Ensemble of Classifiers to Improve Accuracy of the CLIP4 Machine Learning Algorithm

Lukasz A. Kurgan [*,a,b] and Krzysztof J. Cios [**,a,b,c,d]
[a] University of Colorado at Denver, [b] University of Colorado at Boulder
[c] University of Colorado Health Sciences Center, [d] 4cData, LLC, Golden, CO

ABSTRACT

Machine learning, one of the data mining and knowledge discovery tools, addresses automated extraction of knowledge from data, expressed in the form of production rules. The paper describes a method for improving accuracy of rules generated by inductive machine learning algorithm by generating the ensemble of classifiers. It generates multiple classifiers using the CLIP4 algorithm and combines them using a voting scheme. The generation of a set of different classifiers is performed by injecting controlled randomness into the learning algorithm, but without modifying the training data set. Our method is based on the characteristic properties of the CLIP4 algorithm. The case study of the SPECT heart image analysis system is used as an example where improving accuracy is very important. Benchmarking results on other well-known machine learning datasets, and comparison with an algorithm that uses boosting technique to improve its accuracy are also presented. The proposed method always improves the accuracy of the results when compared with the accuracy of a single classifier generated by the CLIP4 algorithm, as opposed to using boosting. The obtained results are comparable with other state-of-the-art machine learning algorithms.

Keywords: ensemble of classifiers, boosting, CLIP4, machine learning, data mining and knowledge discovery

1. INTRODUCTION

In this section the proposed method and the CLIP4 algorithm are described.

1.1. The Approach

An ensemble of classifiers is a set of classifiers, whose individual classification decisions are combined in some way, typically by a weighted or un-weighted voting, to classify new examples [13].

There are four general ways to build ensemble classifiers: by sub-sampling the training data, manipulating the input features or output targets, and finally by injecting randomness [11]. The proposed method uses the latter approach. The idea of injecting the randomness into the learning algorithm was used by many researchers. In the field of neural networks Kollen and Pollack [18] were experimenting with backpropagation algorithm for networks with different, randomly set initial weights values, and Parmanto, Munro, and Doyle [24] generated different classifiers by manipulating the training data set in random fashion. In the field of decision trees Kwok and Carter [20] incorporated randomness into C4.5 algorithm, and Dietterich and Kong [12] have implemented a random variant of C4.5 that performed better than C4.5 and the bagged C4.5. The other example is FOIL algorithm that was modified by Ali and Pazzani [1] by injecting randomness into the algorithm and generating a set of classifiers instead of a single classifier; the overall performance of the algorithm was significantly improved.

After we generate the ensemble of classifiers we combine their classification decisions into a final classification for the tested example. The described method uses weighted voting scheme that is similar to a scheme used in the ADABOOST algorithm [26]. The weights values are obtained by calculating the accuracy of each individual classifier on the entire training data set.

The developed method uses ensemble of classifiers to improve the accuracy of rules generated using the CLIP4 algorithm on the unseen testing data when compared to a single classifier. The resulting algorithm is competitive with other approaches for improvement of accuracy of other machine learning algorithms.

[*] lkurgan@carbon.cudenver.edu; phone 1 303 5562352; fax 1 303 5568369; http://isl.cudenver.edu/lkurgan; University of Colorado at Denver, Department of Computer Science and Engineering, P.O. Box 173364, Denver, CO 80217
[**] Krys.Cios@cudenver.edu; phone 1 303 5564314; fax 1 303 5568369; http://isl.cudenver.edu/cios; University of Colorado at Denver, Department of Computer Science and Engineering, P.O. Box 173364, Denver, CO 80217

1.2. The CLIP4 Algorithm

The proposed method is tightly connected with characteristic features of the CLIP4 algorithm. CLIP4 [8] is a supervised hybrid machine learning algorithm that combines ideas of rule and decision tree algorithms. CLIP stands for Cover Learning (using) Integer Programming. The original algorithm, CLILP2 was developed in 1995 [4, 5] and later improved as CLIP3 algorithm [6]. Most recently several new significant features were added to the algorithm that resulted in the CLIP4 algorithm [8].

The CLIP4 algorithm generates classification rules of a class (concept) from positive examples. It uses negative examples to derive the partitioning of the set of positive examples.

Examples are described [22, 23] by a set of K attribute-value pairs: $e = \wedge_{j=1}^{K}[a_j \# v_j]$ where a_j denotes j-th attribute with value $v_j \in d_j$, and # is a relation (=, <, ≈, ≤, etc.). The CLIP4 algorithm generates production rules in the form of: IF $(s_1 \wedge \ldots \wedge s_m)$ THEN class = class_i, where $s_i = [a_j \neq v_j]$ is a single selector.

