A Combination of Methods for Building Ensembles of Classifiers

L. Nanni,[1] S. Brahnam,[2] and A. Lumini[3]
[1] DIE - University of Padua, Via Gradenigo, 6 - 35131- Padova – Italy.
[2]Missouri State University, 901 S. National, Springfield, MO 65804, USA
[3]DEIS, Università di Bologna, Via Venezia 52, 47521 Cesena, Italy.

Abstract - *In this paper we make an extensive study of different methods for building ensembles of classifiers. We examine variants of ensemble methods that are based on perturbing features. We illustrate the power of using these variants by applying them to a number of different problems. We find that the best performing ensemble is obtained by combining an approach based on random subspace with a cluster-based input decimated ensemble and the principal direction oracle. Compared with other state-of-the-art stand-alone classifiers and ensembles, this method consistently performed well across twelve diverse benchmark datasets. Another useful finding is that this approach does not require parameters to be carefully tuned for each dataset (in contrast to the fundamental importance of parameters tuning when using SVM and extreme learning machines), making our ensemble method well suited for practitioners since there is less risk of over-training. Another interesting finding is that random subspace can be coupled with several other ensemble methods to improve performance.*

Keywords: f input decimated ensembles, random subspace, multiclassifier systems, pattern classification.

1 Introduction

Until recently, science worked with relatively small sets of data since collecting measurements was difficult, time consuming, and expensive. With increasingly cheaper and more powerful forms of computing and data storage, scientists in the 21st century are now producing far more information than can be processed. To truly assist practitioners in other fields, researchers in machine intelligence need to develop general purpose classification methods that are capable of handling a broad variety of problems and data types. These classification methods also need to be easy to use (requiring, for instance, little parameter tuning), and they need to compete well with less flexible state-of-the-art methods that have been crafted for very specific problems.

One of the most promising techniques for improving flexibility and accuracy is to build systems that combine multiple classifiers [1]. The main idea behind a multiclassifier system is to average the hypotheses of a diverse group, or ensemble, of classifiers to produce a better approximation to a true hypothesis [2]. In this paper our aim is to compare several approaches for building ensembles of classifiers to find a method that works well across diverse datasets without careful parameters tuning for each dataset. In our investigation, we find that the best general purpose method combines an approach based on a random subspace with a cluster-based input decimated ensemble and the principal direction oracle. We find that this method compares very well with several state-of-the-art stand-alone and ensemble methods.

The remainder of this paper is organized as follows. In section 2 we describe several methods for constructing ensembles of classifiers. In section 3 we present our best ensemble method. In section 4 we apply our ensemble method to a diverse set of benchmark datasets to examine its flexibility and accuracy. Moreover, results of several state-of-the-art stand-alone and ensembles methods are compared with our approach using the same datasets. Finally, in section 5, we summarize our results and make suggestions for further research.

2 Multiclassifier Systems

A simple classifier takes raw data from an input source, preprocesses and transforms it to reduce noise and to enhance correlation in the data, and then extracts relevant features. Classifier parameters are then continuously fine-tuned as the classifier optimally learns from a training set to assign predefined labels to unknown samples in a testing set. A multiclassifier system, in contrast, predicts class labels from previously unseen records in the testing set by aggregating predictions made by an ensemble of simple classifiers. Some common methods for aggregating the decisions of multiple classifiers include majority voting, sum rule, max rule, min rule, product rule, median rule, and Borda count are some of the most common methods [2].

There are several general approaches for constructing multiclassifiers. One approach is to focus on methods for dividing or perturbing either the patterns or the features in the training set. Another approach is to focus on methods for either combining the results of different classifier types or perturbing the parameters of a set of classifiers of the same type. Combinations of these approaches, or hybrid systems, have also been proposed. The three basic steps involved in

constructing multiclassifier systems using the first approach are 1) generate K new training sets starting from the original training set; 2) train a different classifier for each of the K new training sets; and 3) combine the K classifiers using a decision rule.

