Bagging and Boosting with Dynamic Integration of Classifiers

Alexey Tsymbal and Seppo Puuronen

Department of Computer Science and Information Systems,
University of Jyväskylä, P.O.Box 35, FIN-40351 Jyväskylä, Finland
{Alexey,sepi}@cs.jyu.fi

Abstract. One approach in classification tasks is to use machine learning techniques to derive classifiers using learning instances. The cooperation of several base classifiers as a decision committee has succeeded to reduce classification error. The main current decision committee learning approaches boosting and bagging use resampling with the training set and they can be used with different machine learning techniques which derive base classifiers. Boosting uses a kind of weighted voting and bagging uses equal weight voting as a combining method. Both do not take into account the local aspects that the base classifiers may have inside the problem space. We have proposed a dynamic integration technique to be used with ensembles of classifiers. In this paper, the proposed dynamic integration technique is applied with AdaBoost and bagging. The comparison results using several datasets of the UCI machine learning repository show that boosting and bagging with dynamic integration of classifiers results often better accuracy than boosting and bagging result with their original voting techniques.

1 Introduction

There are several approaches in data mining which tries to find previously unknown and potentially interesting patterns and relations in large databases [6]. One typical data mining task is to predict a value of an attribute of a new instance when the values of the other attributes of the instance are known. A common approach is to use a machine learning technique. A collection of instances with known values of all the attributes is treated as a training set for a machine learning algorithm that derives a classifier. This classifier is later used to predict the unknown value of the attribute for a new instance.

Decision committee learning has succeeded to reduce classification error [1,3,4,7,9,16,17,18,24]. A committee (final classifier) is composed of base classifiers (committee members), each of which makes its own classifications that are combined to create a single classification result of the whole committee. Two decision committee learning approaches, boosting [16,17] and bagging [3], have received extensive attention recently and they have been deeply analyzed [4,9,18]. Both generate by re sampling training sets from the original data set to the learning algorithm [24] which builds up a base classifier for each training set. There are two major differences between bagging and boosting. First, boosting changes adaptively the distribution of the training set based on the performance of previously created classifiers while bagging changes the distribution of the training set stochastically [24]. Second, boosting uses a function of the performance of a classifier as a weight for voting, while bagging uses equal weight voting [24]. With both techniques error decreases when the size of the committee increases, but the marginal error reduction of an additional member tends to decrease [24]. In general, bagging is more consistent, increasing the error of the base learner less frequently than boosting. However, boosting has greater average effect, leading to substantially larger error reductions than bagging on average [4]. The error reduction has been a little mysterious also to researchers and several analysis to reveal the real background of the phenomenon have been suggested [1,9,18]. One explanation is based on that boosting tends to reduce both the bias and variance terms of error while bagging tends to reduce the variance term only [1,4]. Another is using the concept of margin distribution [18], and another recent one establishes connection between boosting and additive models [9] but as comments, as [8] to the last paper show there is still much work to do before the behaviour of methods are deeply understood.

Both bagging and boosting uses a voting technique which is unable to take into account the heterogeneity of the instance space. When majority of the base classifiers give a wrong prediction for a new instance then the majority vote will result in a wrong prediction. The problem may consist in discarding base classifiers (by assigning small weights) that are highly accurate in a restricted region of the instance space because this accuracy is swamped by their inaccuracy outside the restricted area. It may also consist in the use of classifiers that are accurate in most of the space but still unnecessarily confuse the whole classification committee in some restricted areas of the space. To overcome this problem we have suggested a dynamic integration approach [13]. In this paper we compare bagging and boosting with and without our dynamic integration approaches using using several data sets from the UCI machine learning repository [2]. Some preliminary results with small committee sizes were presented in [20].

In chapter 2 the bagging and boosting algorithms are reviewed. Chpater 3 includes a short describtion of our dynamic integration methods. Chapter 4 includes the results of comparisons and chapter 5 concludes with a short summary and some further research topics.

2 Bagging and Boosting

In this chapter bagging and boosting algorithms to be used in comparisons are reviewed.

