Decision Fusion on Boosting Ensembles

Joaquín Torres-Sospedra, Carlos Hernández-Espinosa, and Mercedes Fernández-Redondo

Departamento de Ingenieria y Ciencia de los Computadores, Universitat Jaume I, Avda. Sos Baynat s/n, C.P. 12071, Castellon, Spain
{jtorres,espinosa,redondo}@icc.uji.es

Abstract. Training an ensemble of neural networks is an interesting way to build a Multi-net System. One of the key factors to design an ensemble is how to combine the networks to give a single output. Although there are some important methods to build ensembles, *Boosting* is one of the most important ones. Most of methods based on *Boosting* use an specific combiner (*Boosting Combiner*). Although the *Boosting combiner* provides good results on boosting ensembles, the results of previouses papers show that the simple combiner *Output Average* can work better than the *Boosting combiner*. In this paper, we study the performance of sixteen different combination methods for ensembles previously trained with *Adaptive Boosting* and *Average Boosting*. The results show that the accuracy of the ensembles trained with these original boosting methods can be improved by using the appropriate alternative combiner.

1 Introduction

One technique often used to increase the generalization capability with respect to a single neural network consists on training an ensemble of neural networks . This procedure consists on training a set of neural network with different weight initialization or properties in the training process and combining them in a suitable way.

The two key factors to design an ensemble are how to train the individual networks and how to combine the outputs provided by the networks to give a single output. Among the methods of training the individual networks and combining them there are an important number of alternatives. Our research group has performed some comparisons on methods to build and combine ensembles.

Reviewing the bibliography we can see that *Adaptive Boosting* (*Adaboost*) is one of the best performing methods to create an ensemble [3]. *Adaboost* is a method that construct a sequence of networks which overfits the training set used to train a neural network with hard to learn patterns. A sampling distribution is used to select the patterns we use to train the network.

In previouses papers, we successfully proposed three new boosting methods [13, 14, 15]. In those papers we noticed that in the majority of cases the *Output average* was better than the specific *Boosting combiner*.

Some authors like Breiman [1], Kuncheva [9] or Oza [11] have deeply studied and successfully improved *Adaboost* but any study on combining boosting methods has not been done.

In this paper, we present a comparison of sixteen different combiners on ensembles previously trained with *Adaboost* and *Aveboost*, two of the most important boosting methods, in order to test if the *Boosting combiner* is the most appropriate way to combine boosting ensembles.

This paper is organized as follows. Firstly, some theoretical concepts are reviewed in section 2. Then, the ten databases used and the experimental setup are described in section 3. Finally, the experimental results and their discussion are in section 4.

2 Theory

2.1 Adaptive Boosting - Adaboost

In *Adaboost*, the successive networks are trained with a training data set T' selected at random from the original training data set T, the probability of selecting a pattern from T is given by a *sampling distribution* associated to the network $Dist_{net}$. The sampling distribution associated to a network is calculated when the previous network learning process has finished. *Adaboost* is described in algorithm 1.

Algorithm 1. AdaBoost $\{T, V, k\}$

```
Initialize Sampling Distribution: $Dist_x^1 = 1/m \; \forall x \in T$
for $net = 1$ to $k$ do
    Create $T^{'}$ sampling from $T$ using $Dist^{net}$
    MF Network Training $T'$ , $V$
    Calculate missclassified vector:
        $miss_x^{net} = \begin{cases} 0 \text{ if } x \text{ is correctly classified} \\ 1 \text{ otherwise} \end{cases}$
    Calculate error:
        $\epsilon_{net} = \sum_{x=1}^{m} Dist_x^{net} \cdot miss_x^{net}$
    Update sampling distribution:
        $Dist_x^{net+1} = Dist_x^{net} \cdot \begin{cases} \frac{1}{(2\epsilon_{net})} & \text{if } miss_x^{net} \\ \frac{1}{2(1-\epsilon_{net})} & \text{otherwise} \end{cases}$
end for
```

Adaboost and *Aveboost* use an specific combination method, *Boosting combiner*, to combine the networks and get the final output or hypothesis eq.1.

$$h(x) = \underset{c=1,\ldots,classes}{\arg\max} \sum_{net:h^{net}(x)=c}^{k} \log \frac{1-\epsilon_{net}}{\epsilon_{net}} \tag{1}$$

2.2 Averaged Boosting - Aveboost

Oza proposed in [11] *Averaged Boosting* (Algorithm 2). *Aveboost* is a method based on *Adaboost* in which the sampling distribution related to a neural network is also based on the number of networks previously trained. The whole description is detailed in algorithm 2.

