Combining Prototype Selection with Local Boosting

Christos K. Aridas(✉), Sotiris B. Kotsiantis, and Michael N. Vrahatis

Computational Intelligence Laboratory (CILab), Department of Mathematics, University of Patras, 26110 Patras, Greece
char@upatras.gr, {sotos,vrahatis}@math.upatras.gr

Abstract. Real life classification problems require an investigation of relationships between features in heterogeneous data sets, where different predictive models can be more proper for different regions of the data set. A solution to this problem is the application of the local boosting of weak classifiers ensemble method. A main drawback of this approach is the time that is required at the prediction of an unseen instance as well as the decrease of the classification accuracy in the presence of noise in the local regions. In this research work, an improved version of the local boosting of weak classifiers, which incorporates prototype selection, is presented. Experimental results on several benchmark real-world data sets show that the proposed method significantly outperforms the local boosting of weak classifiers in terms of predictive accuracy and the time that is needed to build a local model and classify a test instance.

Keywords: Local boosting · Weak learning · Prototype selection · Pattern classification

1 Introduction

In machine learning, instance-based (or memory-based) learners classify an unseen object by comparing it to a database of pre-classified objects. The fundamental assumption is that similar instances will share similar class labels.

Machine learning models' assumptions would not necessarily hold globally. Local learning [1] methods come to solve this problem. The latter allow to extend learning algorithms, that are designed for simple models, to the case of complex data, for which the models' assumptions are valid only locally. The most common case is the assumption of linear separability, which is usually not fulfilled globally in classification problems. Despite this, any supervised learning algorithm that is able to find only a linear separation, can be used inside a local learning process, producing a model that is able to model complex non-linear class boundaries.

A technique of boosting local weak classifiers, that is based on a reduced training set after the usage of prototype selection [11], is proposed. It is common that boosting algorithms are well-known to be susceptible to noise [2]. In the case

L. Iliadis and I. Maglogiannis (Eds.): AIAI 2016, IFIP AICT 475, pp. 94–105, 2016.
DOI: 10.1007/978-3-319-44944-9_9

of local boosting, the algorithm should manage reasonable noise and be at least as good as boosting, if not better. For the experiments, we used two variants of Decision Trees [21] as weak learning models: one-level Decision Trees, which are known as Decision Stumps [12] and two-level Decision Trees. An extensive comparison over several data sets was performed and the results show that the proposed method outperforms simple and local boosting in terms of classification accuracy.

In the next Section, specifically in Subsect. 2.1, the localized experts are discussed, while boosting approaches are described in Subsect. 2.2. In Sect. 3 the proposed method is presented. Furthermore, in Sect. 4 the results of the experiments on several UCI data sets, after being compared with standard boosting and local boosting, are portrayed and discussed. Finally, Sect. 5 concludes the paper and suggests further directions in current research.

2 Background Material

For completeness purposes, local weighted learning, prototype selection methods as well as boosting classifier techniques are briefly described in the following subsections.

2.1 Local Weighted Learning and Prototype Selection

Supervised learning algorithms are considered global if they use all available training sets, in order to build a single predictive model, that will be applied in any unseen test instance. On the other hand, a method is considered local if only the nearest training instances around the testing instance contribute to the class probabilities.

When the size of the training data set is small in contrast to the complexity of the classifier, the predictive model frequently overfits the noise in the training data. Therefore, the successful control of the complexity of a classifier has a high impact in accomplishing good generalization. Several theoretical and experimental results [23] indicate that a local learning algorithm provides a reasonable solution to this problem.

In local learning [1], each local model is built completely independent of all other models in a way that the total number of local models in the learning method indirectly influences how complex a function can be estimated - complexity can only be controlled by the level of adaptability of each local model. This feature prevents overfitting if a strong learning pattern exists for training each local model.

Prototype selection is a technique that aims to decrease the training size without surfacing the prediction performance of a memory based learner [18]. Besides this, by reducing the training set size it might decrease the computational cost that will be applied in the prediction phase.

