA	IONO	BREAST	VEI	VOW	WDBC	Credit
RS-A	0.9206	0.9351	0.8189	0.9777	0.9907	0.9368	0.9825	0.9959	0.7832
ORACLE	0.9199	0.9345	0.8157	0.9810	0.9904	0.9355	0.9805	0.9963	0.7973
KNORA	0.9214	0.9246	0.8144	0.9796	0.9900	0.9365	0.9830	0.9971	0.8009
OpSVM	0.9146	**0.9595**	0.8224	0.9799	0.9925	0.9460	0.9929	0.9971	**0.8134**
E + R	**0.9215**	0.9344	0.8152	0.9813	0.9905	0.9361	0.9820	0.9967	0.7834
S + R	0.9195	0.9499	0.8265	**0.9836**	0.9923	0.9478	0.9934	0.9971	0.8021
S + E + R	0.9204	0.9490	0.8240	0.9827	0.9921	0.9472	0.9929	**0.9972**	0.7998
2 × S + R + E	0.9169	0.9548	**0.8276**	0.9830	**0.9926**	**0.9486**	**0.9937**	0.9971	0.8031

	WINE	HIV	HE	LE	LT	CH	RN	RANK	AV
RS-A	**0.9996**	0.9584	0.9840	0.9928	0.9923	0.9995	0.9379	5.6250	0.9503
ORACLE	0.9991	0.9523	0.9815	0.9932	0.9930	0.9988	0.9375	6.2500	0.9504
KNORA	0.9993	0.9516	0.9810	0.9925	0.9945	0.9990	0.9385	5.7500	0.9502
OpSVM	0.9984	**0.9647**	0.9862	0.9892	0.9930	0.9994	0.9393	4.3750	0.9555
E + R	0.9991	0.9567	0.9843	**0.9940**	0.9950	0.9993	0.9390	5.5000	0.9505
S + R	0.9994	0.9588	0.9869	0.9931	0.9958	0.9994	0.9447	3.3750	0.9556
S + E + R	0.9995	0.9607	**0.9874**	0.9936	0.9959	**0.9996**	0.9510	2.8125	0.9558
2 × S + R + E	0.9993	0.9612	0.9873	0.9939	**0.9961**	**0.9996**	**0.9522**	**2.3125**	**0.9567**

The bold values are the highest performance in each dataset (i.e. each column of the tables).

Table 6 Comparison when artificial outliers are created.

Dataset	LIN	RBF	POL	BEST
HEART	0.8948	0.8803	0.8745	**0.9146**
SONAR	0.8195	0.9452	0.8931	**0.9595**
PIMA	0.8182	0.8221	0.8220	**0.8224**
IONO	0.8557	0.9711	0.8751	**0.9799**
BREAST	**0.9925**	0.9877	0.9908	**0.9925**
VEI	0.9244	0.9110	0.9389	**0.9460**
VOWEL	0.8181	0.9717	0.9645	**0.9929**
WDBC	0.9920	0.9962	0.9901	**0.9971**
CreditG	0.8008	0.7557	0.6778	**0.8134**
WINE	0.9948	0.9982	0.9946	**0.9984**
HIV	0.9487	0.9395	0.9542	**0.9647**
HE	0.9567	**0.9862**	0.9843	**0.9862**
LE	0.9819	0.9869	0.9864	**0.9892**
LT	0.9778	0.9922	**0.9930**	**0.9930**
CHO	0.9928	**0.9994**	0.9974	**0.9994**
RNA	0.8897	0.9336	0.9351	**0.9393**

The bold values are the highest performance in each dataset (i.e. each column of the tables).

ensemble in the training set, W' is the number of selected weak classifiers, and W is the total number of weak classifiers.

The weak learner selection is performed independently in each training set obtained by RRP.

Analyzing the results reported in Table 4, we can draw the conclusion that different approaches obtain a very similar performance (there is no statistical difference considering the Wilcoxon rank test).