Algorithm 2. Aveboost $\{T, V, k\}$

```
Initialize Sampling Distribution: $Dist_x^1 = 1/m \; \forall x \in T$
for $net = 1$ to $k$ do
    Create $T^{'}$ sampling from $T$ using $Dist^{net}$
    MF Network Training $T'$ , $V$
    Calculate missclassified vector:
        $miss_x^{net} = \begin{cases} 0 \text{ if } x \text{ is correctly classified} \\ 1 \text{ otherwise} \end{cases}$
    Calculate error:
        $\epsilon_{net} = \sum_{x=1}^{m} Dist_x^{net} \cdot miss_x^{net}$
    Update sampling distribution:
        $C_x^{net} = Dist_x^{net} \cdot \begin{cases} \frac{1}{(2\epsilon_{net})} & \text{if } miss_x^{net} \\ \frac{1}{2(1-\epsilon_{net})} & \text{otherwise} \end{cases}$
        $Dist_x^{net+1} = \frac{net \cdot Dist_x^{net} + C_x^{net}}{net+1}$
end for
```

2.3 Alternative Combiners

In this subsection, we briefly review the alternative combiners we have used to obtain the experimental results.

Average. This approach simply averages the individual classifier outputs across the different classifiers. The output yielding the maximum of the averaged values is chosen as the correct class.

Majority Vote. Each classifier provides a vote to a class, given by the highest output. The correct class is the one most often voted by the classifiers.

Winner Takes All (WTA). In this method, the class with overall maximum output across all classifier and outputs is selected as the correct class.

Borda Count. For any class c, the *Borda count* is the sum of the number of classes ranked below c by each classifier [5, 16]. The class with highest count is selected as correct class.

Bayesian Combination. This combination method was proposed in references [19]. According to this reference a belief value that the pattern x belongs to class c can be approximated by the following equation based on the values of the confusion matrix [16]

$$Bel(c) = \frac{\prod_{net=1}^{k} P(x \in q_c | \lambda_{net}(x) = j_{net})}{\sum_{i=1}^{classes} \prod_{net=1}^{k} P(x \in q_i | \lambda_{net}(x) = j_{net})} \tag{2}$$

Weighted Average. This method introduces weights to the outputs of the different networks prior to averaging. The weights try to minimize the difference between the output of the ensemble and the *desired or true* output. The weights can be estimated from the error correlation matrix. The full description of the method can be found in [8, 16].

Choquet Integral. This method is based in the fuzzy integral [2, 4] and the Choquet integral. The method is complex, its full description can be found in [16].

Fuzzy Integral with Data Dependent Densities. It is another method based on the fuzzy integral and the Choquet integral. But in this case, prior to the application of the method it is performed a partition of the input space into n regions by frequency sensitive learning algorithm (*FSL*). The full description can be found in reference [16].

Weighted Average with Data Dependent weights. This method is the weighted average described above. But in this case, a partition of the space is performed by *FSL* algorithm and the weights are calculated for each partition. We have a different combination scheme for the different partitions of the space. The method is fully described in [16].

BADD Defuzzification Strategy. It is another combination method based on fuzzy logic concepts. The method is complex and the description can also be found in [16].

Zimmermann's Compensatory Operator. This combination method is based in the Zimmermann's compensatory operator described in [20]. The method is complex and can be found in [16].

Dynamically Averaged Networks. Two versions of *Dinamically Averaged Networks* were proposed by Jimenez [6, 7]. In these methods instead of choosing static weights derived from the network output on a sample of the input space, we allow the weights to adjust to be proportional to the certainties of the respective network output.