When ensembles are composed using pattern perturbation, new training sets are constructed by changing the patterns in the original training set. This is usually done iteratively. Common methods for constructing new training sets include Bagging [3], Class Switching [4], Decorate [5], and Boosting/AdaBoost [6]. In Bagging [3] new training sets, $S_1,\ldots, S_K$, are subsets of the original training set. In Arcing [7] misclassified patterns are used to calculate the patterns included in each new training set. In Class Switching [4] K training sets are randomly created by changing the labels of a subset of the training set, and in Decorate [5] the training sets are constructed by adding patterns that the combined decision of the multiclassifier system misclassifies. In Boosting/ AdaBoost [6] each training pattern is given a weight that increases at each iteration for patterns that are most difficult to classify.

Feature perturbation generates new training sets by changing the feature set. Common methods for perturbing features include Random Subspace [8], Cluster-based Pattern Discrimination [9], and Input Decimated Ensemble [10]. In Random Subspace [8] K new training sets are subsets of the feature set. In Cluster-based Pattern Discrimination[9] K new training sets are built by independently partitioning the classes into clusters and then by defining different features for each cluster. In Input Decimated Ensemble [10] the new training set S_i is obtained using the principal component analysis (PCA) transform which is calculated on the training patterns that belong to class *i*. One drawback using Input Decimated Ensemble is that the size of the ensemble is bounded by the number of classes. This limitation is avoided in [11], where the training patterns are partitioned into clusters, and the PCA transform is then performed on the training patterns within each cluster. In the case of combining different classifiers, either different classifiers are used to build the ensemble or classifiers of the same type are used but with different parameter settings. In either case, the classifiers are trained on the same training set and the decisions are combined. Examples of systems that combine different types of classifiers include [12], where the decisions of five different classifiers (Logistic Regression, Linear Discriminant Analysis, Quadratic Discriminant Analysis, Naive Bayes, and K-Nearest Neighbors) are combined using a weighted-vote decision rule to predict which genes respond to stress.

Hybrid methods combine different perturbation methods. Some examples include Random Forest [13], Rotation Forest [14], and RotBoost [15]. Random Forest [13] uses a bagging ensemble of Decision Trees, where a random selection of features are used to split a given node. In Rotation Forest [14], an ensemble of Decision Trees is used, where K new training sets are constructed by applying several PCA projections onto subsets of the training patterns. Several researchers have also reported the value of using Independent Component Analysis (ICA) as a feature transform for building a rotation forest ensemble (see, for example, [11]). In RotBoost [15] ensembes are constructed from Decision Trees that combine Rotation Forest and Adaboost. RotBoost has been shown to outperform Bagging, MultiBoost, Rotation Forest, and AdaBoost [15]).

3 System architecture

After extensive testing, we found that the best general purpose classifier system is a multiclassifier system that combines the random subspace supervised approach with a cluster based input decimated ensemble and the principal direction oracle. In this section, we provide a more detailed description of the ensemble methods used in our experiments. The classifier used in our ensembles, is the decision tree with pruning, where the information gain is used as the binary splitting criterion [16].

The proposed algorithm can be outlined as follows ;

1. Extract a set of *T* subspace using the Learn.MF algorithm;
2. Perform a linear hyperplane split of training samples by principal direction linear oracle;
3. Train a cluster-based Input Decimated Ensemble;
4. Combine the scores of the classifiers that built the ensemble by sum rule.[1]

3.1 System ensemble methods

Below we provide a short description of the various ensemble methods used in our experiments.

Random Subspace (RS0) [8] reduces dimensionality by randomly sampling subsets of features (50% of all the features in our experiments). RS modifies the training dataset by generating K (K=50 in our experiments) new training sets and by building classifiers using these modified training sets. The results are combined using the sum rule

Learn.MF (LM) [17] is a variant of random subspace. It trains an ensemble of classifiers with a subset of features, randomly drawn from a feature distribution, which is iteratively updated to favor the selection of those features that were previously undersampled. In our experiments, each subspace contains 50% of the original features.