Finally, we attempt to improve the performance of our approach Rs-A by coupling it with the following:

- KNORA, the classifier selection approach proposed in (Ko et al., 2008).
- Principal direction linear oracle (PDLO), this ensemble classifier (Peterson and Coleman, 2007) is used to invoke a linear hyperplane split of training patterns. It is a variant of random oracle. The data of each of the two subsets (obtained by splitting the training set using the hyperplane) are used to train two different classifiers. For each test pattern the hyperplane is used to choose which of the two classifiers is chosen to classify the given pattern.

Table 7 Comparison computation time.

Method	PIMA		HE	
	Training time	Test time	Training time	Test time
RotB	38.36	0.97	355.56	12.50
Real	3.15	0.09	4.94	0.14
Gentle	1.56	0.09	4.93	0.14
Modest	1.55	0.09	4.71	0.15
A	4.48	0.14	26.51	3.15
RA-s	6.52	0.12	9.25	0.45
ED-A	1720	15.25	15200.50	225.25
RS-A	45.50	1.56	930.25	21.50

Table 8 Comparison using accuracy as performance indicator.

Dataset	RS-A	OpSVM	S + R	2 × S + R + E
HE	91.05	90.70	91.40	**91.63**
LE	96.20	95.80	**96.60**	96.00
LT	95.45	94.55	95.45	**94.73**
CHO	98.77	99.08	**99.38**	99.08
RNA	80.00	79.50	**82.00**	**82.00**
Average	92.29	91.93	**92.97**	92.69

The bold values are the highest performance in each dataset (i.e. each column of the tables).

In Table 5 we also include for comparison purposes the performance of the SVM (OpSVM), where the best kernel and the best set of parameters are chosen separately in each dataset. Moreover, in this table we report some fusions by sum rule/weighted sum rule between OpSVM and our committee of adaboost:

- E + R, fusion by sum rule between ED-A and RS-A.
- S + R, fusion by sum rule between OpSVM and RS-A.
- S + R + E, fusion by sum rule among OpSVM, ED-A and RS-A.
- 2 × S + R + E, fusion by weighted sum rule among OpSVM, ED-A and RS-A. The weight of OpSVM is 2, while the other weights are 1.

It is interesting to note that Rs-A obtains a performance similar to that of SVM (using the well know LibSVM implementation), even when both kernel selection and the various parameters of the SVM are carefully tuned for each dataset. We want to stress that the best method is 2 × S + R + E. Using the Wilcoxon rank test, we reject the null hypothesis (level of significance 0.10) and accept that 2 × S + R + E outperforms the stand-alone SVM. This is the most significant finding of this paper since it shows that our general-purpose systems perform better than a finely-tuned SVM. Before the fusions, the scores of classifiers must be normalized (we normalize to mean 0 and standard deviation 1).

In Table 6 we show how important a careful tuning of the SVM parameters is for each dataset by examining the following:

- LIN: performance obtained by liner SVM using the best average parameters (i.e., the parameters that obtain the highest AUC average on the set of tested databases).
- RBF: performance obtained by radial-basis function SVM using the best average parameters.
- POL: performance obtained by polynomial SVM using the best average parameters.
- BEST: performance obtained by a SVM with the parameters tuned for that dataset.

We have compared BEST with LIN, RBF, POL. In each comparison we reject the null hypothesis with a very low level of significance (0.05); it is clear that for SVM it is essential that parameters be fine-tuned for each dataset[4].

In Table 7 we report the computation time[5] (in seconds) of several adaboost approaches in two datasets PIMA and HeLa. Notice that we report the performance for classifying the entire testing set (154 patterns for PIMA and 172 patterns in HE). The proposed approaches are thus suited for real application even in cases demanding more computational time, given the computation power of most modern PCs.

In Table 8 we compare some approaches in the image datasets using accuracy as the performance indicator. Accuracy is less reliable than AUC, but in the medical literature this indicator is widely used for assessing the performance of systems using the image medical datasets used in this paper. We used local ternary patterns (LTP) as the texture descriptor. LTP the best performing texture descriptors used with in these problems. Moreover, almost all the state-of-the-art approaches use LibSVM as classifier. From the results reported in Table 8, we observe that the fusion by sum rule between OpSVM and RS-A outperforms OpSVM. This is another very interesting finding of this paper, making RS-A a very useful system for practitioners. In our opinion a heterogeneous system based on an ensemble of unstable classifiers (decision trees or neural networks) and strong classifiers (such as SVM) is the most feasible way for trying to overcome as best as possible the "no free lunch" hypothesis that there is no single classifier that works best on all given problems, i.e., that "any two algorithms are equivalent when their performance is averaged across all possible problems" (Wolpert and Macready, 2005).