Nash Vote. In this method each voter assigns a number between zero and one for each candidate output. The product of the voter's values is compared for all candidates. The higher is the winner. The method is reviewed in reference [17].

Stacked Combiners (Stacked and Stacked+). The training in *Stacked Generalization* is divided into two steps. In the first one, the expert networks are trained. In the second one, the combination networks are trained with the outputs provided by the experts.

Stacked Generalization [18] can be adapted to combine ensembles of neural networks if the networks of the ensembles are used as expert networks. In [12], *Stacked* and *Stacked+*, two combiners based on *Stacked Generalization*, were successfully proposed.

3 Experimental Setup

In the experiments, the *Boosting combiner* and the alternative combiners have been applied on ensembles of 3, 9, 20 and 40 *MF* networks previously trained with *Adaptive*

Boosting and *Averaged Boosting* on the databases described in subsection 3.1 using the training parameters described in table 1. In the case of *Stacked combiners*, a single *MF* combination network has been applied.

Moreover, we have repeated the whole learning process 10 times using different training, validation and test sets. With this procedure we can obtain a mean performance of the ensemble for each database and an error in the performance calculated by standard error theory.

3.1 Datasets

We have used the following ten classification problems from the *UCI repository of machine learning databases* [10]:*Arrhythmia* (aritm), *Dermatology* (derma), *Ecoli* (ecoli), *Solar Flares* (flare), *Image segmentation* (img), *Ionosphere Database* (ionos), *Pima Indians Diabetes* (pima), *Haberman's Survival Data* (survi), *Vowel Database* (vowel) and *Wisconsin Breast Cancer* (wdbc).

The optimal parameters of the *Multilayer Feedforward* networks (*Hidden units*, *Adaptation step*, *Momentum rate* and *Number of iterations*) we have used to train the networks of the ensembles are shown in table 1.

Table 1. MF training parameters

database	*hidden*	*step*	*mom*	*ite*	*accuracy*
aritm	9	0.1	0.05	2500	75.6 ± 0.7
derma	4	0.1	0.05	1000	96.7 ± 0.4
ecoli	5	0.1	0.05	10000	84.4 ± 0.7
flare	11	0.6	0.05	10000	82.1 ± 0.3
img	14	0.4	0.05	1500	96.3 ± 0.2
ionos	8	0.1	0.05	5000	87.9 ± 0.7
pima	14	0.4	0.05	10000	76.7 ± 0.6
survi	9	0.1	0.2	20000	74.2 ± 0.8
vowel	15	0.2	0.2	4000	83.4 ± 0.6
wdbc	6	0.1	0.05	4000	97.4 ± 0.3

The optimal parameters of the *Multilayer Feedforward* networks we have used to train the combination networks of combiners *Stacked* and *Stacked+* on ensembles trained with *Adaboost* and *Aveboost* is shown in table 2.

Finally, we set to $n = 5$ the numbers of regions used in the combiners based on *data depend densities*. The parameters have been set after an exhaustive trial and error procedure using the training and validation sets.

4 Results and Discussion

Due to the lack of space, the general results on combining ensembles trained with *Adaboost* and *Aveboost* are shown in this section instead of showing the complete results. The general measurements used in this paper are described in subsection 4.1.