Principal direction linear oracle (PD) [18] is an ensemble classifier used to invoke a linear hyperplane split of training samples. It is a variant of random oracle. The data of each of

[1] We want to stress that all the parameters of the proposed system are the same in all the datasets without any ad hoc dataset tuning.

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 the best classifier for the given pattern.

Input Decimated Ensemble (IDE) is a method for constructing ensembles based on pattern perturbation. In our experiments we use a variant of IDE, proposed in [11], where each classifier is trained using the feature transform Neighborhood Preserving Embedding (NPE) (see [19] for details) on a subset, randomly extracted, of the training patterns, with each subset containing patterns of only one class. The different classifiers are combined by sum rule. The MATLAB code used for NPE is freely available at http://www.cs.uiuc.edu/homes/dengcai2/Data data.html.

4 Experimental results

For comparing our proposed approaches with other state-of-the-art methods, we report results obtained on the following twelve benchmark datasets, most of which are available in the UCI Repository (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 *(BR);* 2) The heart disease dataset *(HE)*; 3) The Pima Indians Dataset *(PI)*; 4) The Wisconsin breast dataset *(WD)*; 5) A erythemato-squamous disease classification dataset (*DE*); 6) The Ionosphere Data Set (*IO*); 7) The vehicle silhouettes dataset (*VE*); 8) The vowel dataset *(VO);* 9) The German Credit *(CG);* 10) The wine dataset *(WI);* 11) The sonar dataset *(SO);* and 12) The HIV dataset2 *(HI).*

The evaluation protocol used in our experiments is fairly standard. As suggested by many classification approaches, the features in these datasets are linearly normalized between 0 and 1. Results using these datasets are averaged over ten experiments. For each experiment 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. This is repeated ten times. The results are reported as the error area under the ROC curve (AUC).

In the first set of experiments, we compare several approaches for building ensembles of classifiers using a feature perturbation approach. Several classifiers are tested for each method:

- Support vector machine (OS): where the best kernel and the best set of parameters are chosen in each dataset;
- RotationBoosting (RB): i.e., the method proposed in [15], in our experiments, 50 classifiers are combined;
- Our improved version of IDE [11] (DE): where each class is partitioned into several clusters, see section 3.1.
- Edited (ED), the ensemble based on adaboost of neural networks proposed in [20], where it is shown that it outperform the standard rotation forest.

In Table 1 we report the performance obtained by:

- SA: the stand-alone classifiers are used;
- RS: a random subspace ensemble of 50 classifiers;
- RR: the supervised subspace approach of [21] is used to create an ensemble of 50 classifiers. In this method the features are chosen according to their importance calculated using the mutual information.
- LM: the method Learn++.MF (see section 3.3) is used to build an ensemble of 50 classifiers.

The column *Rank* is reporting the average rank in the different datasets of the different classifiers (e.g., if a classifier always obtains the best performance in each dataset, its rank is 1).

Analyzing the results reported in Table 1, we can draw the following conclusions:

- The best performing ensemble methods are LM-DE and RS-OS.
- None of the tested classifiers generalizes better than any of the others, i.e., none outperforms any of the others across all the datasets.
- It is very interesting to note that RS ensembles prove quite useful; the RS-RB outperforms stand-alone RB, and the RS- OS outperforms stand-alone OS.
- RS-OS outperforms LM-DE; however, we want to stress the LM-DE uses the same parameters set in all the datasets.

In Table 2 we try to improve LM-DE performance in the following way:

- Missing: as proposed in [22], where a small percentage (5%) of features are replaced by mean imputation. In this way we create 50 different training set for building an ensemble of classifiers.
- Switch, it is the switching method proposed in [4]; also in this case 50 classifiers are built.
- PD: the principal direction linear oracle [18].
- RAND: the random spherical oracle where each method is based on 50% of the original features [23].

