Robust Alternating AdaBoost*

Héctor Allende-Cid[1], Rodrigo Salas[2], Héctor Allende[1], and Ricardo Ñanculef[1]

[1] Universidad Técnica Federico Santa María, Dept. de Informática, Casilla 110-V, Valparaíso-Chile
vector@inf.utfsm.cl, hallende@inf.utfsm.cl, jnancu@inf.utfsm.cl
[2] Universidad de Valparaíso, Departamento de Ingeniería Biomédica, Valparaíso-Chile
rodrigo.salas@uv.cl

Abstract. Ensemble methods are general techniques to improve the accuracy of any given learning algorithm. Boosting is a learning algorithm that builds the classifier ensembles incrementally. In this work we propose an improvement of the classical and inverse AdaBoost algorithms to deal with the problem of the presence of outliers in the data. We propose the Robust Alternating AdaBoost (RADA) algorithm that alternates between the classic and inverse AdaBoost to create a more stable algorithm. The RADA algorithm bounds the influence of the outliers to the empirical distribution, it detects and diminishes the empirical probability of "bad" samples, and it performs a more accurate classification under contaminated data.

We report the performance results using synthetic and real datasets, the latter obtained from a benchmark site.

Keywords: Machine ensembles, AdaBoost, Robust Learning Algorithms.

1 Introduction

Boosting algorithms, since the mid nineties, have been a very popular technique for constructing ensembles in the areas of Pattern Recognition and Machine Learning (see [2], [6], [8]). Boosting is a learning algorithm to construct a predictor by combining, what are called, weak hypotheses. The AdaBoost algorithm, introduced by Freund and Schapire [6], builds an ensemble incrementally, placing increasing weights on those examples in the data set, which appear to be "difficult". The Inverse AdaBoost [6] is a variant of the classic approach but with exactly the opposite philosophy, it decreases the weights of difficult objects.

In real engineering and scientific applications, data are noisy and present outlying observations. Assumptions of the underlying data generation process no longer holds and the model estimates are badly affected obtaining a poor performance. For example, outliers may occur in recording data.

* This work was supported in part by Research Grant Fondecyt 1061201 and 1070220, in part by the international cooperation Research Grant Fondecyt 7070262 and 7070093.

© Springer-Verlag Berlin Heidelberg 2007

In this work, we start by empirically demonstrating that the typical boosting algorithm, AdaBoost, is seriously affected by outliers. Similar results were reported in [1] and [2] for this algorithm and in [7] for bagging, another method to build ensembles [4]. Since AdaBoost forces each learner to focus on the difficult examples, learning rounds can strongly depend on isolated examples more than on local patterns of the featured space. Learning such outlying observations could seriously compromise the generalization ability of the final obtained learner. Inverse Boosting on the other hand, systematically ignores difficult examples and hence tends to be more robust than AdaBoost. However this is usually achieved at the expense of a lower performance.

Following this analysis, we propose an improvement of the classical and inverse AdaBoost algorithms to deal with the problem of the presence of outliers in the data. This consists in *Robust Alternating AdaBoost (RADA)*, an algorithm capable to alternate between the classic and inverse AdaBoost to create a more stable algorithm. The RADA algorithm bounds the influence of the outliers to the empirical distribution, detects and diminishes the empirical probability of "bad" samples, and will perform a more accurate classification under contaminated data.

This paper is organized as follows. In section 2, we briefly introduce AdaBoost and Inverse AdaBoost algorithms. In section 3, we make an analysis of the sensitivity of AdaBoost to the presence of outliers. Our proposed algorithm is illustrated in section 4. In section 5 we compare the performance of the classic AdaBoost, inverse AdaBoost and our proposed algorithm called RADA. The last section is devoted to concluding remarks.