Many of the key operations performed within the CLIP4 algorithm are modeled and solved by a simplified version of the integer programming model, known as the set-covering (SC) problem. The CLIP4 algorithm consists of three phases:

1. In phase I positive data is partitioned, using the SC problem, into subsets of similar data. The subsets are stored in a decision-tree like manner, where node of the tree represents one data subset. Each level of the tree is generated using one negative example for building the SC model. The solution of the SC problem is used to find selectors that distinguish between all positive and this particular negative example. These selectors are used to generate new branches of the tree. During the tree growing pruning is performed to eliminate noise from the data, and to avoid excessive growth of the tree.
2. In phase II a set of best subsets, which are stored as tree leaves, is selected. Selection is performed based on two criteria: large subsets are preferred over small ones since the rules generated using them will be "stronger" and more general, and all accepted subsets (between them) must cover the entire positive training data. All selected subsets are used to generate the rules. To generate a rule, back-projection of selected positive data subsets is performed using the entire negative data set and then the resulting data structure (matrix) is solved using the SC problem. The solution of the SC problem is used to generate a rule, and the process repeats itself for every selected positive data subset.
3. In phase III a set of strongest rules is selected from all generated rules. Rules that cover the most examples from the positive set are chosen. If there is a tie between two or more "best rules" the shortest rule is chosen, i.e. the rule that involves minimal number of selectors.

Detailed description of the algorithm can be found in [8, 9]. The CLIP4 algorithm is robust to noisy and missing-value data, and generates strong classifiers. Common to all phases of the CLIP4 algorithm is the use of the SC problem to perform crucial operations on the data, like selecting the most discriminating features to grow new branches of the tree, selecting the data subset that will generate the least overlapping and the most general rules, and finally generating the rules from the selected subsets. The proposed method is based on introduction of the controlled randomness into the heuristic method that was developed to solve the SC problem in the CLIP4 algorithm [8].

1.2.1. The Set Covering Problem

Integer programming models are used for minimization or maximization of a function, subject to a large number of constraints [25]. There are several solutions to an integer programming model depending on the method used. There are polynomial algorithms [10, 17] and non-polynomial algorithms [25] that find solutions to an integer programming model.

In the CLIP4 algorithm a simplified version of the general integer programming model is used. Several simplifications are applied: the function that is subject of optimization has all coefficient values equal to one, constraint function coefficients have binary values and all constraint functions are greater or equal to one. This integer linear programming problem is known in the literature as the set-covering problem [15, 2]. For this problem only *approximate solution* can be found, which in some cases may be optimal.

An example SC problem used by the CLIP4 algorithm is shown in Figure 1. The model can be transformed and solved in its matrix representation, see Figure 1.

$$\begin{gathered}\textit{Minimize}:\\ x_1+x_2+x_3+x_4=Z\\ \textit{Subject to}:\\ x_1+x_2+x_3\geq 1\\ x_1+x_4\geq 1\\ x_3\geq 1\\ \textit{Solution}:\\ Z=2,\quad \textit{when}\ x_1=1,x_2=0,x_3=1,x_4=0\end{gathered}\qquad\qquad \begin{gathered}\textit{Minimize}:\\ x_1+x_2+x_3+x_4=Z\\ \textit{Subject to}:\\ \begin{bmatrix}1,1,1,0\\1,0,0,1\\0,0,1,0\end{bmatrix}\cdot\begin{bmatrix}x_1\\x_2\\x_3\\x_4\end{bmatrix}\geq 1\end{gathered}$$

Fig. 1. Set covering model used by the CLIP4 algorithm and its solution: on the left in standard form, on the right in matrix form.