Analyzing the results reported in Table 2, we can draw the following conclusion: only PD improves (and only very slightly) the performance of LMF-DEC. In [24] it is shown that many types of classifier ensembles are improved by random oracle (one exception is the rotation forest). In our opinion, the performance of LMF-DEC is very high, so it probably could not be improved.

[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 projetced by PCA in a 50-dimensional space

Table 1. Experimental results #1: Comparison of several approaches for building ensembles of classifiers.

		HE	SO	PI	IO	BR	VE	VO	WD	CG	WI	HI	DE	RANK
SA	RB	0.9140	0.9156	0.8094	0.9812	0.9911	0.9394	0.9947	0.9968	0.7875	0.9970	0.9515	0.9998	9.75
	OS	0.8943	0.9522	0.8241	0.9811	0.9925	0.9460	0.9929	0.9962	0.8104	0.9982	0.9544	0.9995	6.58
RS	RB	0.9206	0.9331	0.8170	0.9847	0.9919	0.9396	0.9951	0.9961	0.7982	0.9990	0.9573	0.9995	6.66
	DE	0.9203	0.9198	0.8181	0.9864	0.9913	0.9446	0.9928	0.9956	0.8102	**0.9992**	0.9589	0.9986	6.66
	ED	0.9192	0.9284	0.8128	0.9830	0.9924	0.9120	0.9882	0.9952	0.8028	0.9989	0.9552	0.9997	8.66
	OS	0.9146	0.9595	0.8224	0.9799	0.9925	0.9425	0.9873	**0.9971**	**0.8134**	0.9984	0.9647	**1.0000**	**5.25**
RR	RB	0.9088	0.9152	0.8222	0.9844	0.9927	0.9272	0.9914	0.9959	0.7978	0.9986	0.9581	0.9995	7.91
	DE	0.9062	0.9181	**0.8243**	0.9854	0.9924	0.9294	0.9865	0.9952	0.8093	0.9986	0.9602	0.9987	8.25
	ED	0.9070	0.9445	0.8200	0.9819	**0.9934**	0.9042	0.9769	0.9945	0.8029	0.9986	0.9572	0.9993	9.00
	OS	0.8971	**0.9784**	0.8187	0.9784	0.9930	0.9258	0.9767	0.9976	0.8079	0.9985	0.9631	0.9995	7.66
LM	RB	0.9132	0.9354	0.8195	0.9830	0.9920	0.9415	**0.9954**	0.9964	0.7940	0.9988	0.9568	0.9998	6.83
	DE	0.9195	0.9327	0.8198	**0.9882**	0.9912	**0.9479**	0.9927	0.9956	0.8050	0.9991	0.9593	0.9991	6.16
	ED	**0.9215**	0.9277	0.8147	0.9816	0.9923	0.9135	0.9899	0.9956	0.8007	0.9987	0.9541	0.9997	8.91
	OS	0.9136	0.9604	0.8184	0.9811	0.9925	0.9451	0.9894	0.9958	0.8034	0.9982	**0.9666**	**1.0000**	6.66

Table 2. Experimental results #2: comparison of methods for improving LMF-DEC.

	HE	SO	PI	IO	BR	VE	VO	WD	CG	WI	HI	DE	RANK
LM-DE	0.9195	0.9327	0.8198	**0.9882**	0.9912	0.9479	0.9927	0.9956	0.8050	0.9991	0.9593	0.9991	2.8
OS	0.9146	**0.9595**	**0.8224**	0.9799	**0.9925**	0.9425	0.9873	**0.9971**	**0.8134**	0.9984	**0.9647**	**1.0000**	**2.6**
Missing	**0.9212**	0.9138	0.8179	0.9833	0.9916	0.9386	0.9918	0.9943	0.8121	0.9991	0.9584	0.9992	3.3
Switch	0.8988	0.8931	0.8081	0.9853	0.9895	0.9412	0.9908	0.9918	0.7924	0.9988	0.9361	0.9995	4.8
PD	0.9180	0.9408	0.8146	**0.9882**	0.9914	**0.9484**	**0.9939**	0.9959	0.8076	**0.9993**	0.9584	0.9991	2.7
RAND	0.9177	0.9180	0.8064	0.9832	0.9914	0.9401	0.9849	0.9961	0.8104	0.9988	0.9572	0.9989	4.5

Table 3. Experimental results #3: comparison of methods for pruning LMF-PDO-DEC.