5. Conclusion

In this paper we attempted to discover new methods for building general-purpose ensembles of classifiers that would require minimum to no parameter tuning and that would perform well across a broad spectrum of classification problems. We performed a number of empirical comparisons of several multi-classifier systems using several benchmark datasets, and our experimental results demonstrate that our new methods outperform other adaboost methods.

Unfortunately, we were not able to discover a single ensemble method that outperformed all others across the tested datasets, (supporting the "no free lunch" metaphor). Nonetheless, a number of significant practical findings are reported. We show that the best approach (tradeoff between performance and complexity) is obtained by combining random subspace

[4] While for adaboost the parameters tuning is not so important since several weak learners are combined together.

[5] Core i5 750, 2.66 Ghz with 8G Ram runing MATLAB 2011a 64 bit.

with an adaboost M2 using a neural network as classifier. In addition, we developed an approach that does not require careful tuning of parameters for each dataset, yet outperforms other high performing methods, such as support vector machines. Since there is less risk of over-training using our new method, it is well-suited for practitioners. Another interesting finding is that random subspace can be coupled with several adaboost methods for improving the performance.

Acknowledgements

The authors would like to thank the several authors that have shared their MATLAB code: Vezhnevets A; Vezhnevets V; Zhang C-X; Zhang J-S; Zhang L; Zhou W-D; Xu-Cheng Y.

References

Boland, M.V., Murphy, R.F., 2001. A neural network classifier capable of recognizing the patterns of all major subcellular structures in fluorescence microscope images of HeLa cells. Bioinformatics (Oxford, England) 17 (12), 1213–1223.

Bologna, G., Appel, R.D., 2002. A comparison study on protein fold recognition. In: The Ninth International Conference on Neural Information Processing, Singapore, pp. 2492–2496.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Breiman, L., 2001. Random forest. Mach. Learn. 45 (1), 5–32.

Bylander, T., Tate, L., 2006. Using validation sets to avoid overfitting in adaboost. Paper presented at the nineteenth international florida artificial intelligence research society (FLAIRS).

Demsar, J., 2006. Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. 7, 1–30.

Dezhen, Z., Kai, Y., 2008. Genetic algorithm based optimization for adaboost. Paper presented at the international conference on computer science and software engineering (CSSE).

Domingo, C., Watanabe, O., 2000. Scaling up a boosting-based learner via adaptive sampling. Paper presented at the Pacific-Asia conference on knowledge discovery and data mining.

Fink, J.L., Aturaliya, R.N., Davis, M.J., Zhang, F., Hanson, K., Teasdale, M.S., et al., 2006. LOCATE: a mouse protein subcellular localization database. Nucleic Acids Res. 34 (Database issue), D213–D217.

Freund, Y., 1999. An adaptive version of the boost by majority algorithm. Paper presented at the COLT. In: Proceedings of the Workshop on Computational Learning Theory.

Freund, Y., Schapire, R.E., 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.

Friedman, J., Hastie, T., Tibshirani, R., 2000. Additive logistic regression: a statistical view of boosting. Ann. Stat. 38 (2), 337–374.

Gómez-Verdejo, V., Arenas-García, J., Figueiras-Vidal, A.R., 2010. Committees of Adaboost ensembles with modified emphasis functions. Neurocomputing 73 (7–9), 1289–1292.

Guo, J., Lin, Y., 2006. Tssub: eukaryotic protein subcellar localization by extracting features from profiles. Bioinformatics 22, 1784–1785.

Güvenir, H.A., Demiröz, G., Iter, N., 1998. Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. Artif. Intell. Med. 13, 147–165.

Ho, T.K., 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20 (8), 832–844.

Kittler, J., 1998. On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20 (3), 226–239.