Table 2. Training parameters - Combiners Stacked and Stacked+

		Adaboost								*Aveboost*							
		Stacked				*Stacked+*				*Stacked*				*Stacked+*			
	nets	*h.u*	*step*	*mom*	*ite*	*h.u*	*step*	*mom*	*ite*	*h.u*	*step*	*mom*	*ite*	*h.u*	*step*	*mom*	*ite*
aritm	3	30	0.40	0.10	500	15	0.40	0.05	1750	30	0.40	0.01	500	24	0.40	0.20	4000
	9	24	0.05	0.05	500	14	0.40	0.20	500	19	0.40	0.20	500	21	0.40	0.20	1750
	20	25	0.05	0.01	500	5	0.40	0.20	500	28	0.05	0.20	500	30	0.10	0.20	500
	40	2	0.20	0.10	500	24	0.20	0.20	500	2	0.05	0.01	500	9	0.40	0.20	500
derma	3	4	0.40	0.20	2500	3	0.40	0.20	7500	24	0.10	0.10	1500	3	0.40	0.20	1500
	9	3	0.10	0.20	4000	3	0.10	0.05	7500	3	0.40	0.05	3000	3	0.20	0.10	3000
	20	3	0.20	0.20	6000	3	0.05	0.10	7500	3	0.20	0.05	4000	3	0.40	0.10	3000
	40	5	0.10	0.10	7500	3	0.10	0.01	2500	4	0.10	0.20	7500	3	0.40	0.05	1500
ecoli	3	3	0.10	0.20	1750	4	0.20	0.05	1750	3	0.40	0.05	1500	11	0.40	0.05	2500
	9	5	0.10	0.20	1500	3	0.40	0.10	6500	6	0.40	0.01	1500	7	0.40	0.10	4000
	20	25	0.40	0.20	500	26	0.40	0.20	500	29	0.40	0.20	500	7	0.40	0.05	5000
	40	19	0.40	0.10	1500	16	0.40	0.01	1500	25	0.10	0.01	2500	23	0.10	0.10	1750
flare	3	20	0.10	0.05	7500	2	0.10	0.01	7500	28	0.05	0.01	500	9	0.05	0.05	7500
	9	11	0.05	0.20	7500	30	0.05	0.10	500	30	0.05	0.10	500	21	0.40	0.01	4500
	20	16	0.10	0.05	2500	30	0.20	0.20	500	8	0.40	0.10	7500	27	0.05	0.05	500
	40	2	0.40	0.20	500	2	0.40	0.05	500	30	0.40	0.05	1500	5	0.10	0.10	6500
img	3	3	0.40	0.05	1500	5	0.40	0.20	500	13	0.40	0.20	500	8	0.40	0.10	500
	9	2	0.40	0.01	4000	3	0.10	0.05	1500	3	0.05	0.20	1500	2	0.20	0.10	7500
	20	2	0.40	0.05	4500	2	0.40	0.05	2500	2	0.20	0.01	3000	3	0.05	0.20	1500
	40	2	0.20	0.01	1750	3	0.05	0.05	7500	4	0.10	0.05	1500	3	0.05	0.05	1500
ionos	3	3	0.40	0.01	500	29	0.05	0.05	500	18	0.05	0.01	500	8	0.05	0.01	1500
	9	3	0.20	0.10	500	5	0.05	0.05	500	6	0.40	0.20	500	29	0.05	0.05	500
	20	2	0.40	0.20	500	2	0.40	0.05	500	16	0.05	0.20	500	21	0.05	0.05	500
	40	23	0.05	0.10	500	19	0.05	0.01	500	2	0.05	0.20	500	3	0.10	0.10	500
pima	3	28	0.05	0.10	500	30	0.05	0.20	500	2	0.20	0.20	1750	20	0.05	0.10	500
	9	7	0.20	0.10	500	13	0.20	0.10	500	16	0.10	0.05	500	21	0.40	0.05	500
	20	9	0.05	0.20	500	14	0.05	0.20	500	10	0.20	0.05	500	7	0.20	0.10	500
	40	10	0.05	0.01	500	4	0.05	0.05	500	7	0.20	0.20	500	20	0.40	0.20	1750
survi	3	2	0.40	0.20	1500	9	0.40	0.05	3000	5	0.40	0.20	500	10	0.20	0.20	500
	9	6	0.20	0.20	7500	2	0.40	0.20	1500	9	0.40	0.01	6500	8	0.20	0.20	4000
	20	2	0.40	0.20	1500	8	0.40	0.01	7500	25	0.20	0.20	3000	6	0.40	0.20	500
	40	3	0.40	0.05	1500	2	0.40	0.20	1500	24	0.40	0.10	6000	21	0.40	0.20	1500
vowel	3	14	0.10	0.05	1500	4	0.20	0.20	2500	11	0.05	0.01	5000	19	0.20	0.20	3000
	9	24	0.40	0.10	1500	11	0.20	0.01	7500	5	0.40	0.10	4000	6	0.10	0.01	7500
	20	6	0.10	0.05	6500	4	0.05	0.05	7500	6	0.10	0.05	7500	7	0.20	0.01	7500
	40	11	0.10	0.05	7500	12	0.05	0.20	7500	6	0.40	0.01	7500	4	0.20	0.01	4000
wdbc	3	29	0.40	0.20	500	29	0.40	0.20	500	30	0.40	0.20	500	28	0.40	0.20	500
	9	11	0.40	0.20	3000	14	0.40	0.20	7500	30	0.10	0.05	500	29	0.05	0.10	500
	20	15	0.05	0.20	4500	13	0.10	0.10	2500	30	0.20	0.01	500	30	0.10	0.05	500
	40	28	0.05	0.01	500	30	0.05	0.10	500	28	0.10	0.01	500	30	0.05	0.10	500