	HE	SO	PI	IO	BR	VE	VO	WD	CG	WI	HI	DE	RANK
LM-PD-DE	0.9180	0.9408	0.8146	**0.9882**	0.9914	0.9484	**0.9939**	0.9959	0.8076	**0.9993**	0.9584	0.9991	2.9
OS	0.9146	**0.9595**	**0.8224**	0.9799	**0.9925**	0.9425	0.9873	**0.9971**	**0.8134**	0.9984	**0.9647**	**1.0000**	3.2
ORD	0.9180	0.9126	0.8058	0.9871	0.9912	0.9469	0.9924	0.9952	0.7523	0.9988	0.9433	0.9986	5.2
KNORA	**0.9197**	0.9250	0.8173	0.9863	0.9916	**0.9500**	0.9935	0.9958	0.8119	0.9991	0.9595	0.9992	**2.5**
SFFS1	0.9081	0.9007	0.8193	0.9862	0.9910	0.9489	0.9911	0.9942	0.8042	0.9985	0.9532	0.9992	5.1
SFFS2	0.9137	0.8978	0.8206	0.9863	0.9912	0.9475	0.9930	0.9958	0.8027	0.9992	0.9541	0.9992	4.5
SFFS3	0.9177	0.9134	0.8174	0.9874	0.9915	0.9487	0.9923	0.9951	0.8076	0.9983	0.9479	0.9992	4.4

Table 4. Experimental results #4: comparison of our approach with several other state-of-the-art approaches.

	HE	SO	PI	IO	BR	VE	VO	WD	CG	WI	HI	DE	RANK	AVE
LM-PD-DE	0.9180	0.9408	0.8146	**0.9882**	0.9914	**0.9484**	0.9939	0.9959	0.8076	**0.9993**	0.9584	0.9991	5.2	0.9463
OS	0.9146	**0.9595**	0.8224	0.9799	0.9925	0.9425	0.9873	0.9971	**0.8134**	0.9984	0.9647	**1.0000**	3.8	0.9477
FUS	0.9177	0.9497	0.8222	0.9871	0.9920	0.9456	0.9961	0.9968	0.8103	0.9996	0.9605	0.9997	**3.5**	**0.9481**
Real	0.8734 0.8910	0.8987 0.9156	0.7701 0.8010	0.9747 0.9804	0.9888 0.9908	0.9020 0.9271	0.9877 0.9909	0.9938 0.9950	0.7315 0.7330	***** *****	0.9363 0.9348	***** *****	***** *****	***** *****
Gentle	0.8827 0.8957	0.9078 0.9250	0.7707 0.7920	0.9708 0.9776	0.9857 0.9898	0.9165 0.9246	0.9891 0.9907	0.9947 0.9955	0.7449 0.7598	0.8885 0.9963	0.9331 0.9344	0.9970 0.9992	13.2 10.3	0.9151 0.9317
Modest	0.8741 0.8971	0.8979 0.9169	0.7919 0.8018	0.9687 0.9758	0.9877 0.9906	0.7415 0.8172	0.4358 0.4447	0.9936 0.9959	0.6895 0.7017	0.9877 0.9955	0.9337 0.9426	0.8354 0.8743	15.0 12.3	0.8448 0.8628
AdaN	0.8893 **0.9206**	0.9066 0.9351	0.7848 0.8189	0.9601 0.9777	0.9865 0.9907	0.9303 0.9369	0.9890 0.9827	0.9935 0.9959	0.7167 0.7832	0.9810 0.9990	0.9543 0.9568	0.9975 0.9997	12.5 7.4	0.9241 0.9414
ELM	0.8879 0.9161	0.7808 0.8991	0.8193 0.8224	0.8930 0.9752	**0.9938** 0.9932	0.9119 0.9265	0.9079 0.9568	0.9914 0.9955	0.7589 0.8073	0.9906 0.9981	0.8218 0.9584	0.9924 0.9995	13.2 8.0	0.8958 0.9373
GP	0.9030 0.9166	0.9411 0.9483	**0.8268** 0.8214	0.9711 0.9821	0.9924 0.9914	0.9462 0.9360	0.9833 **0.9951**	0.9961 0.9964	0.8017 0.8040	0.9980 0.9980	0.9630 0.9575	0.9997 **1.0000**	6.4 5.6	0.9435 0.9456
Gasen	0.8444 0.9201	0.8223 0.8897	0.8003 0.8219	0.9338 0.9833	0.9865 0.9921	0.9285 0.9438	0.9623 0.9884	0.9869 0.9970	0.7351 0.8107	0.9931 0.9990	0.9177 0.9547	0.9971 **1.0000**	14.2 5.3	0.9090 0.9417
RA-we	0.9120 0.9189	0.8857 0.8794	0.8159 0.8101	0.9476 0.9674	0.9908 0.9914	0.9220 0.9040	0.9254 0.9149	**0.9989** 0.9963	0.8038 0.8121	0.9992 0.9992	**0.9665** 0.9651	0.9998 0.9998	8.2 8.2	0.9306 0.9299