The bagging algorithm (**B**ootstrap **agg**regat**ing**) [3] uses bootstrap samples to build the base classifiers. Each bootstrap sample of *m* instances is formed by uniformly sampling *m* instances from the training set with replacement. The amount of the different instances in the bootstrap sample is for large *m* about 1-1/*e*=63.2%. This results that dissimilar base classifiers are built with unstable learning algorithms (e.g., neural networks, decision trees) [1] and the performance of the committee can become better. However, bagging may slightly degrade the performance of stable algorithms (e.g., *k*-nearest neighbor) because effectively smaller training sets are used to train each base classifier [3]. The bagging algorithm generates *T* bootstrap samples B_1, B_2, …, B_T and then the corresponding *T* base classifiers C_1, C_2, …, C_T. The final classification produced by the committee of these base classifiers is received using equal weight voting where ties are broken arbitrarily

Boosting was developed as a method for boosting the performance of any weak learning algorithm [17], which needs only to be a little bit better than random guessing [16]. The AdaBoost algorithm (**Ada**ptive **Boost**ing) [7] was introduced as an improvement of the initial boosting algorithm and several further variants are presented. The concept used to correspond a generation of one base classifier is a trial. AdaBoost changes the weights of the training instances after each trial based on the misclassifications made by the resulting base classifier trying to force the learning algorithm to minimize the expected error over different input distributions [1]. Similarly, the correctly classified instances will have a lower total weight.Thus AdaBoost generates during the *T* trials *T* training sets S_1, S_2, …, S_T where instances have weights and thus *T* base classifiers C_1, C_2, …, C_T are built. A final classification produced by the committee of these base classifiers is received using a weighted voting scheme where the weight of each classifier depends on its performance on the training set used to build it.

The AdaBoost algorithm requires a weak learning algorithm whose error is bounded by a constant strictly less than ½. In practice, the learning algorithm that we use provide no such guarantee. The original algorithm aborted when the error bound was breached [17], but this case is fairly frequent for some multiclass problems [1]. In this paper, a bootstrap sample from the original data *S* is generated in such case, and the boosting continues up to a limit of 25 such samples at a given trial, as proposed in [1]. The same is done when the error is equal to zero at some trial. Some implementations of AdaBoost use boosting by resampling because the inducers used were unable to support weighted instances [1]. In this work, boosting by reweighting is implemented with the C4.5 decision tree learning algorithm [15], which is a more direct implementation of the theory. Some evidence exists that reweighting works better in practice [1].

3 Dynamic Integration of Classifiers

The challenge of integration is to decide which classifier to rely on or how to combine classifications produced by several classifiers. The integration approaches can be divided into static and dynamic ones. Recently, two main approaches to integration have been used: first, combination of the classifications produced by the base classifiers, and, second, selection of the best classifier from the base classifiers. The most popular method of combining classifiers is voting [1]. More sophisticated selection approaches use estimates of the local accuracy of the base classifiers by considering errors made in similar instances [12] or the meta-level classifiers ("referees"), which predict the correctness of the base classifiers for a new instance [11].

We have presented a dynamic integration technique that estimates the local accuracy of the base classifiers by analyzing the accuracy in near-by instances [13]. Knowledge is collected about the errors that the base classifiers make with the training instances and this knowledge is taken benefit during the classification of new instances. The goal is to use each base classifier just in that subarea for which it is the most reliable one. The C4.5 decision tree induction algorithm is used to predict the errors of the base classifiers [15] and in the dynamic integration the weighted nearest neighbor classification (WNN) is used [5].

The dynamic integration approach contains two phases. In the learning phase, information is collected about how each classifier succeeds with each training set instance. With bagging and boosting this is done directly using the base classifiers produced by C4.5. In the original version of our algorithm we have used cross-validation technique to estimate the errors of the base classifiers on the training set [13]. In the application phase, the final classification of the committee is formed so that for a new instance the seven nearest neighbor instances of the learning set are found out. The error information of each classifier with these instances is collected and then depending on the dynamic integration approach the final classification is made using this information. Two of these three different functions implementing the application phase were considered [13]. The first function DS implements Dynamic Selection. In the DS application phase the classification error is predicted for each base classifier using the WNN procedure and a classifier with the smallest error (with the least global error in the case of ties) is selected to make the final classification. The second function DV implements Dynamic Voting. In the DV application phase each component classifier receives a weight that depends on the local classifier's performance and the final classification is conducted by voting classifier predictions with their weights. The third function DVS was first presented in [21] and it applies mixed approach. Using the collected error information first about half of classifiers (the better half) is selected and then the final classification is derived using weighted voting among them.