Prototype selection techniques can be grouped in three categories: preservation techniques, which aim to find a consistent subset from the training data set, ignoring the presence of noise, noise removal techniques, which aim to remove noise, and hybrid techniques, which perform both objectives concurrently [22].

2.2 Boosting Classifiers

Experimental research works have proven that ensemble methods usually perform better, in terms of classification accuracy, than the individual base classifier [2], and lately, several theoretical explanations have been advised to explain the success of some commonly used ensemble methods [13]. In this work, a local boosting technique that is based on a reduced training set, after the usage of prototype selection [11], is proposed and for this reason this section introduces the boosting approach.

Boosting constructs the ensemble of classifiers by subsequently tweaking the distribution of the training set based on the accuracy of the previously created classifiers. There are several boosting variants. These methods assign a weight to each training instance. Firstly, all instances are equally weighted. In each iteration a new classification model, named base classifier, is generated using the base learning algorithm. The creation of the base classifier has to consider the weight distribution. Then, the weight of each instance is adjusted, depending on the accuracy of the prediction of the base classifier for that instance. Thus, Boosting attempts to construct new classification models that are able to better classify the "hard" instances for the previous ensemble members. The final classification is obtained from a weighted vote of the base classifiers. AdaBoost [8] is the most well-known boosting method and the one that is used over the experimental analysis that is presented in Sect. 3.

Adaboost is able to use weights in two ways to generate a new training data set to provide to the base classifier. In boosting by sampling, the training instances are sampled with replacement with probability relative to their weights. In [26] the authors showed empirically that a local boosting-by-resampling technique is more robust to noise than the standard AdaBoost. The authors of [17] proposed a Boosted k-NN algorithm that creates an ensemble of models with locally modified distance weighting that has increased generalization accuracy and never performs worse than standard k-NN. In [10] the authors presented a novel method for instance selection based on boosting instance selection algorithms in the same way boosting is applied to classification.

3 The Proposed Algorithm

Two main disadvantages of simple local boosting are: (i) When the amount of noise is large, simple local boosting does not have the same performance [26] as Bagging [3] and Random Forest [4]. (ii) Saving the data for each pattern increases storage complexity. This might restrict the usage of this method to limited training sets [21]. The proposed algorithm incorporates prototype selection to handle, among others, the two previous problems. In the learning phase, a prototype selection [11] method based on the Edited Nearest Neighbor (ENN)

[24] technique reduces the training set by removing the training instances that do not agree with the majority of the k nearest neighbors. In the application phase, it constructs a model for each test instance to be estimated, considering only a subset of the training instances. This subset is selected according to the distance between the testing sample and the available training samples. For each testing instance, a boosting ensemble of a weak learner is built using only the training instances that are lying close to the current testing instance. The prototype selection aims to improve the classification accuracy as well as the time that is needed to build a model for each test instance at the prediction.

The proposed ensemble method has some free parameters, such as the number of neighbors (k_1) to be considered when the prototype selection is executed, the number of neighbors (k_2) to be selected in order to build the local model, the distance metric and the weak learner. In the experiments, the most well - known Euclidean similarity function was used as a distance metric.

In general, the distance between points **x** and **y** in a Euclidean space R^n is given by (1).

$$d(x, y) = \|x - y\|_2 = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}. \tag{1}$$

The most common value for the nearest neighbor rule is 5. Thus, the k_1 was set to 5 and $k_2 = 50$. since at about this size of instances, it is appropriate for a simple algorithm to build a precise model [14]. The proposed method is presented in Algorithm 1.