In Table 3 we test methods for pruning LMF-PD-DEC:

- ORD: the pruning method proposed in [25]. We retain the 100 best classifiers.
- KNORA: as proposed in [26].
- SFFS1: SFFS for selecting 100 classifiers; the fitness function is the AUC obtained by the ensemble in the training set.
- SFFS2: SFFS for selecting 100 classifiers; the fitness function is the AUC obtained by the ensemble in the training set + AUC obtained by the stand-alone classifier that we are choosing to decide whether to add or not to add it in the ensemble, in the training set.
- SFFS3: SFFS for selecting 100 classifiers; the fitness function is the AUC obtained by the ensemble in the training set and the Q-statistic[3] among the selected classifiers.

Analyzing the results reported in Table 3, we can conclude that only KNORA permits a slight performance improvement. Unfortunately, all the pruning methods reduce performance. We also tested other methods, such as using a genetic algorithm for weighing each classier or the pruning method reported in [27], but they obtained even lower performances.

As stated above, in our opinion the extremely high performance of LM-PD-DE is so high that it probably cannot be improved. It may be the case that pruning methods using it should be studied using very large datasets where a large validation set could be extracted.

In Table 4 we compare our approach with several other state-of-the-art approaches (for a fair comparison we used the MATLAB code shared by the original developers of each method). Each cell in Table 4 contains two values: the first is the performance obtained using the standard approach, and the second is the performance obtained using a random subspace of 50 classifiers. FUS is the fusion by sum rule of GPC and our LM-PD-DE. Our aim is to investigate whether a heterogeneous ensemble improves the two highest performing systems.

We compare the following methods in experiment 4:

- Real: RealAdaboost as implemented in GML AdaBoost Matlab Toolbox using the decision tree as classifier [28];
- Gentle: GentleAdaboost as implemented in GML AdaBoost Matlab Toolbox using the decision tree as classifier [29];
- Modest: ModestAdaboost as implemented in GML AdaBoost Matlab Toolbox using the decision tree as classifier [30];
- AdaN: AdaBoost.M2 using the neural network as classifier.
- ELM [31]: Extreme Learning Machine where the type of activation and the number of hidden neurons is optimized in each dataset http://www3.ntu.edu.sg/home/egbhuang/
- GP[4]: the Gaussian process classifier [32].
- Gasen [33]: selective ensemble method using genetic algorithm to select a subset of neural networks http://lamda.nju.edu.cn/datacode/GASEN/gasen.htm. As the validation set, we have tested two methods: 1) all the training set; and 2) a subset of patterns is extracted from the training set and used as validation set. For each dataset we have reported only the best method.
- RA-we: the ensemble of modified RealAdaboost proposed in [34]; in this approach the neural network is used as classifier.