Previously an application of the dynamic classifier integration in medical diagnostics was considered in [19,20,22]. A number of experiments comparing the dynamic integration with such widely used integration approaches as CVM, and weighted voting were also conducted [13,14,22,23]. The comparison results show that the dynamic integration technique outperforms often weighted voting and CVM. In [21] the dynamic classifier integration was applied to decision committee learning, combining the generated classifiers in a more sophisticated manner than voting. This paper continues this research considering more experiments comparing bagging and boosting with and without the three dynamic integration functions.

4 Experiments

In this chapter we present experiments where the dynamic classifier integration algorithm is used with classifiers generated by AdaBoost and bagging. The experimental setting is described before the results of the experiments. We have used nine datasets from the UCI machine learning repository [2]. Previously the dynamic classifier integration was experimentally evaluated in [13] and preliminary comparisons with AdaBoost and bagging were made in [21].

The main characteristics of the eight datasets are presented in Table 1. The table includes the name of the dataset, number of instances included in the dataset, number of different classes of instances, and numbers of different kind of features included in the instances.

Table 1. Characteristics of the datasets

Dataset	Instances	Classes	Features	
			Discrete	Continuous
Breast	286	2	9	0
Diabetes	768	2	0	8
Glass	214	6	0	9
Heart	270	2	8	5
Iris	150	3	0	4
Liver	345	2	0	6
MONK-1	432	2	6	0
MONK-2	432	2	6	0

For each dataset 30 test runs are made. First, in each run 30 percent of the instances of the data set are randomly picked up to the test set. Then the rest 70 percent of the instances are passed to bagging and boosting algorithm. These manipulate the learning set as described in Chapter 2 producing required amount of base classifiers using C4.5 decision tree algorithm with pruning [15]. We make experiments with 5, 10, and 25 base classifiers in the committees.

We calculate average accuracy numbers for base classifiers, for ordinary bagging and boosting, and for bagging and boosting with the three dynamic integration method described in Chapter 3 separately. In the learning phase of the dynamic integration part information about the errors made by each base classifier is collected for all the instances included in the 70 percent of the original data set, selected in the beginning of each run. In the application phase of the dynamic integration the instances included in the test set is used. For each of them the classification errors of all the base classifiers of the committee is collected for seven nearest neighbors of the new instance. The selection of the number of nearest neighbors has been discussed in [13]. Based on the comparisons between different distance functions for dynamic integration presented in [14] we decided to use the Heterogeneous Euclidean-Overlap Metric, which produced good test results earlier. The test environment was implemented within the MLC++ framework (the machine learning library in C++) [10].

Table 2 presents accuracy values for Bagging with different voting methods. The first column includes the name of the corresponding data set. The second column tells how many base classifiers were produced to the committee. The next three columns include the average of the minimum accuracies of the base classifiers (min), the average of average accuracies of the base classifiers (aver), and the average of the maximum accuracies of the base classifiers (max) over the 30 runs.

Table 2. Accuracy values for Bagging with different voting methods

	# of base	C4.5 base classifiers			Bagging with different voting			
DB	classifiers	min	aver	max	EWV	DS	DV	DVS
Breast	5	0.649	0.691	0.739	0.710	0.697	0.710	0.706
	10	0.625	0.686	0.746	0.717	0.693	0.712	0.702
	25	0.605	0.689	0.763	0.717	0.692	0.720	0.714
Diabetes	5	0.675	0.706	0.737	0.740	0.704	0.739	0.724
	10	0.668	0.708	0.745	0.752	0.707	0.751	0.740
	25	0.659	0.710	0.760	0.755	0.710	0.755	0.751
Glass	5	0.544	0.608	0.670	0.656	0.663	0.675	0.677
	10	0.522	0.608	0.687	0.669	0.666	0.683	0.689
	25	0.500	0.610	0.711	0.682	0.681	0.696	0.705
Heart	5	0.677	0.731	0.788	0.777	0.747	0.777	0.765
	10	0.659	0.732	0.804	0.788	0.748	0.790	0.779
	25	0.641	0.735	0.818	0.799	0.751	0.798	0.786
Iris	5	0.921	0.942	0.957	0.950	0.944	0.950	0.944
	10	0.914	0.943	0.963	0.947	0.944	0.949	0.943
	25	0.904	0.942	0.965	0.947	0.941	0.948	0.947
Liver	5	0.557	0.610	0.657	0.645	0.621	0.653	0.637
	10	0.541	0.606	0.669	0.649	0.622	0.663	0.657
	25	0.526	0.610	0.695	0.669	0.621	0.682	0.675
MONK-1	5	0.727	0.827	0.928	0.894	0.975	0.900	0.964
	10	0.699	0.834	0.947	0.895	0.988	0.931	0.975
	25	0.666	0.826	0.976	0.913	0.995	0.942	0.976
MONK-2	5	0.510	0.550	0.588	0.567	0.534	0.565	0.547
	10	0.499	0.551	0.600	0.600	0.536	0.566	0.540
	25	0.487	0.551	0.610	0.582	0.539	0.571	0.537
Average	5	0.658	0.708	0.758	0.742	0.736	0.746	0.746
	10	0.641	0.709	0.770	0.752	0.738	0.756	0.753
	25	0.624	0.709	0.787	0.758	0.741	0.764	0.761