Algorithm 1. PSLB(k_1, k_2, *distanceMetric*, *weakLearner*)

```
procedure TRAINING($k_1$, $distanceMetric$)
    for each training instance do
        Find the $k_1$ nearest neighbors using the selected $distanceMetric$
        if instance does not agree with the majority of the $k_1$ then
            Remove this instance from the training set
        end if
    end for
end procedure
procedure CLASSIFICATION($k_2$, $distanceMetric$, $weakLearner$)
    for each testing instance do
        Find the $k_2$ nearest neighbors using the selected $distanceMetric$
        Apply boosting to the base $weakLearner$ using the $k_2$ nearest neighbors
        The answer of the boosting ensemble is the prediction for the testing instance
    end for
end procedure
```

4 Numerical Experiments

In order to evaluate the performance of the proposed method, an initial version was implemented[1] and a number of experiments were conducted using several data sets from different domains. From the UCI repository [16] several data sets were chosen. Discrete features transformed to numeric by using a simple quantization. Each feature is scaled to have zero mean and standard deviation one. Also all missing values were treated as zero. In Table 1 the name, the number of patterns, the attributes, as well as the number of different classes for each data set are shown.

Table 1. Benchmark data sets used in the experiments

Data set	#patterns	#attribues	#classes
cardiotocography	2126	21	10
cylinder-bands	512	25	2
dermatology	366	24	6
ecoli	336	7	8
energy-y1	768	8	3
glass	214	9	6
low-res-spect	531	100	9
magic	19020	10	2
musk-1	476	166	2
ozone	2536	72	2
page-blocks	5473	10	5
pima	768	8	2
synthetic-control	600	60	6
tic-tac-toe	958	9	2

All experiments were run on an Intel Core i3–3217U machine at 1.8 GHz, with 8 GB of RAM, running Linux Mint 17.3 64bit using Python and the scikit-learn [19] library.

For the experiments, we used two variants of Decision Trees [25] as weak learners. One-level Decision Trees [12], also known as Decision Stumps, and two-level Decision Trees [20]. We used the Gini Impurity [5] as criterion to measure the quality of the splits in both algorithms. The boosting process for all classifiers performed using the AdaBoost algorithm with 25 iterations in each model. In order to calculate the classifiers accuracy, the whole data set was divided into five mutually exclusive folds and for each fold the classifier was trained on the union of all of the other folds. Then, cross-validation was run five times for each algorithm and the mean value of the five folds was calculated.

[1] https://bitbucket.org/chkoar/pslb.

4.1 Prototype Selection

The prototype selection process is independent of the base classifier and it takes place once in the training phase of the proposed algorithm. It depends only on the k_1 parameter. The number of neighbors to be considered when the prototype selection is executed. In Table 2 the average of training patterns, the average of the removed patterns as well as the average reduction of each data set is presented. The average refers to the average of all training folds during the 5-fold cross-validation.

Table 2. Average reduction

Data set	#avg training patterns	#avg removed patterns	%avg reduction
cardiotocography	1701	268	15,73
cylinder-bands	410	60	14,60
dermatology	293	7	02,53
ecoli	269	29	10,86
energy-y1	614	17	02,77
glass	171	40	23,13
lowresspect	425	47	11,11
magic	15216	1798	11,81
musk-1	381	24	06,25
ozone	2029	50	02,48
page-blocks	4378	111	02,53
pima	614	109	17,77
synthetic-control	480	8	01,67
tic-tac-toe	766	1	00,13

4.2 Using Decision Stump as Base Classifier

In the first part of the experiments, Decision Stumps [12] were used as weak learning classifiers. Decision Stumps (DS) are one-level Decision Trees that classify instances based on the value of just a single input attribute. Each node in a decision stump represents a feature in an instance to be classified and each branch represents a value that the node can take. Instances are classified starting at the root node and are sorted based on their attribute values. In the worst case, a Decision Stump will behave as a base line classifier and will possibly perform better, if the selected attribute is particularly informative.

The proposed method, denoted as PSLBDS, is compared with the Boosting Decision Stumps, denoted as BDS and the Local Boosting of Decision Stumps, denoted as LBDS. Since the proposed method uses fifty neighbors, a 50-Nearest

Neighbors (50NN) classifier has included in the comparisons. In Table 3 the average accuracy of the compared methods is presented. Table 3 indicates that the hypotheses generated by PSLBDS are apparently better since the PSLBDS algorithm has the best mean accuracy score in nearly all cases.