2 Boosting Algorithms

The AdaBoost Algorithm [6], introduced in 1995 by Freund and Schapire, has a theoretical background based on the "PAC" learning model [9]. The authors of this model were the first to pose the question of whether a weak learning algorithm that is slightly better than random guessing can be "boosted" in a strong learning algorithm. The classic AdaBoost takes as an input a training set $\mathcal{Z} = \{(x_1, y_1)...(x_n, y_n)\}$ where each x_i is a variable that belongs to $\mathcal{X} \subset \mathbb{R}^d$ and each label y_i is in some label $\mathcal{Y}$. In this particular paper we assume that $\mathcal{Y} = \{-1, 1\}$. AdaBoost calls a weak or base learning algorithm repeatedly in a sequence of stages $t = 1...T$. The main idea of AdaBoost is to maintain a sampling distribution over the training set. This sample set is used to train the learner at round t. Let $\mathcal{D}_t(i)$ be the sampling weight assigned to the example i on round t. At the beginning of the algorithm the distribution is uniform, that is distribution $\mathcal{D}_1(i) = \frac{1}{n}$ for all i. At each round of the algorithm however, the weights of the incorrectly classified examples are increased, so that the following weak learner is forced to focus on the "hard" examples of the training set. The job of each weak or base learner is to find a hypothesis $h_t : \mathcal{X} \rightarrow \{-1, 1\}$

appropriate for the distribution $\mathcal{D}_t$. The goodness of the obtained hypothesis can be quantified as the weighted error:

$$\epsilon_t = Pr_{i \sim \mathcal{D}_t}[h_t(x_i) \neq y_i] = \sum_{i:h_t(x_i) \neq y_i} \mathcal{D}_t(i) \tag{1}$$

Notice that the error is measured with respect to the distribution $\mathcal{D}_t$ on which the weak learner was trained. Once AdaBoost has computed a weak hypothesis h_t, it measures the importance that the algorithm assigns to h_t with the parameter.

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right) \tag{2}$$

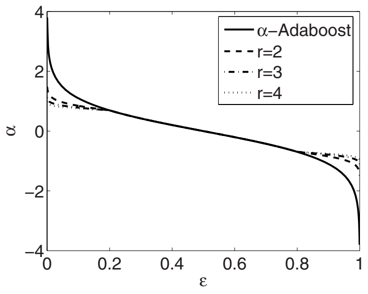

Fig. 1. Graph of the α variable given by equation (4) with the robust parameter values $r = 2, 3, 4$, compared to the AdaBoost α

Figure 1 shows the graph of α_t. After choosing α_t the next step is to update the distribution $\mathcal{D}_t$ with the following rule,

$$\mathcal{D}_{t+1}(i) = \frac{1}{Z_t} \mathcal{D}_t(i) e^{-\alpha_t y_i h_t(x_i)}$$

where Z_t is a normalization factor. The effect of this rule is that, when a example is misclassified its sampling weight for the next round is increased, and the opposite occurs when the classification is correct. This updating rule makes the algorithm to focus on the "hard" examples, instead in the correctly classified examples. After a sequence of T rounds have been carried out, the final hypothesis H is computed as

$$H_T(x) = sign \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right) \tag{3}$$

The other Boosting variant considered in this paper is the Inverse AdaBoost. This algorithm is more robust than the classic approach because it focuses on the correct classified examples, and tries to diminish the influence of the incorrect examples in the distribution, however the classifiers will tend to be more and more similar, eliminating any diversity in the process [8]. Inverse Boosting is a variant similar to the "hedge" algorithm, first proposed in [6]. The philosophy is completely the opposite to classic boosting. Instead of increasing the sampling probability of the hard examples, we decrease it, thereby gradually filtering them out. Each learner is created to reinforce the knowledge acquired by previous learners, and hence the capability of the algorithm to discover new patterns is low.

3 Empirical Robustness Analysis of AdaBoost

In this section we empirically show that the classic AdaBoost algorithm lacks of robustness and stability under the presence of outlying data.