The last four columns on the right-hand side of Table 2 include average accuracies for different voting methods. These are: equal weight voting (EWV), dynamic selection (DS), dynamic voting (DS), and dynamic voting with selection (DVS). DS, DV, and DVS were considered in chapter 3. Previous experiments with these two integration strategies [13] have shown that the accuracies of the strategies usually differ significantly; however, it depends on the dataset, what a strategy is preferable. In this paper we apply also a combination of the DS and DV strategies that were in [21] expected to be more stable than the other two.

The accuracies in the table 2 confirm the known result that raising the number of committee members makes bagging more accurate. In this experiment the average increase of accuracy over the 8 data sets was 1.3% and 2.1% when the number of the members was raised from 5 to 10 and 25 correspondingly. The increase was almost same for DV (1.3% and 2.4%) and DVS (1.0% and 2.1%) but much lower for DS (0.3% and 0.8%). It seems to be that bagging with dynamic selection is not able to take as efficiently benefit of the increase of the base classifiers, at least with these data sets. When individual data sets are looked it is noticed that with ordinary equal weight voting and dynamic voting the accuracy decreases only with Iris dataset, with dynamic selection with Iris and Breast datasets, and with DVS with MONK-2 dataset and with Breast and Iris datasets when the number is raised from 5 to 10.

When the different voting techniques are compared it is noticed that in average over the 8 datasets DV and DVS produce only about 0.5% higher accuracies than the ordinary voting. Dynamic selection, on the other hand gives in average about 2% smaller accuracy for bigger committees. There is big difference between individual data sets in the average accuracies achieved. The biggest ones are that both DVS and DS produce over 10% smaller accuracy with MONK-2 dataset with 10 base classifiers, and DS over 10% higher accuracy with MONK-1 dataset with 10 base classifiers than the ordinary voting. As a summary it can be said that in average over these 8 datasets there seems not to be great benefit about the dynamic integration with bagging but because there are many dataset related difference it needs further systematic research before being able to make any firm conclusions.

Table 3 presents accuracy values for AdaBoost with different voting methods. The columns are same as in Table 2 except that the ordinary voting with boosting is weighted voting (WV) and again the accuracies are averages over 30 runs.

The accuracies in the table 3 confirm the known result that raising the number of committee members makes also boosting more accurate. In this experiment the average increase of accuracy over the 8 data sets was 0.7% and 1.3% when the number of the members was raised from 5 to 10 and 25 correspondingly. The increase was a little bit higher for DV (1.3% and 1.7%) and DVS (0.9% and 1.7%) but DS suffered from additional base classifiers (-0.0% and -0.7%). It seems to be that also boosting with dynamic selection is not able to take as efficiently benefit of the increase of the base classifiers, at least with these data sets. When individual data sets are looked it is noticed that with ordinary weighted voting and dynamic voting the accuracy decreases only with MONK-2 dataset and with Liver dataset when the amount is raised from 5 to 10. The accuracy of dynamic voting with selection decreases only with the MONK-2 dataset but the dynamic selection has problems with almost every dataset.

When the different voting techniques are compared it is noticed that in average over the 8 datasets DVS produces 0.9% higher accuracy with 25 base classifiers, 0.8% higher accuracy with 10 base classifiers, and 06% higher accuracy with 5 base classifiers than the ordinary weighted voting. The average numbers of DV over the datasets are also a little bit better than the ordinary weigted voting. Instead, the dynamic selection produces lower accuracies, in average about 1.0%, 1.7%, and 3.0% for 5, 10, and 25 base classifiers, correspondingly.