Table 3. Average accuracy of the compared algorithms using an one-level decision tree as base classifier

Data set	PSLBDS	BDS	LBDS	50NN
cardiotocography	**0.682** $\pm$ 0.028	0.548 $\pm$ 0.088	0.659 $\pm$ 0.015	0.607 $\pm$ 0.029
cylinder-bands	**0.613** $\pm$ 0.030	0.560 $\pm$ 0.037	0.584 $\pm$ 0.014	0.582 $\pm$ 0.017
dermatology	**0.942** $\pm$ 0.027	0.641 $\pm$ 0.142	0.940 $\pm$ 0.022	0.902 $\pm$ 0.020
ecoli	**0.821** $\pm$ 0.029	0.622 $\pm$ 0.129	0.794 $\pm$ 0.026	0.780 $\pm$ 0.050
energy-y1	**0.844** $\pm$ 0.090	0.706 $\pm$ 0.050	0.836 $\pm$ 0.092	0.822 $\pm$ 0.091
glass	**0.582** $\pm$ 0.085	0.285 $\pm$ 0.094	0.568 $\pm$ 0.065	0.446 $\pm$ 0.169
low-res-spect	0.850 $\pm$ 0.025	0.584 $\pm$ 0.069	0.846 $\pm$ 0.012	**0.851** $\pm$ 0.023
magic	**0.849** $\pm$ 0.005	0.828 $\pm$ 0.005	0.834 $\pm$ 0.005	0.828 $\pm$ 0.004
musk-1	**0.727** $\pm$ 0.096	**0.727** $\pm$ 0.052	0.718 $\pm$ 0.085	0.618 $\pm$ 0.096
ozone	0.966 $\pm$ 0.008	0.960 $\pm$ 0.010	0.887 $\pm$ 0.133	**0.971** $\pm$ 0.001
page-blocks	**0.954** $\pm$ 0.012	0.853 $\pm$ 0.163	0.950 $\pm$ 0.013	0.942 $\pm$ 0.007
pima	**0.757** $\pm$ 0.028	0.755 $\pm$ 0.024	0.685 $\pm$ 0.024	0.749 $\pm$ 0.017
synthetic-control	**0.947** $\pm$ 0.011	0.472 $\pm$ 0.074	0.943 $\pm$ 0.020	0.887 $\pm$ 0.030
tic-tac-toe	**0.884** $\pm$ 0.084	0.733 $\pm$ 0.034	0.882 $\pm$ 0.083	0.747 $\pm$ 0.101

Demšar [6] suggests that the non-parametric tests should be preferred over the parametric in the context of machine learning problems, since they do not assume normal distributions or homogeneity of variance. Therefore, in the direction of validating the significance of the results, the Friedman test [9], which is a rank-based non-parametric test for comparing several machine learning algorithms on multiple data sets, was used, having as a control method the PSLBDS algorithm. The null hypothesis of the test states that all the methods perform equivalently and thus their ranks should be equivalent. The average rankings, according to the Friedman test, are presented in Table 4.

Assuming a significance level of 0.05 in Table 4, the p-value of the Friedman test indicates that the null hypothesis has to be rejected. So, there is at least one method that performs statistically different from the proposed method. With the intention of investigating the aforementioned, Finner's [7] and Li's [15] post hoc procedures were used.

In Table 5 the p-values obtained by applying post hoc procedures over the results of the Friedman statistical test are presented. Finner's and Li's procedure rejects those hypotheses that have a p-value ≤ 0.05. That said, the adjusted p-values obtained through the application of the post hoc procedures are presented in Table 6. Hence, both post hoc procedures agree that the PSLBDS algorithm performs significantly better than the BDS, the LBDS as well as the 50NN rule.

4.3 Using Two-Level Decision Tree as a Base Classifier

Afterwards, two-level Decision Trees were used as weak learning base classifiers. A two-level Decision Tree is a tree with max depth=2. The proposed method, denoted as PSLBDT, is compared to the Boosting Decision Tree, denoted as BDT and the Local Boosting of Decision Trees, denoted as LBDT. Since the proposed method uses fifty neighbors a 50-Nearest Neighbors (50NN) classifier has included in the comparisons. In Table 7 the average accuracy of the compared methods is presented. Table 7 indicates that the hypotheses generated by PSLBDT are apparently better, since the PSLBDT algorithm has the best mean accuracy score in most cases.