Analyzing the results reported in Table 5, we observe:

- GPC obtains a performance that is similar to SVM,
- It is very interesting to note that RS ensembles are shown to be very useful when coupled with some methods. Gasen, ELM and several kinds of AdaBoost (all, except RA-we) are greatly improved when coupled with random subspace.

We want to stress that the fusion method named FUS outperforms a finely tuned ensemble of SVMs, where a different kernel is used for each dataset along with a different set of parameters.

5 Conclusion and Discussion

The goal of this paper was to discover methods for building a generalized ensemble of classifiers requiring little or no parameter tuning that performs well across an array of different problems. We performed an empirical comparison of several multiclassifier systems using several benchmark datasets that address very different problems. Our experimental results show that our new ensembles of decision trees outperform state-of-the-art stand-alone and ensemble methods.

Unfortunately, it was not possible for us to find a single ensemble method that outperformed all the other classifiers across all the tested datasets (the "no free lunch" theorem). Nonetheless, some practical findings are reported. We show that the highest performance is obtained by combining a "supervised" RS with a cluster-based input decimated ensemble and the principal direction oracle.

.

6 References

[1] Kuncheva, L. I., and Whitaker, C. J., "Measures of Diversity in Classifier Ensembles and their Relationship

[3] It is a measure of the statistical independence among a set of classifiers

[4] The software is available at http://www.gaussianprocess.org/gpml/, the parameters are set *loghyper* = [0.0; 2.0];

with the ensemble accuracy," *Machine Learning,* vol. 51, no. 2, pp. 181-207, 2003.

[2] Kittler, J., "On combining classifiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 3, pp. 226–239, 1998.

[3] Breiman, L., "Bagging predictors," *Machine Learning,* vol. 24, no. 2, pp. 123-140, 1996.

[4] Martínez-Muñoz, G., and Suárez, A., "Switching class labels to generate classification ensembles," *Pattern Recognition,* vol. 38, no. 10, pp. 1483 – 1494, 2005.

[5] Melville, P., and Mooney, R. J., "Creating Diversity in Ensembles Using Artificial, Information Fusion," *Special Issue on Diversity in Multiclassifier Systems,* vol. 6, no. 1, pp. 99-111, 2005.

[6] Freund, Y., and Schapire, R. E., "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of Computer and System Sciences,* vol. 55, no. 1, pp. 119 - 139, 1997.

[7] Bologna, G., and Appel, R. D., "A comparison study on protein fold recognition." pp. 2492-2496.

[8] Ho, T. K., "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 20, no. 8, pp. 832-844, 1998.

[9] Nanni, L., "Cluster-based pattern discrimination: A novel technique for feature selection," *Pattern Recognition Letters,* vol. 27, no. 6, pp. 682-687, 2006.

[10] Tumer, K., and Oza, N. C., "Input decimated Ensembles," *Pattern Analysis Application,* vol. 6, pp. 65-77, 2003.

[11] Nanni, L., and Lumini, A., "Ensemble generation and feature selection for the identification of students with learning disabilities,," *Expert System with Applications,* vol. 36, no. 2, pp. 3896-3900, 2009.

[12] Lan, H., Carson, R., Provart, N., and Bonner, A., "Combining classifiers to predict gene function in arabidopsis thaliana using large-scale gene expression measurements," *BMC Bioinformatics,* vol. 8, pp. 358, 2007.

[13] Breiman, L., "Random Forest," *Machine Learning,* vol. 45, no. 1, pp. 5-32, 2001.