Ko, A.H.R., Sabourin, R., de Souza Britto Jr., A., 2008. From dynamic classifier selection to dynamic ensemble selection. Pattern Recogn. 41 (5), 1718–1731.

Kuncheva, L.I., Whitaker, C.J., 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. 51 (2), 181–207.

Lan, H., Carson, R., Provart, N., Bonner, A., 2007. Combining classifiers to predict gene function in arabidopsis thaliana using large-scale gene expression measurements. BMC Bioinformatics 8, 358.

Liu, K., Huang, D., 2008. Cancer classification using rotation forest. Comput. Biol. Med. 38 (5), 601–610.

Martínez-Muñoz, G., Suárez, A., 2005. Switching class labels to generate classification ensembles. Pattern Recognit. 38 (10), 1483–1494.

Melville, P., Mooney, R.J., 2005. Creating diversity in ensembles using artificial, information fusion. Special Issue on Diversity in Multiclassifier Systems 6 (1), 99–111.

Milidiú, R.L., Duarte, J.C., 2009. Improving BAS Committee Performance with a Semi-supervised Approach. ESANN.

Nakamura, M., Nomiya, H., Uehara, K., 2002. Improvement of boosting algorithm by modifying the weighting rule. Annals of Mathematics and Artificial Intelligence 41, 95–109.

Nanni, L., Franco, A., 2011. Reduced reward-punishment editing for building ensembles of classifiers. Expert Syst. Appl. 38, 2395–2400.

Nanni, L., Lumini, A., 2007. Ensemblator: an ensemble of classifiers for reliable classification of biological data. Pattern Recognit. Lett. 28 (5), 622–630.

Nanni, L., Lumini, A., 2008a. Ensemble generation and feature selection for the identification of students with learning disabilities. Expert System with Applications.

Nanni, L., Lumini, A., (Eds.), 2008b. Using Ensemble of Classifiers in Bioinformatics. Machine Learning Research Progress, vol. 36. Nova publishers, pp. 3896–3900.

Peterson, L.E., Coleman, M.A., 2007. Principal direction linear oracle for gene expression ensemble classification. Paper presented at the computational intelligence approaches for the analysis of bioinformatics data (CIBIO07).

Platt, J., 1999. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola, A.J., Bartlett, P., Scholkopf, B., Schuurmans, D. (Eds.), Advances in Large Margin Classiers. MIT Press.

Ranawana, R., Palade, V., 2005. A neural network based multiclassifier system for gene identification in DNA sequences. Nerual Computing and Applications 14, 122–131.

Ratsch, G., Onoda, T., Muller, K.R., 2001. Soft margins for adaboost. J. Mach. Learn. 42 (3), 287–320.

Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J., 2006. Rotation forest: a new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. 28 (10), 1619–1630.

Schapire, R.E., Singer, Y., 1999. Improved boosting algorithms using confidence-rated predictions. Mach. Learn. 37 (3), 297–336.

Servedio, R.A., 2003. Smooth boosting and learning with malicious noise. J. Mach. Learn. Res. 4, 633–648.

Shamir, L., Orlov, N., Eckley, D.M., Macura, T., Johnston, J., Goldberg, I., 2008. Wndchrm – an open source utility for biological image analysis. BioMed. Central 3 (13).

Tumer, K., Oza, N.C., 2003. Input decimated ensembles. Pattern Anal. Appl. 6, 65–77.

Vezhnevets, A., Vezhnevets, V., 2005. 'Modest adaboost' – teaching adaboost to generalize better. Paper presented at the Graphicon, Novosibirsk Akademgorodok, Russia.

Wolpert, D.H., Macready, W.G., 2005. Coevolutionary free lunches. IEEE Trans. Evol. Comput. 9 (6), 721–735.

Zhang, C-X., Zhang, J-S., 2008. Rotboost: a technique for combining rotation forest and adaboost. Pattern Recognit. Lett. 29 (10), 1524–1536.

Zhang, L., Zhoum, W.-D., 2011. Sparse ensembles using weighted combination methods based on linear programming. Pattern Recogn. 44 (1), 97–106.