Suppose that the weak learner h_{t*} at stage t^* correctly learns almost all its training samples $\mathcal{Z}_{t^*}$ obtaining a very low classification error (1) ($\epsilon_{t^*} \approx 0$), in this case the α_{t^*} parameter (2) will approach to infinity. This mean that the weak learner h_{t^*} will receive a very high weight in the strong hypothesis $H_T(x)$ and the ensemble decision will strongly depended on this weak learner. If there exists a sample in $\mathcal{Z}$, say (x_j, y_j), that was misclassified by the weak learner h_{t^*}, then the expression $\exp(-\alpha_{t^*} y_j h_{t^*}(x_j))$ will tend to infinity as long as α_{t^*} tends to infinity. Then, the sample (x_j, y_j), will obtain a very high weight $0 << \mathcal{D}_{t+1}(j) \approx 1$ for the next round. Due to the high probability of the data (x_j, y_j), it will be sampled several times during the bootstrap step and the weak hypothesis will learn it. If this sample is a "bad" sample (for example an error) or an outlier, an undesired effect of overfitting, poor generalization, very complex model and learning bad data will occur.

If we even construct a robust weak learner (see [1]) it will also be affected by a "bad sample" where most of the data in the bootstrap sample behaves different than the majority of the original sample.

To empirically show the behavior of AdaBoost we create the following synthetic case. We randomly generated 41 samples, where the 20 samples of the class -1 were generated with a gaussian distribution $\mathcal{N}(-2, 1)$, 20 samples with of the class 1 were generated with a gaussian distribution $\mathcal{N}(2, 1)$ and we arbitrarily introduce an additional "bad" sample at location $x = 5$ with target $y = -1$. We bootstrapped only the original 40 samples and construct a weak learner h_1, where

$$h_1(x) = \begin{cases} 1 & |\overline{X}^{[1]} - x| < |\overline{X}^{[-1]} - x| \\ -1 & |\overline{X}^{[1]} - x| \geq |\overline{X}^{[-1]} - x| \end{cases}$$

where $\overline{X}^{[k]} = \frac{1}{n}\sum_{i=1}^{n} x_i^{[k]}$, $k = \{-1, 1\}$, and $x_i^{[k]}$ is the bootstrapped sample i of the class k. Figure 2 shows the empirical distribution at stage 1, 2 and 3 (from left to right), where the first row is the empirical distribution of the AdaBoost algorithm, while the second row is the Robust Alternating AdaBoost algorithm.

Note that in the first round all the samples weight are equal to $\frac{1}{41}$, but in the second round, the misclassified sample number 41 considerably increased its weight over 0.5 in the AdaBoost algorithm while our proposal remains lower. Due to the high weight of this sample, in the next round, the bootstrap sample is composed mostly by this data. The weak learner of stage two learns the outlying sample but at the same time degrades the performance of the strong hypothesis misclassifying other three data that were correctly classify previously. This phenomenon does not occur to our proposal, remaining more stable.

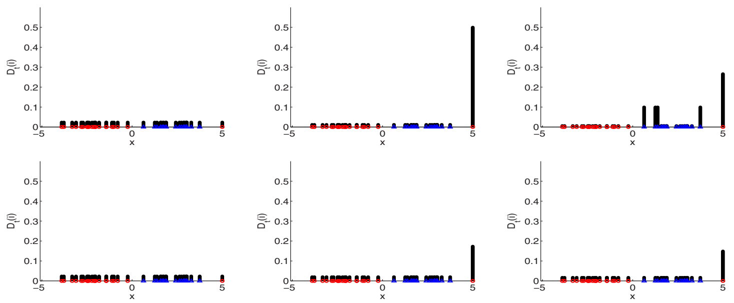

Fig. 2. Empirical Distribution of the AdaBoost **(first row)** and the Robust Alternating AdaBoost **(second row)** algorithms at stages 1 **(left)**, 2 **(middle)** and 3 **(right)**

4 Robust Alternating AdaBoost

In this section we improve the classical and inverse AdaBoost algorithms to deal with the problem of the presence of outliers in the data. Our proposal mainly consists in creating a variant of the AdaBoost algorithm that will alternate between the classic and inverse AdaBoost to create a more stable algorithm that will perform a more adequate classification.