4.1 General Measurements

In our experiments, we have calculated the *Increase of Performance* (IoP eq.3) and the *Percentage of Error Reduction* (PER eq.4) of the results with respect to a single network in order to perform an exhaustive comparison. The IoP value is an absolute measurement whereas the PER value is a relative measurement. A negative value on these measurements mean that the ensemble performs worse than a single network.

$$IoP = Error_{SingleNet} - Error_{Ensemble} \tag{3}$$

$$PER = 100 \cdot \frac{Error_{SingleNet} - Error_{Ensemble}}{Error_{SingleNet}} \tag{4}$$

Finally, we have calculated the mean IoP and the mean PER across all databases to get a general measurement to compare the methods presented in the paper. The results on combining *Adaboost* are presented in subsection 4.2 whereas the results on combining *Aveboost* are in subsection 4.3.

4.2 Adaboost Results

In this subsection the results of the different combiners on ensembles trained with *Adaptive Boosting* are shown. Table 3 shows the mean IoP and the mean PER for the ensembles trained and combined with the *Boosting combiner* as in the original method *Adaboost* and for the same ensembles combined with the alternative combiners described in subsection 2.3.

Table 3. Adaptive Boosting - Mean IoP and PER among all databases

	Mean IoP				*Mean PER*			
Method	*3 Nets*	*9 Nets*	*20 Nets*	*40 Nets*	*3 Nets*	*9 Nets*	*20 Nets*	*40 Nets*
adaboost	0.3	0.86	1.15	1.26	1.33	4.26	9.38	12.21
average	−0.18	−0.15	−0.36	−0.35	−20.49	−19.71	−18.81	−22.05
voting	−0.97	−1.48	−1.74	−1.26	−27.18	−27.2	−25.53	−26.21
wta	−1.07	−4.78	−8.22	−11.07	−16.66	−78.22	−132.97	−184.84
borda	−2.74	−2.76	−2.84	−2.07	−50.55	−35.57	−32.71	−32.13
bayesian	−0.36	−1.28	−3.22	−5.46	−6.28	−16.17	−38.15	−65.05
wave	0.59	1.19	0.41	0.44	1.15	6.5	3.7	7.3
choquet	−1.26	−6.23	−	−	−23.68	−107.65	−	−
fidd	−1.37	−6.91	−	−	−27.85	−123.9	−	−
wave dd	0.68	0.72	−	−	4.18	0.67	−	−
badd	−0.18	−0.15	−0.36	−0.35	−20.49	−19.71	−18.81	−22.05
zimm	0.35	−1.16	−13.37	−13.02	−0.28	−28.25	−150.69	−212.35
dan	−12.54	−17.08	−20.23	−19.13	−123.17	−199.31	−244.94	−278.47
dan2	−12.59	−17.45	−20.46	−19.47	−123.5	−202.67	−248.34	−282.06
nash	−1.27	−1.76	−1.71	−1.57	−16.55	−30.14	−28.14	−29.13
stacked	0.69	0.83	0.7	0.51	3.39	2.87	4.7	4.27
stacked+	0.71	0.95	0.64	−0.14	5.43	6.25	4.39	1.81