Table 3. Accuracy values for AdaBoost with different voting methods

	# of base	C4.5 base classifiers			AdaBoost with different voting			
DB	classifiers	min	aver	max	WV	DS	DV	DVS
Breast	5	0.577	0.637	0.703	0.684	0.664	0.686	0.689
	10	0.548	0.628	0.708	0.686	0.658	0.693	0.689
	25	0.504	0.618	0.717	0.687	0.655	0.697	0.697
Diabetes	5	0.640	0.680	0.720	0.722	0.690	0.722	0.710
	10	0.618	0.674	0.726	0.727	0.685	0.729	0.723
	25	0.528	0.658	0.732	0.738	0.685	0.741	0.742
Glass	5	0.524	0.595	0.666	0.683	0.667	0.685	0.685
	10	0.457	0.572	0.676	0.697	0.668	0.702	0.705
	25	0.371	0.533	0.683	0.703	0.661	0.704	0.708
Heart	5	0.634	0.705	0.773	0.763	0.727	0.763	0.751
	10	0.589	0.685	0.776	0.770	0.719	0.771	0.763
	25	0.495	0.655	0.785	0.769	0.719	0.773	0.776
Iris	5	0.911	0.939	0.965	0.945	0.943	0.945	0.945
	10	0.895	0.939	0.967	0.949	0.946	0.949	0.949
	25	0.846	0.933	0.974	0.951	0.945	0.949	0.948
Liver	5	0.550	0.601	0.650	0.642	0.609	0.642	0.624
	10	0.526	0.593	0.656	0.639	0.606	0.640	0.634
	25	0.474	0.581	0.668	0.645	0.593	0.646	0.647
MONK-1	5	0.740	0.849	0.936	0.949	0.966	0.948	0.967
	10	0.705	0.843	0.950	0.968	0.980	0.974	0.984
	25	0.570	0.803	0.965	0.989	0.983	0.990	0.993
MONK-2	5	0.473	0.535	0.594	0.509	0.572	0.509	0.559
	10	0.458	0.535	0.607	0.503	0.575	0.517	0.538
	25	0.442	0.534	0.625	0.493	0.558	0.501	0.520
Average	5	0.631	0.693	0.751	0.737	0.730	0.738	0.741
	10	0.600	0.684	0.758	0.742	0.730	0.747	0.748
	25	0.529	0.664	0.769	0.747	0.725	0.750	0.754

There is big difference between individual data sets in the average accuracies only with the dynamic selection method and MONK-2 dataset with the dynamic voting with selection. As a summary it can be said that in average over these 8 datasets there might be benefit about the dynamic integration with boosting when dynamic voting especially with selection is used, but further systematic research is needed to confirm this.

5 Conclusion

Decision committee learning has demonstrated spectacular success in reducing classification error with learned classifiers. These techniques a committee of base classifiers which produce the final classification using some voting method. Ordinary voting methods has however an important shortcoming, that they do not take into account local context related things.

In this paper a technique for dynamic integration of classifiers was experiment as a method instead of ordinary voting methods with bagging and boosting. The technique for dynamic integration of classifiers is based on the assumption that each base classifier is the best inside certain sub areas of the whole feature space. The considered algorithm for dynamic integration of classifiers is a new variation of stacked generalization, which uses a distance metric to locally estimate the errors of base classifiers.

The proposed dynamic integration technique was evaluated with AdaBoost and Bagging, the decision committee approaches which have received extensive attention recently, on eight datasets from the UCI machine learning repository. The results achieved are promising and show that especially boosting but in some contexts also bagging might give better accuracy with dynamic integration of classifiers than with simple voting.

Further analysis and experiments are needed to make deeper analysis of combining the dynamic integration of classifiers with different approaches to decision committee learning.

Acknowledgments: This research is partly supported by the COMAS Graduate School of the University of Jyväskylä. We would like to thank the UCI machine learning repository of databases, domain theories and data generators for the datasets, and the machine learning library in C++ for the source code used in this study. We want to thank also the anonymous reviewers for their positive and helpful criticisms.

References

1. Bauer, E., Kohavi, R.: An Empirical Comparison of Voting Classification Algorithms: Bagging, Boosting, and Variants. Machine Learning, Vol.36 (1999) 105-139.
2. Blake, C.L., Merz, C.J.: UCI Repository of Machine Learning Databases [http:// www.ics.uci.edu/ ~mlearn/ MLRepository.html]. Dep-t of Information and CS, Un-ty of California, Irvine CA (1998).
3. Breiman, L.: Bagging Predictors. Machine Learning, Vol. 24 (1996) 123-140.
4. Breiman, L.: Arcing classifiers. The Annals of Statistics, Vol. 26, No. 3 (1998) 801-823.
5. Cost, S., Salzberg, S.: A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine Learning, Vol. 10, No. 1 (1993) 57-78.
6. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R.: Advances in Knowledge Discovery and Data Mining. AAAI/ MIT Press (1997).
7. Freund, Y., Schapire, R.E.: A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. In: Proc. 2nd European Conf. on Computational Learning Theory, Springer-Verlag (1995) 23-37.