In the previous section we empirically showed that the empirical distribution $D_t(i)$ of the data sample was strongly affected by the presence of outliers. The actualization of the empirical probability $\mathcal{D}_{t+1}(i)$ of the sample (x_i, y_i) depends on its weight of the previous stage $\mathcal{D}_t(i)$, the result of the classification $h_t(x_i)$ compared to the target y_i and the variable α_t. If the weak learner has a very low error ϵ_t, then the α_t variable becomes very large as it is shown in figure 1. Furthermore if the weak learner misclassifies the sample (x_i, y_i), then the new probability weight $\mathcal{D}_{t+1}(i)$ will considerably increase its value. This could mean that if the sample corresponds to an outlying observation that was misclassified by a good learner, it will receive a high probability distribution and will be more probable to be selected more than once, and the next weak learner h_{t+1} will mostly learn this isolated observation. The resulting model could overfit the training data, it could be more complex with several weak learners composing the strong hypothesis, or worst, it could learn erroneous data samples.

To overcome this drawback of the classic AdaBoost, we bound the value of the α_t variable by taking the r-th square root to low values of ϵ_t, i.e., we compute α_t as:

$$\alpha_t = \begin{cases} \frac{1}{2}\sqrt[r]{\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)} + \alpha_\gamma & \epsilon_t < \gamma \\ \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) & \epsilon_t \geq \gamma \end{cases} \tag{4}$$

where $\alpha_\gamma = \frac{1}{2}\ln\left(\frac{1-\gamma}{\gamma}\right) - \frac{1}{2}\sqrt[r]{\ln\left(\frac{1-\gamma}{\gamma}\right)}$ is a constant needed so that equation (4) is continuous.

Applying equation (4) will prevent that the empirical distribution considerably grow in one step for any sample. However the empirical distribution is updated at each stage, and after few iterations the probability weight of the samples that were repeatedly misclassified will have bigger values compared to other samples. For this reason we introduce an inverting variable $\beta(i)$ and an age variable $age(i)$ for each sample $i = 1..n$. The variable $\beta(i)$ has an initial value of one, meaning that the algorithm behaves for the sample as the classical AdaBoost, i.e., increases its empirical distribution when it is misclassified and decreases its value otherwise. If the value of $\beta(i)$ is -1, then the algorithm behaves for the sample as the inverse AdaBoost, i.e., decreases its empirical distribution when it is misclassified and increases its value otherwise. The variable $age(i)$ counts the number of times that the sample i was sequentially misclassified, if this number exceeds the threshold τ then the value of $\beta(i)$ is inverted to -1. In other words, the weight of the misclassified sample grows until the number of iterations reaches a limit τ and then begins to decrease. If the $\beta(i)$ was inverted to -1 and the sample was correctly classified in further stages, then the value of $\beta(i)$ is inverted back to 1.

Algorithm 1 shows our proposed *Robust Alternating AdaBoost* algorithm.

5 Experimental Results

In this section we empirically show the performance of our *Robust Alternating AdaBoost* **(RADA)** model proposal compared to classical Adaboost **(ADA)** and Inverse Adaboost **(IADA)** models (see [6] and [8] respectively for further details about these models) in both Synthetic and Real data sets, the latter was obtained from a benchmark site.

The data of both, synthetic and real data sets, were separated in training and test sets. All the results reported were obtained for each model as the mean value of the metrics computed for 20 runs with the same sets of data.