1.2.2. The Set Covering Problem Solution in CLIP4

The solution to the SC problem can be found using only constraint coefficients matrix, which we call BINary matrix. Columns of this matrix correspond to variables of the optimized function (attributes). Rows correspond to function constrains (examples). The solution is obtained by selecting minimal number of matrix columns in such a way that for every row there will be at least one matrix cell with the value of 1 for the selected matrix columns. The solution consists of the binary vector of the selected columns. In order to solve this SC problem a heuristic method was developed, its pseudo-code follows.

```
Given: BINary matrix, Initialize: Remove all empty rows from the BINary matrix; if the matrix has no ones then
    return error.
1   Select active rows that have the minimum number of 1's in rows – min-rows
2   Select the columns that have the maximum number of 1's within the min-rows – max-columns
3   Within max-columns find the columns that have the maximum number of 1's in all active rows – max-max-
    columns, if there is more then one max-max-column go to 4., otherwise go to 5.
4   Within max-max-columns find the first column that has the lowest number of 1's in the inactive rows
5   Add the selected column to the solution
6   Mark the inactive rows, if all the rows are inactive then terminate; otherwise go to 1.
where: the active row is the row that is not covered by the partial solution, and the inactive row is the row that is
already covered by the partial solution.
```

The goal is to get the best *approximate solution* in terms of the minimal number of columns (attributes) used, and the least number of 1's in the selected columns. This can be solved by minimizing the number of 1's that overlap among different columns and within the same row (step 4 of the method). This solution results in generating more general and accurate rules. The above method was developed for the CLIP4 algorithm, and the comparison of the results obtained by the CLIP3 and the CLIP4 algorithms [8] shows that the new method resulted in significant improvement of accuracy of the generated rules. An example of the new method for solving the SC problem is shown in Figure 2. The final solution consists of the second and fourth columns. It has exactly as many 1's in the selected two columns as there are rows. There are no overlapping 1's in the same rows, thus the optimal solution is obtained.

The previous approach to solving the SC problem (used in the CLIP3 algorithm) [6, 7] was based on counting number of 1's in active columns. The columns that have the bigger score were chosen and added to the solution. Then the inactive rows were marked and the process is repeated. Figure 2 shows the results that are obtained by using the method for solving the SC problem used in the CLIP3 algorithm. There are two possible final solutions, depending on how the choice of the columns is performed. Solution one has one 1 overlapping in the first row; solution two has two 1's overlapping in the first and third rows. The average number of overlapping 1's is 1.5, and thus no optimal solution is obtained.

2. THE DEVELOPED METHOD TO OBTAIN ENSEMBLE OF CLASSIFIERS

The developed ensemble classifier generation method is based on the semi-random method for obtaining solutions for the SC problem. Each classifier is generated separately using the same training data set. By generating randomly sub-optimal solutions different classifiers are obtained that perform very similarly to a classifier generated by a regular (without any randomness) method for solving the SC problem. The generated classifiers are later combined by weighted voting scheme using the accuracy of each classifier on the training data as a weight.

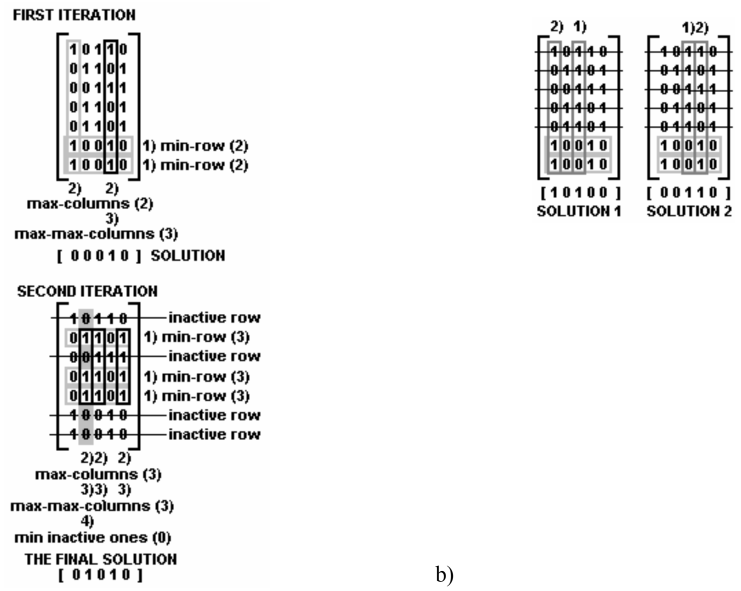

Fig. 2. a) Solution of the SC problem using the heuristic algorithm developed for the CLIP4 algorithm, b) Solution using the heuristic algorithm developed for the CLIP3 algorithm.