The average rankings, according to the Friedman test, are presented in Table 8. The proposed algorithm was ranked in the first place again. Assuming significance level of 0.05 in Table 8, the *p*-value of the Friedman test indicates that the null hypothesis has to be rejected. So, there is at least one method that performs statistically different from the proposed method. Aiming to investigate the aforesaid, Finner's and Li's post hoc procedures were used again.

In Table 9 the *p*-values obtained by applying post hoc procedures over the results of Friedman's statistical test are presented. Finner's and Li's procedure rejects those hypotheses that have a *p*-value ≤ 0.05. That said, the adjusted *p*-values obtained through the application of the post hoc procedures are presented in Table 10. Both post hoc procedures agree that the PSLBDT algorithm performs significantly better than the BDT and the 50NN rule but not significantly better than the LBDT as far as the tested data sets are concerned.

4.4 Time Analysis

One of the two contributions of this study was to improve the classification time over the local boosting approach. In order to prove this, the total time that is required to predict all instances in the test folds was recorded. Specifically,

Table 4. Average rankings of Friedman test (DS)

Algorithm	Ranking
PSLBDS	1.1429
LBDS	2.4286
50NN	2.8571
BDS	3.5714
Statistic	26.228571
p-value	0.000009

Table 5. Post hoc comparison for the Friedmans test (DS)

i	Algorithm	$z = (R_0 - R_i)/SE$	p	Finner	Li
3	BDS	4.97709	0.000001	0.016952	0.052189
2	50NN	3.51324	0.000443	0.033617	0.052189
1	LBDS	2.63493	0.008415	0.05	0.05

Table 6. Adjusted p-values (DS)

i	Algorithm	$p_{Unadjusted}$	p_{Finner}	p_{Li}
3	BDS	0.000001	0.000002	0.000001
2	50NN	0.000443	0.000664	0.000446
1	LBDS	0.008415	0.008415	0.008415

Table 7. Average accuracy of the compared algorithms using a two-level decision tree as base classifier

Data set	PSLBDT	BDT	LBDT	50NN
cardiotocography	0.683 ± 0.020	0.584 ± 0.072	$\mathbf{0.686} \pm 0.017$	0.607 ± 0.029
cylinder-bands	$\mathbf{0.609} \pm 0.049$	0.608 ± 0.034	0.564 ± 0.025	0.582 ± 0.017
dermatology	$\mathbf{0.958} \pm 0.040$	0.800 ± 0.041	0.951 ± 0.020	0.902 ± 0.020
ecoli	$\mathbf{0.813} \pm 0.032$	0.753 ± 0.036	0.800 ± 0.030	0.780 ± 0.050
energy-y1	$\mathbf{0.845} \pm 0.060$	0.830 ± 0.064	0.844 ± 0.071	0.822 ± 0.091
glass	0.608 ± 0.048	0.569 ± 0.112	$\mathbf{0.652} \pm 0.057$	0.446 ± 0.169
low-res-spect	$\mathbf{0.877} \pm 0.042$	0.573 ± 0.136	0.872 ± 0.023	0.851 ± 0.023
magic	0.849 ± 0.006	$\mathbf{0.856} \pm 0.007$	0.841 ± 0.006	0.828 ± 0.004
musk-1	0.738 ± 0.072	$\mathbf{0.752} \pm 0.024$	0.746 ± 0.074	0.618 ± 0.096
ozone	0.967 ± 0.008	0.925 ± 0.064	0.888 ± 0.132	$\mathbf{0.971} \pm 0.001$
page-blocks	$\mathbf{0.960} \pm 0.010$	0.924 ± 0.023	0.956 ± 0.010	0.942 ± 0.007
pima	$\mathbf{0.763} \pm 0.023$	0.742 ± 0.014	0.730 ± 0.018	0.749 ± 0.017
synthetic-control	0.950 ± 0.011	0.830 ± 0.036	$\mathbf{0.953} \pm 0.016$	0.887 ± 0.030
tic-tac-toe	$\mathbf{0.893} \pm 0.078$	0.665 ± 0.126	0.889 ± 0.081	0.747 ± 0.101