[14] Rodriguez, J. J., Kuncheva, L. I., and Alonso, C. J., "Rotation forest: a new classifier ensemble method," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, no. 10, pp. 1619-1630, 2006.

[15] Zhang, C.-X., and Zhang, J.-S., "RotBoost: a technique for combining Rotation Forest and AdaBoost,," *Pattern Recognition Letters,* vol. 29, no. 10, pp. 1524-1536, 2008.

[16] Kuncheva, L. I., *Combining pattern classifiers: Methods and algorithms*, New York: Wiley, 2004.

[17] Polikar, R., DePasquale, J., Syed Mohammed, H., Brown, G., and Kuncheva, L. I., "Learn++.MF : A Random Subspace Approach for the Missing Feature Problem," *Pattern Recognition,* vol. 43, no. 1, pp. 3817-3832, 2010.

[18] Peterson, L. E., and Coleman, M. A., "Principal direction linear oracle for gene expression ensemble classification," in Computational Intelligence Approaches for the Analysis of Bioinformatics Data (CIBIO07), 2001.

[19] Xiaofei, H., Cai, D., Yan, S., and Zhang, H.-J., "Neighborhood preserving embedding."

[20] Nanni, L., and Franco, A., "Reduced Reward-Punishment Editing for building ensembles of classifiers," *Expert Systems With Applications,* vol. 38, no. 3, pp. 2395-2400, 2011.

[21] Yaslan, Y., and Cataltepe, Z., "Co-training with Relevant Random Subspaces, ," *Neurocomputing,* vol. 73, no. 10-12, pp. 1652-1661, 2010.

[22] Su, X., Khoshgoftaar, T. M., and Zhu, X., "VoB predictors: Voting on bagging classifications," in ICPR, 2008, pp. 1-4.

[23] Rodríguez, J. J., and Kuncheva, L. I., "Naïve Bayes Ensembles with a Random Oracle," in MCS, 2007, pp. 450-458.

[24] Kuncheva, L. I., and Rodríguez, J. J., "Classifier ensembles with a random linear oracle," *IEEE Transactions on Knowledge and Data Engineering,* vol. 19, no. 4, pp. 500-508, 2007.

[25] Martinez-Munoz, G., Hernandez-Lobato, D., and Suarez, A., "An analysis of ensemble pruning techniques based on ordered aggregation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 31, no. 2, pp. 245-259, 2008.

[26] Ko, A. H. R., Sabourin, R., and de Souza Britto Jr., A., "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognition,* vol. 41, no. 5, pp. 1718-1731, 2008.

[27] Zhang, L., and Zhou, W.-D., "Sparse ensembles using weighted combination methods based on linear programming," *Pattern Recognition,* vol. 44, no. 1, pp. 97-106, 2011.

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

[29] Friedman, J., Hastie, T., and Tibshirani, R., "Additive logistic regression: A statistical view of boosting," *The Annals of Statistics,* vol. 38, no. 2, pp. 337–374, 2000.

[30] Vezhnevets, A., and Vezhnevets, V., "'Modest AdaBoost' - Teaching AdaBoost to Generalize Better," in Graphicon, Novosibirsk Akademgorodok, Russia, 2005.

[31] Huang, G.-B., Wang, D. H., and Lan, Y., "Extreme learning machines: A Survey," *International Journal of Machine Leaning and Cybernetics,* vol. 2, no. 2, pp. 107-122, 2011.

[32] Rasmussen, C. E., and Williams, K. I., *Gaussian Processes for Machine Learning*: The MIT Press, 2006.

[33] Zhou, Z.-H., Wu, J., and Tang, W., "Ensembling neural networks: many could be better than all," *Artificial Intelligence,* vol. 137, no. 1-2, pp. 239-263, 2002.

[34] Gómez-Verdejo, V., Arenas-García, J., and Figueiras-Vidal, A. R., "Committees of Adaboost ensembles with modified emphasis functions," *Neurocomputing archive,* vol. 73, no. 7-9, pp. 1289-1292, 2010.