Table 4. Averaged Boosting - Mean IoP and PER among all databases

	Mean IoP				*Mean PER*			
Method	*3 Nets*	*9 Nets*	*20 Nets*	*40 Nets*	*3 Nets*	*9 Nets*	*20 Nets*	*40 Nets*
aveboost	0.5	1.49	1.83	1.82	1.13	10.46	11.7	10.79
average	0.87	1.61	2	1.8	4.26	11.64	12.93	12.99
voting	0.37	1.54	1.76	1.91	0.28	11.15	12.67	13.01
wta	0.49	0.36	-0.38	-0.88	-0.1	-2.31	-9.2	-10.88
borda	-0.34	1.15	1.57	1.73	-6.73	8.13	11.05	12.12
bayesian	0.02	-0.13	-1.3	-2.87	-4.14	-7.94	-23.39	-40.81
wave	0.85	1.17	1.19	0.32	4.29	8.36	7.65	3.78
choquet	0.21	-0.19	$-$	$-$	-3.69	-10.02	$-$	$-$
fidd	0.14	-0.35	$-$	$-$	-3.76	-11.14	$-$	$-$
wave dd	0.92	1.62	$-$	$-$	5.62	11.88	$-$	$-$
badd	0.87	1.61	2	1.8	4.26	11.64	12.93	12.99
zimm	0.74	0.59	-2.75	-7.53	4.17	5.17	-18.5	-63.01
dan	-2.65	-3.04	-5.06	-5.13	-30.32	-25.63	-34.57	-34.21
dan2	-2.64	-3.13	-5.1	-5.27	-30.37	-26.56	-34.89	-35.38
nash	-0.09	1.03	1.63	1.4	-2.56	7.33	11.34	8.86
stacked	0.9	0.99	0.95	0.96	6.67	7.79	7.15	8.43
stacked+	0.95	1.02	0.83	0.8	6.42	6.22	7.64	6.84

Table 5. Adaboost - Best performance

	Boosting Combiner		*Alternative Combiners*		
Database	*Performance*	*Nets*	*Performance*	*Method*	*Nets*
aritm	73.8 ± 1.1	40	75.3 ± 0.9	$bayes$	9
derma	98 ± 0.5	3	98.1 ± 0.7	$stacked+$	3
ecoli	86 ± 1.3	20	87.2 ± 1.0	$w.ave$	3
flare	81.7 ± 0.6	3	82.2 ± 0.6	$w.ave$	3
img	97.3 ± 0.2	20	97.4 ± 0.3	$average$	20
ionos	91.6 ± 0.7	40	92 ± 0.9	$average$	20
pima	75.7 ± 1.0	3	76.6 ± 1.1	$average$	3
survi	75.4 ± 1.6	3	74.8 ± 1.0	$zimm$	3
vowel	97 ± 0.6	40	97.1 ± 0.6	$average$	40
wdbc	96.7 ± 0.9	40	96.6 ± 0.6	$bayes$	20

Moreover, table 5 shows the best performance for each database on ensembles trained with the original *Adaboost* (applying the *Boosting combiner*). The table also shows the best performance of these ensembles combined with the sixteen alternative combiners.

4.3 Aveboost Results

In this subsection the results of the different combiners on ensembles trained with *Averaged Boosting* are shown. Table 4 shows the mean IoP and the mean PER for the ensembles trained and combined with the *Boosting combiner* as in the original method *Aveboost* and for the same ensembles combined with the alternative combiners.