the prediction of each test fold was executed three times and the minimum time was recorded for each fold. Then, the average of all folds was calculated. In Table 11 the average prediction time in seconds of LBDS, PSLBDS, LBDT and PSLBDTS is presented. In the case of one-level decision trees (LBDS, PSLBDS) the proposed method reduced the expected prediction time in more than 15 % in 6 of 14 cases, while in the case of two-level decision trees (LBDT, PSLBDT) the proposed method reduced the expected prediction time in more than 15 % in 7 of 14 cases. In Fig. 1 the absolute percentage changes are presented.

Table 8. Average rankings of Friedman test (two-level tree)

Algorithm	Ranking
PSLBDT	1.5
LBDT	2.2857
50NN	3.0714
BDT	3.1429
Statistic	15
p-value	0.001817

Table 9. Post hoc comparison for the Friedmans test (two-level tree)

i	Algorithm	$z = (R_0 - R_i)/SE$	p	Finner	Li
3	BDT	3.366855	0.00076	0.016952	0.046982
2	50NN	3.22047	0.00128	0.033617	0.046982
1	LBDT	1.610235	0.107347	0.05	0.05

Table 10. Adjusted p-values (two-level tree)

i	Algorithm	$p_{Unadjusted}$	p_{Finner}	p_{Li}
3	BDT	0.00076	0.002279	0.000851
2	50NN	0.00128	0.002279	0.001432
1	LBDT	0.107347	0.107347	0.107347

Table 11. Average prediction times, in seconds

Data set	LBDS	PSLBDS	LBDT	PSLBDT
cardiotocography	33.89	33.26	32.43	29.20
cylinder-bands	8.16	8.07	8.45	7.86
dermatology	3.56	3.52	3.28	3.20
ecoli	5.00	3.61	4.66	2.92
energy-y1	8.58	7.19	7.59	6.25
glass	3.39	3.37	3.46	3.16
low-res-spect	6.74	6.38	5.77	3.77
magic	257.14	160.31	213.59	107.98
musk-1	9.53	9.50	8.80	7.99
ozone	14.84	4.89	7.24	1.69
page-blocks	17.27	9.34	12.28	4.27
pima	11.72	8.90	11.07	7.56
synthetic-control	6.32	6.12	3.89	3.76
tic-tac-toe	13.56	13.56	12.18	12.00

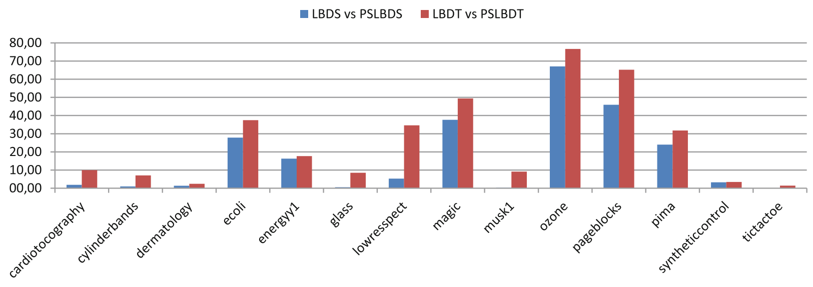

Fig. 1. Percentage change of prediction time between Local Boosting and the proposed method

5 Synopsis and Future Work

Local memory-based techniques delay the processing of the training set until they receive a request for an action like classification or local modelling. A data set of observed training examples is always retained and the estimate for a new test instance is obtained from an interpolation based on a neighborhood of the query instance.