For the synthetic experiment we have created a synthetic data set $\{(x_i, y_i)\}_{i=1}^n$, as an independent sample obtained from a mixture of gaussian distributions labeled with the class $\{-1, 1\}$. For the class $y = -1$, the random variable $X^{[-1]}$ is a mixture of three gaussian distributions with mean $\mu_1^{[-1]} = -10$, $\mu_2^{[-1]} = 0$ and $\mu_3^{[-1]} = 10$ and standard deviation $\sigma_i^{[-1]} = \sigma = 1, i = 1..3$ respectively. For the class $y = 1$, the random variable $X^{[1]}$ is a mixture of two gaussian distributions

Algorithm 1. Robust Alternating AdaBoost

```
1: Given is the training data set $\mathcal{Z} = \{(x_1, y_1), ..., (x_n, y_n)\}$ with $n$ elements, where
   $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{-1, 1\}$.
2: Initialize the parameters. Pick the age threshold $\tau$, the bound threshold $\lambda$ and the
   robust parameter $r$. Let $T = 0$.
3: Initialize the empirical distribution $\mathcal{D}_1(i) = \frac{1}{n}$, the inverting variable $\beta(i) = 1$ and
   the age variable $age(i) = 0$ for each data sample $(x_i, y_i), i = 1..n$.
4: repeat
5:    Increment $T$ by one.
6:    Take a bootstrap sample $\mathcal{Z}_T$ from $\mathcal{Z}$ with distribution $\mathcal{D}_T$.
7:    Train the weak learner $h_T : \mathcal{X} \to \{-1, 1\}$ with the bootstraped sample $\mathcal{Z}_T$ as
      the training set.
8:    Compute the weighted error $\epsilon_T$ of the weak hypothesis $h_T$ as
          $$\epsilon_T = Pr_{i \sim \mathcal{D}_T}[h_T(x_i) \neq y_i]$$
9:    Compute $\alpha_T$ with equation
          $$\alpha_T = \begin{cases} \frac{1}{2}\sqrt[r]{\ln\left(\frac{1-\epsilon_T}{\epsilon_T}\right)} + \alpha_\gamma & \epsilon_T < \gamma \\ \frac{1}{2}\ln\left(\frac{1-\epsilon_T}{\epsilon_T}\right) & \epsilon_T \geq \gamma \end{cases}$$
10:   Update the empirical distribution as
          $$\mathcal{D}_{T+1}(i) = \frac{\mathcal{D}_T(i)}{Z_T} \times e^{(-\alpha_T \beta(i) y_i h_T(x_i))}$$
      where $Z_T$ is the normalization factor.
11:   The strong hypothesis $H_T(x)$ at stage $T$ is given by
          $$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$$
12:   Classify the training data set $\mathcal{Z} = \{(x_1, y_1), ..., (x_n, y_n)\}$ with the strong hypoth-
      esis $H_T(x)$.
13:   if sample $(x_i, y_i)$ is correctly classified by $H_T(x)$ (i.e, $H_T(x_i)y_i > 0$) then
14:      Let $age(i) = 0$ and $\beta(i) = 1$.
15:   else
16:      Increment $age(i)$ by one.
17:      if $age(i) > \tau$ then let $\beta(i) = -1$ and $age(i) = 0$
18:   end if
19: until The stopping criterion is met
20: Output: The strong hypothesis $H_T(x)$
```

with mean $\mu_1^{[1]} = -5$, $\mu_2^{[1]} = 5$ and standard deviation $\sigma_i^{[1]} = \sigma = 1$, $i = 1, 2$ respectively. For each gaussian we generated 20 examples, i.e., the class $y = -1$ has 60 samples while the class $y = 1$ has 40 data. The observational process was obtained by including additive outliers: $Z^{[k]} = X^{[k]} + V\,U$, where $X^{[k]}, k = -1, 1$, are the gaussian variables generated as was previously explained, V is a zero-one process with $P(V \neq 0) = \delta$, $0 < \delta \ll 1$, and U has distribution $\mathcal{N}(0, \sigma_U^2)$

Table 1. Summary results of the performance evaluation of the ADA, IADA and RADA algorithms