The pseudo-code for semi-random method for solving the SC problem is shown below. We emphasize that this method generates in general good *approximate solutions*, and the randomness that is introduced is well “controlled”.

```
Given: BINary matrix, Initialize: Remove all empty rows from the BINary matrix; if the matrix has no ones then
    return error.
1   Select active rows that have the minimum number of 1’s in rows – min-rows
2   Select the columns that have the maximum number of 1’s within the min-rows – max-columns
3   Select randomly one of the max-columns
4   Add the selected column to the solution
5   Mark the inactive rows, if all the rows are inactive then terminate, otherwise go to 1.
```

The example illustrating the method is shown in Figure 3. There are five possible final solutions, depending on the random choice performed in step 3 of the algorithm. The two optimal solutions were achieved (1 and 5), and two solutions with one 1 overlapping (3 and 4) and one solution with two 1’s overlapping (3). The average number of overlapping ones is 0.8 and the two optimal solutions were achieved. The advantage of this method is that by applying it within the CLIP4 algorithm we generate multiple, different, and very strong classifiers.

The developed algorithm for generation of the ensemble classifier for the CLIP4 algorithm follows.

```
Given: Training data, where $x_j \in X$ is an example, and $y_i \in Y$ is its corresponding class label
Initialize: Set the semi-random mode of solving the SC problem, T - the number of the classifiers to be generated
For t=1,…,T
1   Generate a set of rules (classifier) using the CLIP4 algorithm
2   Assign a weight $\alpha_t$ to the generated classifier $c_i : X \rightarrow Y$, where: $\alpha_t$ = accuracy of $c_i$ on the entire training data set

Classify the examples using:
    $y_i = \arg\max_i (\sum_{j=1}^{T} \alpha_j c_j \; if \; c_j = y_i)$
```

The developed method generates a set of T classifiers and applies the weighted voting scheme to classify the examples. The method was tested on six datasets showing significant improvement in the accuracy of the generated classifiers and was compared with a single classifier generated by the CLIP4 algorithm.

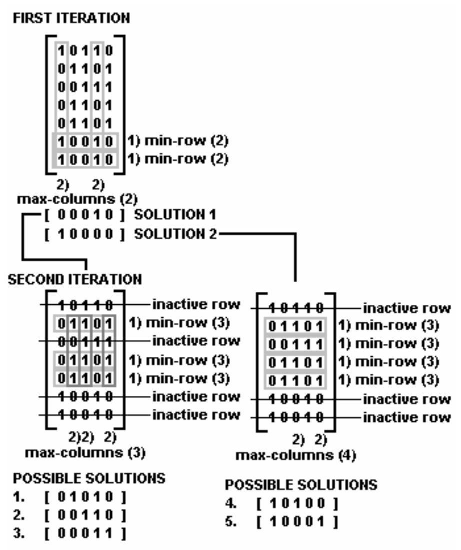

Fig. 3. Solution of the SC problem by using the heuristic semi-random algorithm developed to generate ensemble classifier with the CLIP4 algorithm.

3. EXPERIMENTS AND RESULTS

The goal of the experiments was to investigate the gain in terms of the accuracy when using the ensemble of classifiers vs. a single classifier generated by the CLIP4 algorithm. The comparison of accuracy results with other learning algorithms was also performed.

Another goal was to investigate how the number of generated classifiers in the ensemble affects its accuracy and the number of generated rules. Because of a trade-off between the maximal accuracy that can be obtained and the size of the ensemble of classifiers the heuristic guideline was developed.

3.1. Case study of the System for Analysis of the SPECT Heart Images

One of the applications of the CLIP4 classifier is a system for computerized diagnosis of myocardial perfusion from cardiac Single Proton Emission Computed Tomography (SPECT) [19]. The CLIP4 classifier is used to generate the overall diagnosis, which describes the perfusion of the entire left ventricle muscle for the SPECT study. The overall diagnosis is computed based on 22 partial diagnoses that are generated during the automated diagnostic process. The achieved accuracy of the CLIP4 rules was 86.1%. Since the problem concerns medical domain the accuracy of computer-generated diagnosis is an extremely important factor, so we investigate alternative methods to improve the existing results.

The SPECT dataset (*SPECT*) consists of 22 binary attributes that describe the partial diagnoses. There are 267 patients in the dataset. It was divided into the training data of 80 patients and testing data of 187 patients. Detailed description is shown in the Table 1.

In this investigation we applied the ensemble classifier method to the SPECT data and achieved accuracy of 90.4% for the ensemble of four classifiers. The gained 4% accuracy reduces the previous error rate by 31%, which is a significant and essential improvement for the entire medical diagnostic system.