8. Freund, Y., Schapire, R.E.: Discussion of the Paper "Additive Logistic Regression: a Statistical View of Boosting by Jerome Friedman, Trevor Hastie and Robert Tibshirani. The Annals of Statistics, Vol. 28, No. 2 (2000).
9. Friedman, J., Hastie, T, Tibshirani, R.: Additive Logistic Regression: a Statistical View of Boosting. The Annals of Statistics, Vol. 28, No. 2 (2000).
10. Kohavi, R., Sommerfield, D., Dougherty, J.: Data Mining Using MLC++: A Machine Learning Library in C++. Tools with Artificial Intelligence, IEEE CS Press (1996) 234-245.
11. Koppel, M., Engelson, S.P.: Integrating Multiple Classifiers by Finding their Areas of Expertise. In: AAAI-96 Workshop On Integrating Multiple Learning Models (1996) 53-58.
12. Merz, C.: Dynamical Selection of Learning Algorithms. In: D.Fisher, H.-J.Lenz (eds.), Learning from Data, Artificial Intelligence and Statistics, Springer-Verlag, NY (1996).
13. Puuronen, S., Terziyan, V., Tsymbal, A.: A Dynamic Integration Algorithm for an Ensemble of Classifiers. In: Z.W. Ras, A. Skowron (eds.), Foundations of Intelligent Systems: ISMIS'99, Lecture Notes in AI, Vol. 1609, Springer-Verlag, Warsaw (1999) 592-600.
14. Puuronen, S., Tsymbal, A., Terziyan, V.: Distance Functions in Dynamic Integration of Data Mining Techniques. In: B.V. Dasarathy (ed.), Data Mining and Knowledge Discovery: Theory, Tools, and Techniques, SPIE Press, USA (2000) to appear.
15. Quinlan, J.R.: C4.5 Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA (1993).
16. Schapire, R.E.: A Brief Introduction to Boosting. In: Proc. 16th Int. Joint Conf. AI (1999).
17. Schapire, R.E.: The Strength of Weak Learnability. Machine Learning, Vol. 5, No. 2 (1990) 197-227.
18. Schapire, E., Freaund, Y., Bartlett, P., Lee, W.S.: Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics, Vol. 26, No 5 (1998) 1651-1686.
19. Skrypnik, I., Terziyan, V., Puuronen, S., Tsymbal, A.: Learning Feature Selection for Medical Databases. In: Proc. 12th IEEE Symp. on Computer-Based Medical Systems CBMS'99, IEEE CS Press, Stamford, CT (1999) 53-58.
20. Terziyan, V., Tsymbal, A., Puuronen, S.: The Decision Support System for Telemedicine Based on Multiple Expertise. Int. J. of Medical Informatics, Vol. 49, No. 2 (1998) 217-229.
21. Tsymbal, A.: Decision Committee Learning with Dynamic Integration of Classifiers. In: Proc. 2000 ADBIS-DASFAA Symposium on Advances in Databases and Information Systems, Prague, September 2000, Lecture Notes in Computer Science, Springer-Verlag (to appear).
22. Tsymbal, A., Puuronen, S., Terziyan, V.: Advanced Dynamic Selection of Diagnostic Methods. In: Proceedings 11th IEEE Symp. on Computer-Based Medical Systems CBMS'98, IEEE CS Press, Lubbock, Texas, June (1998) 50-54.
23. Tsymbal, A., Puuronen, S., Terziyan, V.: Arbiter Meta-Learning with Dynamic Selection of Classifiers and its Experimental Investigation. In: J.Eder, I.Rozman, T.Welzer (eds.), Advances in Databases and Information Systems: 3rd East European Conference ADBIS'99, Lecture Notes in CS, Vol. 1691, Springer-Verlag, Maribor (1999) 205-217.
24. Webb, G.I.: MultiBoosting: A Technique for Combining Boosting and Wagging. Machine Learning (2000) in press.