Algorithm	% Outliers	T	$CE - Train$	$Min - Train$	$CE - Test$	$Min - Test$
ADA	0	9.8	21.45 ± 7.82	**3.00**	23.60 ± 8.60	**8.00**
IADA	0	8.7	**20.95 ± 1.10**	20.00	23.25 ± **1.02**	22.00
RADA5	0	7.3	24.50 ± 6.51	9.00	24.30 ± 7.58	10.00
RADA10	0	8.7	22.85 ± 6.67	9.00	**22.10** ± 6.80	11.00
RADA20	0	**7.1**	23.00 ± 7.44	7.00	24.55 ± 7.79	10.00
ADA	5	18.8	6.15 ± 2.70	3.00	9.20 ± 3.49	5.00
IADA	5	**14.9**	12.90 ± 7.74	3.00	15.05 ± 7.96	**4.00**
RADA5	5	18.6	6.55 ± 4.81	**2.00**	9.10 ± 4.87	5.00
RADA10	5	17.7	**5.35 ± 2.43**	**2.00**	**8.00 ± 2.27**	5.00
RADA20	5	17.2	5.90 ± 2.59	**2.00**	9.60 ± 3.30	5.00
ADA	10	17.4	**10.70 ± 2.34**	8.00	6.85 ± 4.57	**1.00**
IADA	10	**10.0**	19.80 ± 7.78	7.00	16.65 ± 10.66	2.00
RADA5	10	17.5	11.10 ± 3.65	8.00	6.10 ± 4.40	**1.00**
RADA10	10	18.2	11.80 ± 5.23	**5.00**	6.00 ± 6.64	**1.00**
RADA20	10	16.8	11.25 ± 2.67	8.00	**5.45 ± 3.71**	**1.00**
ADA	20	13.5	20.50 ± 8.52	**13.00**	17.25 ± 8.48	8.00
IADA	20	**12.7**	21.00 ± 7.17	14.00	17.55 ± 7.23	9.00
RADA5	20	13.5	**18.55** ± 6.90	14.00	16.55 ± 8.46	**7.00**
RADA10	20	16.3	18.65 ± 5.72	**13.00**	**15.25** ± 6.95	9.00
RADA20	20	15.6	18.75 ± **5.34**	15.00	16.30 ± **6.42**	9.00

with variance $\sigma_U^2 \gg \sigma^2$. The generating process was affected with $\delta = 0\%$, 5%, 10% and 20% of outliers and $\sigma_U = 9 * \sigma$. The test set was generated similar to the training set. The classifier used in the algorithms is the Bayesian Classifier (QDA) (see [5]).

Table 1 shows the summary results of the performance evaluation of the ADA, IADA and RADA algorithms, where for the latter the robustifying r parameter has the values $r = 0.05, 0.10$ and 0.20. As we can observe in the *CE-Test* column, the algorithm with better generalization ability was the RADA algorithm with its variants outperforming in most of the experiments the ADA and IADA algorithms with statistical significance. Nevertheless that the ADA obtained good results in the training set, the RADA algorithm with its variants obtained lower error in the test set. Note that in the contaminated cases one of the RADA variants always obtained the best strong hypothesis (see column *Min-Test*). We also observed that when the percentage of outliers was increased all the models became more complex and the performance was degraded. When the percentage of outliers was 0%, the algorithm with best performance and lowest standard deviation was IADA, however in the contaminated data sets, it obtained the worst performance. We can also notice, that RADA10 and RADA20 obtained better results than RADA5, implying that the robust r parameter influences in the performance of the RADA algorithm under contaminated data. The IADA algorithm obtained the simplest model because its mean T value was lower than the other algorithms, this implies that the minimum value was reached with less rounds.

Table 2. Summary results of the performance evaluation of the ADA, IADA and RADA algorithms