Table 6. Aveboost - Best performance

	Boosting Combiner		*Alternative Combiners*		
Database	*Performance*	*Nets*	*Performance*	*Method*	*Nets*
aritm	76.3 ± 1.0	40	77.0 ± 1.1	$average$	20
derma	97.9 ± 0.5	20	97.8 ± 0.6	$w.avedd$	3
ecoli	86.5 ± 1.2	9	87.6 ± 0.9	$w.ave$	3
flare	82.4 ± 0.7	20	82.5 ± 0.6	$stacked+$	3
img	97.5 ± 0.2	40	97.6 ± 0.2	$stacked$	40
ionos	91.6 ± 0.9	40	92.4 ± 1	$zimm$	3
pima	76.6 ± 1.0	9	77.1 ± 1	$w.ave$	3
survi	75.1 ± 1.2	3	75.1 ± 1.2	$voting$	3
vowel	96.4 ± 0.6	40	96.7 ± 0.4	$w.ave$	40
wdbc	96.6 ± 0.4	9	96.7 ± 0.3	$zimm$	9

Moreover, table 6 shows the best performance for each database on ensembles trained with the original *Aveboost* (applying the *Boosting combiner*). The table also shows the best performance of these ensembles combined with the sixteen alternative combiners.

4.4 Discussion

We see that the *Boosting combiner* is not the best alternative in *Adaboost* in all cases, *Stacked combiners* and *Weighted Average with Data-Depend Densities* are better (table 3) when the number of networks is reduced. If we analyse table 5, we can see that the *boosting combiner* only provides the best result on databases *survi* and *wdbc*.

We can also see in *Aveboost* that, in the majority of cases, the mean IoP and PER of the boosting combiner is always lower than for the *Output average* (table 4). Moreover, the *boosting combiner* only provides the best result for database *derma* (table 6).

Moreover, we can notice that in some cases the accuracy is highly improved by applying an alternative combiner while the number of networks required to get the best performance is reduced. We got better results by combining a smaller set of networks.

5 Conclusions

In this paper, we have performed a comparison among sixteen combiners on ensembles trained with *Adaptive Boosting* and *Averaged Boosting*. To carry out our comparison we have used ensembles of 3, 9, 20 and 40 networks previously trained with *Adaptive boosting* and *Averaged boosting* and the accuracy of the ensemble using the *Boosting Combiner*. In our experiments we have selected ten databases whose results are not easy to improve with an ensemble of neural networks. Alternatively, we have applied sixteen different combiners on these ensembles to test if the *boosting combiner* is the best method to combine the networks of a *boosting ensemble*. Moreover, we also want to know which is the most appropriate combiner in each case. Finally, we have calculated the mean *Increase of Performance* and the mean *Percentage of Error Reduction* with respect to a single network to compare the combiners. Furthermore, the best accuracy for each database with the original methods, *Boosting combiner* on *Adaboost* and *Aveboost*, and applying the sixteen alternative combiners on these ensembles have been shown.

According the general results, the *Boosting combiner* is not the most effective way to combine an ensemble trained with *Boosting* in a wide number of cases. The original results have been improved with the use of the appropriate alternative combiner. In general, the *Output average*, the *Weighted average* and the *Stacked combiners* are the best combiners on ensembles trained with *Adaboost*. In a similar way, the *Output average*, the *Voting* and the *Borda Count* are the best combiners on ensembles trained with *Aveboost*.

According the best performance for each database (tables 5 and 6), we can see that the *Output average* and both versions of the *Weighted average* should be seriously considered for combining *boosting ensembles* because the *Boosting combiner* provides the best results only in 16.6% of the cases. In addition, in a some cases not only have the accuracy of the ensembles been improved with the use of an alternative combiner, the numbers of networks to get the best result is also reduced. For instance, the best accuracy for database *ecoli* using the original *Adaboost* (applying the *Boosting combiner*) was got with the 20-network ensembles (86 ± 1.3). The best overall accuracy for this database using *Adaboost* was got by applying the *Weighted average* to the 3-networks ensembles (87.2 ± 1.0). The accuracy was improved in 1.2%, the error rate was reduced in 0.3% and the number of networks required were reduced from 20 to 3.

Nowadays, we are extending the comparison we have performed adding more methods and databases. The results we are getting also show that the *Boosting combiner* does not provide either the best general results (best mean IoP or PER) or the best performance for each database. Furthermore, we are working on an advanced combination method based on the *boosting combiner* we think could increase the accuracy of the *Boosting ensembles*.

We can conclude by remarking that the accuracy of a *boosting ensemble* can be improved and its size can be reduced by applying the *Output average* or advanced combiners like the *Weighted average* or the *Stacked combiners* on ensembles previously trained with *Adaboost* or *Aveboost*.