3.2. Other experiments

Five well-known datasets were used to test the developed ensemble of the CLIP4 generated classifiers:

1. Waveform dataset (*wav*) [3]
2. Contraceptive Method Choice dataset (*cmc*) (author: Tjen-Sien Lim)
3. Statlog Satellite Image dataset (*sat*) (author: Ashwin Srinivasan)

4. Wisconsin Breast Cancer dataset (bcw) (author: Dr. W.H. Wolberg)
5. Boston Housing dataset (bos) (author: D. Harrison and D.L. Rubinfeld)

The experimental results for the datasets 2,3,4 and 5 were published in [21]. The authors compared 33 classification algorithms on different datasets. In this paper, the intervals of accuracy achieved by these algorithms are shown in Table 3, after [21].
The detailed description of the datasets is shown in the Table 1.

Table 1. Major properties and number of generated rules for the datasets considered in the experimentation.

Dataset	# class	# examples	# train / test examples	# attr.	# rules for different # classifiers in the ensemble							
					2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
bcw	2	699	10 CV	11	15.9 (mean value for ten-fold CV and 4 classifiers)							
bos	3	506	10 CV	13	42.7 (mean value for ten-fold CV and 4 classifiers)							
cmc	3	1473	10 CV	10	31.9 (mean value for ten-fold CV and 4 classifiers)							
sat	6	6435	3435 / 2000	37	125	185	245	307	371	431	492	562
SPECT	2	267	80 / 187	22	2	3	4	5	6	7	9	10
wav	3	3600	600 / 3000	22	18	27	36	45	54	63	72	81

An inductive program called C5.0 [14] was used for comparing the results achieved by the CLIP4 algorithm. C5.0 supports boosting [26]. Boosting is a technique for generating and combining multiple classifiers to give improved predictive accuracy. Unfortunately, boosting does not always help, especially when the training cases are noisy. Boosting cannot be directly applied to the CLIP4 algorithm because it requires to account for the weights attached to the examples, and replication of the training examples would increase the size of the training data, and thus slow down the learning process.
Because most of real datasets cover only a small portion of the state space, we did not consider bagging, which is a different approach to generate ensemble of classifiers. Bagging selects randomly, with replacement, a subset of examples from the original training data and generates classifiers using the subset. Thus, bagging was not considered since selecting a subset from the data that already describes/fills only a small part of the state space would weaken the generated classifiers, if compared to classifiers generated using the entire data set.

3.3. Analysis of the results

The t-statistics test [27] was used for comparison of learning schemes. The t-statistics test is used to determine if the difference in accuracy between two classifiers on a single dataset is statistically significant. The 5% confidence level test was performed, thus for experiments involving 10CV the confidence limit was 1.83, and for experiments with different number of classifiers the confidence limit was 1.94. If the t value is greater then the confidence limit then there is a statistically significant difference between tested classifiers.
The results of all experiments are shown in the Table 2. For the *SPECT, sat* and *wav* datasets experiments involving generation of ensemble of different number of classifiers (from 2 to 9 classifiers) were performed. The results in terms of the gained accuracy are summarized in Figure 4. The ensemble of classifiers generated by the CLIP4 algorithm gives significant improvement in the accuracy when compared to a single classifier generated by the same algorithm. The error rates for *wav* dataset decreased on average by about 17%, for *sat* dataset by about 19%, and for *SPECT* dataset by about 19% when comparing to a single classifier. The C5.0 algorithm with boosting, compared to a single classifier, achieved the same level of improvements for *sat* and *wav* datasets as CLIP4, but for the *SPECT* dataset application of boosting caused decrease of accuracy. For the *cmc, bcw,* and *bos* datasets we use 4 classifiers ensemble and the ten-fold cross validation to show that the generalization abilities of the generated ensembles, or boosted decision tree, remain the same. In this case the CLIP4 error rates decreased 6% for *cmc* dataset, 10% for *bcw* dataset, and 17% for *bos* dataset. The C5.0 algorithm with boosting, compared to a single classifier, achieved only slight improvement of accuracy for *cmc* dataset, no improvement for *bcw* dataset, and decrease of accuracy for *bos* dataset.

In Figure 4 the gained accuracy as function of the number of classifiers in the ensemble for the CLIP4 algorithm is shown. When the number of CLIP4 classifiers in the ensemble increases the accuracy gain (defined as the difference between accuracy for an ensemble and a single classifier) also increases. After a big increase of accuracy gain for ensemble of 3 or 4 classifiers the accuracy gain value stabilizes. The increase of the accuracy gain is almost not significant for bigger number of classifiers in the ensemble. The *SPECT* dataset accuracy results behave slightly differently. The big jump of the accuracy gain for 4 classifiers ensemble can be explained by random nature of the approach. The quick saturation of gain for 6 or more classifiers in the ensemble can be explained as an upper limit of the accuracy that can be obtained on the dataset by the CLIP4 algorithm.