Dataset	Algorithm	T	$CE - Train$	$Min - Train$	$CE - Test$	$Min - Test$
Breast Cancer	ADA	6.2	**1.64** ± 0.82	**0.24**	3.33 ± 0.29	2.93
	IADA	8.9	2.52 ± **0.33**	1.96	3.24 ± **0.27**	2.93
	RADA5	**4.2**	1.86 ± 0.77	0.49	3.41 ± 0.41	**2.56**
	RADA10	8.3	2.04 ± 0.53	0.98	3.30 ± **0.27**	**2.56**
	RADA20	7.3	1.93 ± 0.51	0.98	**3.13** ± 0.37	**2.56**
Diabetes	ADA	24.7	22.33 ± 1.04	21.30	23.50 ± 1.27	**20.52**
	IADA	**9.7**	22.47 ± **0.68**	21.96	23.49 ± **0.69**	21.82
	RADA5	22.1	22.23 ± 0.81	21.09	23.40 ± 1.24	21.82
	RADA10	31.9	**21.87** ± 0.81	**20.87**	**22.80** ± 1.08	21.50
	RADA20	30.0	22.58 ± 0.75	21.74	23.14 ± 0.90	22.15
Liver Diseases	ADA	24.4	**29.18** ± 2.25	25.60	40.14 ± 3.51	38.41
	IADA	31.5	35.77 ± 3.06	30.44	46.96 ± 3.88	39.86
	RADA5	24.6	29.95 ± 2.97	26.57	39.57 ± **3.15**	35.51
	RADA10	**21.1**	29.78 ± 2.45	**25.12**	**39.02** ± 3.84	**31.88**
	RADA20	23.1	30.80 ± 2.47	**25.12**	39.86 ± 3.67	34.78

In the real experiment we test the algorithms with three real data sets obtained from the UCI Machine Learning repository [3]. The *Diabetes*, *Liver Diseases* and *Breast-Cancer* data sets were selected. Table 2 shows the summary results of the performance evaluation of the ADA, IADA and RADA algorithms, where RADA has the same robustifying parameters as in the Synthetic Experiments.

The results are shown in table 2. As we can observe in the column *CE-Test*, RADA obtained a better generalization ability than ADA and IADA. Our algorithm outperforms the ADA and IADA algorithms with statistical significance. Furthermore we notice that the algorithms with better results in the 3 different real data sets were RADA10 and RADA20. RADA5 has only a better performance than ADA and IADA in the Diabetes and Liver diseases data sets. In the *CE-Train* column we observe that ADA has better performance in Liver Diseases and Breast-Cancer. In the testing phase, the RADA algorithm and its variants outperform the ADA algorithm. Another drawback of ADA is its standard deviation in the training phase is high compared with RADA.

6 Concluding Remarks

In this paper we have introduced the *Robust Alternating Adaboost* algorithm. This algorithm has the capability to alternate between two differents views of approaching classification problems using a midpoint approach between robustness and sensitivity.

The performance of our algorithm shows better results in the simulation study in both the synthetic and real data sets. In the real case, we investigated three benchmark data known as Breast-Cancer, Diabetes and Liver Diseases. The comparative study with the classic AdaBoost and Inverse AdaBoost without improvement shows that our Robust Alternating AdaBoost outperforms the alternative models with statistical significance, obtaining good results in both synthetic and real data. Further studies are needed in order to analyze the convergence, and to test this algorithm with other weak learners.

References

1. Allende, H., Nanculef, R., Salas, R.: Bootstrapping neural networks. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 813–822. Springer, Heidelberg (2004)
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine Learning 36(1-2), 105–139 (1999)
3. Blake, C.L., Merz, C.J.: UCI repository of machine learning databases (1998)
4. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
5. Duda, R., Hart, P., Stork, D.: Pattern classification. Wiley-Interscience, Chichester (2000)
6. Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
7. Kanamori, T., Takenouchi, T., Eguchi, S., Murata, N.: The most robust loss function for boosting. In: Pal, N.R., Kasabov, N., Mudi, R.K., Pal, S., Parui, S.K. (eds.) ICONIP 2004. LNCS, vol. 3316, pp. 496–501. Springer, Heidelberg (2004)
8. Kuncheva, L., Whitaker, C.: Using diversity with three variants of boosting: Aggressive, conservative and inverse. In: Roli, F., Kittler, J. (eds.) MCS 2002. LNCS, vol. 2364, pp. 81–90. Springer, Heidelberg (2002)
9. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)