In this research work at hand, a local boosting after prototype selection method is presented. Experiments on several data sets show that the proposed method significantly outperforms the boosting and local boosting method, in terms of classification accuracy and the time that is required to build a local model and classify a test instance. Typically, boosting algorithms are well known to be subtle to noise [2]. In the case of local boosting, the algorithm should handle sufficient noise and be at least as good as boosting, if not better. By means of the promising results obtained from performed experiments, one can assume that the proposed method can be successfully applied to the classification task in the real world case with more accuracy than the compared machine learning approaches.

In a following work the proposed method will be investigated as far as regression problems are concerned as well as the problem of reducing the size of the stored set of instances, by also applying feature selection instead of simple prototype selection.

References

1. Atkeson, C.G., Schaal, S., Moore, A.W.: Locally weighted learning. Artif. Intell. Rev. **11**(1), 11–73 (1997)
2. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting, and variants. Mach. Learn. **36**(1), 105–139 (1999)
3. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
4. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
5. Breiman, L., Friedman, J., Stone, C., Olshen, R.: Classification and Regression Trees. Chapman & Hall, New York (1993)

6. Demar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)
7. Finner, H.: On a monotonicity problem in step-down multiple test procedures. J. Am. Stat. Assoc. **88**(423), 920–923 (1993)
8. Freund, Y., Schapire, R.E.: Others: experiments with a new boosting algorithm. ICML. **96**, 148–156 (1996)
9. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**(200), 675 (1937)
10. Garca-Pedrajas, N., de Haro-Garca, A.: Boosting instance selection algorithms. Knowl. Based Syst. **67**, 342–360 (2014)
11. Garcia, S., Derrac, J., Cano, J.R., Herrera, F.: Prototype selection for nearest neighbor classification: taxonomy and empirical study. IEEE Trans. Pattern Anal. Mach. Intell. **34**(3), 417–435 (2012)
12. Iba, W., Langley, P.: Induction of one-level decision trees. In: Proceedings of the Ninth International Workshop on Machine Learning, pp. 233–240, ML 1992. Morgan Kaufmann Publishers Inc., San Francisco (1992)
13. Kleinberg, E.M.: A mathematically rigorous foundation for supervised learning. In: Kittler, J., Roli, F. (eds.) MCS 2000. LNCS, vol. 1857, pp. 67–76. Springer, Heidelberg (2000)
14. Kotsiantis, S.B., Kanellopoulos, D., Pintelas, P.E.: Local boosting of decision stumps for regression and classification problems. J. Comput. **1**(4), 30–37 (2006)
15. Li, J.: A two-step rejection procedure for testing multiple hypotheses. J. Stat. Plann. Infer. **138**(6), 1521–1527 (2008)
16. Lichman, M.: UCI Machine Learning Repository (2013)
17. Neo, T.K.C., Ventura, D.: A direct boosting algorithm for the k-nearest neighbor classifier via local warping of the distance metric. Pattern Recogn. Lett. **33**(1), 92–102 (2012)
18. Olvera-Lpez, J.A., Carrasco-Ochoa, J.A., Martnez-Trinidad, J.F., Kittler, J.: A review of instance selection methods. Artif. Intell. Rev. **34**(2), 133–144 (2010)
19. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, D.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
20. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
21. Rokach, L.: Ensemble-based classifiers. Artif. Intell. Rev. **33**(1–2), 1–39 (2010)
22. Segata, N., Blanzieri, E., Delany, S.J., Cunningham, P.: Noise reduction for instance-based learning with a local maximal margin approach. J. Intell. Inf. Syst. **35**(2), 301–331 (2010)
23. Vapnik, V.N.: Statistical Learning Theory: Adaptive and Learning Systems for Signal Processing, Communications, and Control. Wiley, New York (1998)
24. Wilson, D.L.: Asymptotic properties of nearest neighbor rules using edited data. IEEE Trans. Sys. Man Cybern. **2**(3), 408–421 (1972)
25. Witten, I.H., Frank, E., Hall, M.A.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Series in Data Management Systems, 3rd edn. Morgan Kaufmann, Burlington (2011)
26. Zhang, C.X., Zhang, J.S.: A local boosting algorithm for solving classification problems. Comput. Stat. Data Anal. **52**(4), 1928–1941 (2008)