References

1. Breiman, L.: Arcing classifiers. The Annals of Statistics 26(3), 801–849 (1998)
2. Cho, S.-B., Kim, J.H.: Combining multiple neural networks by fuzzy integral for robust classification. IEEE Transactions on System, Man, and Cybernetics 25(2), 380–384 (1995)
3. Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In: International Conference on Machine Learning, pp. 148–156 (1996)
4. Gader, P.D., Mohamed, M.A., Keller, J.M.: Fusion of handwritten word classifiers. Pattern Recogn. Lett. 17(6), 577–584 (1996)
5. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(1), 66–75 (1994)
6. Jimenez, D.: Dynamically weighted ensemble neural networks for classification. In: Proceedings of the 1998 International Joint Conference on Neural Networks, IJCNN 1998, pp. 753–756 (1998)

7. Jimenez, D., Darm, T., Rogers, B., Walsh, N.: Locating anatomical landmarks for prosthetics design using ensemble neural networks. In: Proceedings of the 1997 International Conference on Neural Networks, IJCNN 1997 (1997)
8. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In: Tesauro, G., Touretzky, D., Leen, T. (eds.) Advances in Neural Information Processing Systems, vol. 7, pp. 231–238. The MIT Press, Cambridge (1995)
9. Kuncheva, L., Whitaker, C.J.: Using diversity with three variants of boosting: Aggressive. In: Roli, F., Kittler, J. (eds.) MCS 2002. LNCS, vol. 2364. Springer, Heidelberg (2002)
10. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998), `http://www.ics.uci.edu/$\sim$mlearn/MLRepository.html`
11. Oza, N.C.: Boosting with averaged weight vectors. In: Windeatt, T., Roli, F. (eds.) MCS 2003. LNCS, vol. 2709, pp. 973–978. Springer, Heidelberg (2003)
12. Torres-Sospedra, J., Hernndez-Espinosa, C., Fernández-Redondo, M.: Combining MF networks: A comparison among statistical methods and stacked generalization. In: Schwenker, F., Marinai, S. (eds.) ANNPR 2006. LNCS (LNAI), vol. 4087, pp. 210–220. Springer, Heidelberg (2006)
13. Torres-Sospedra, J., Hernndez-Espinosa, C., Fernández-Redondo, M.: Designing a multilayer feedforward ensembles with cross validated boosting algorithm. In: IJCNN 2006 proceedings, pp. 2257–2262 (2006)
14. Torres-Sospedra, J., Hernndez-Espinosa, C., Fernndez-Redondo, M.: Designing a multilayer feedforward ensemble with the weighted conservative boosting algorithm. In: IJCNN 2007 Proceedings, pp. 684–689. IEEE, Los Alamitos (2007)
15. Torres-Sospedra, J., Hernndez-Espinosa, C., Fernndez-Redondo, M.: Mixing aveboost and conserboost to improve boosting methods. In: IJCNN 2007 Proceedings, pp. 672–677. IEEE, Los Alamitos (2007)
16. Verikas, A., Lipnickas, A., Malmqvist, K., Bacauskiene, M., Gelzinis, A.: Soft combination of neural classifiers: A comparative study. Pattern Recognition Letters 20(4), 429–444 (1999)
17. Wanas, N.M., Kamel, M.S.: Decision fusion in neural network ensembles. In: Proceedings of the 2001 International Joint Conference on Neural Networks, IJCNN 2001, vol. 4, pp. 2952–2957 (2001)
18. Wolpert, D.H.: Stacked generalization. Neural Networks 5(6), 1289–1301 (1994)
19. Xu, L., Krzyzak, A., Suen, C.Y.: Methods of combining multiple classifiers and their applications to handwriting recognition. IEEE Transactions on Systems, Man, and Cybernetics 22(3), 418–435 (1992)
20. Zimmermann, H.-J., Zysno, P.: Decision and evaluations by hierarchical aggregation of information. Fuzzy Sets and Systems 10(3), 243–260 (1984)