Table 2. T-statistics test comparison of accuracy between the CLIP4 ensemble classifiers method and C5.0 with boosting.

Dataset	Algorithm	**Accuracy** for different # of classifiers in ensemble (CLIP4) or boosting trials (C5.0)								**mean accuracy** [%]	max accur. [%]	**NO boost/ ensemble accuracy** [%]	**mean gained accuracy** [%]	**t-stat accuracy comparison** [value of t]
		2	3	4	5	6	7	8	9					
SPECT	CLIP4	87.2	87.7	90.4	88.8	89.3	89.3	88.8	88.2	**88.7**	90.4	**86.1**	**2.6**	**CLIP4 performs significantly better** [24.3]
	C5.0	74.3	75.4	73.3	73.3	72.2	73.3	73.3	73.3	**73.6**	75.4	**74.3**	**- 0.7**	
sat	CLIP4	81.5	82.9	82.9	83.3	83.7	83.1	83.7	84.0	**83.1**	84.0	**79.5**	**3.6**	**C5.0 performs significantly better** [-25.8]
	C5.0	85.9	88.6	88.3	88.2	89.4	89.5	89.3	89.7	**88.6**	89.7	**85.9**	**2.7**	
wav	CLIP4	76.5	78.3	78.8	78.9	79.5	79.2	79.4	79.7	**78.8**	78.8	**74.5**	**4.3**	**C5.0 performs somehow better** [-1.0]
	C5.0	74.5	78.1	78.4	79.4	80.8	80.8	80.9	81.0	**79.2**	81.0	**74.5**	**4.7**	

Dataset	**Algorithm**	**Accuracy** for 4 classifier in ensemble (CLIP4) or 4 trial of boosting trials (C5.0) for 10 cross-validation trials										**mean accur.** [%]	**NO boost/ ensemble mean ac. [%]**	**mean gained accuracy** [%]	**t-stat accuracy comparison** [value of t]
		1	2	3	4	5	6	7	8	9	10				
bos	CLIP4	76.5	68.6	72.6	76.5	78.4	79.6	70.6	81.6	78.4	72.3	**75.5**	**70.5**	**5.0**	**CLIP4 performs somehow better** [0.4]
	C5.0	80.4	70.6	66.7	68.6	74.5	85.7	62.7	89.8	66.7	80.9	**74.7**	**75.6**	**- 0.9**	
bcw	CLIP4	95.7	92.9	98.6	97.1	98.6	91.4	97.1	94.3	95.7	95.7	**95.7**	**95.2**	**0.5**	**CLIP4 performs somehow better** [1.6]
	C5.0	95.7	88.6	94.3	94.3	97.1	94.3	94.3	98.6	88.6	94.2	**94.0**	**94.0**	**0.0**	
cmc	CLIP4	49.3	49.3	51.4	55.4	48.0	42.6	48.7	51.4	50.7	48.2	**49.5**	**46.9**	**2.6**	**C5.0 performs somehow better** [-0.4]
	C5.0	51.4	53.4	46.6	52.7	50.7	45.9	54.7	53.4	40.5	51.5	**50.1**	**49.5**	**0.6**	

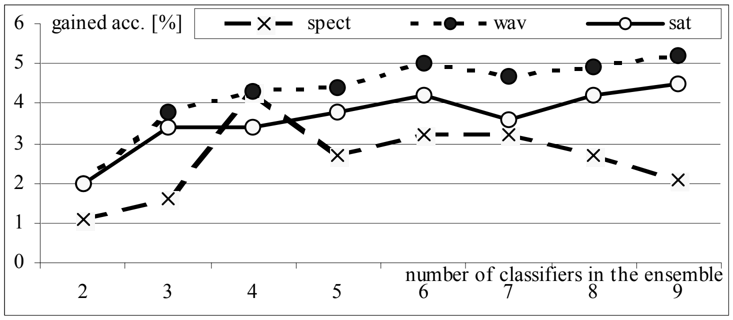

Fig. 4. Gained accuracy in the function of the number of classifier in the ensemble.

The performance of CLIP4 with ensemble and C5.0 with boosting is comparable. The CLIP4 performed better on *SPECT, bos, bcw*, while the C5.0 performed better on *sat, wav, cmc*. The important difference is when comparing the increase in accuracy between a single classifier and the ensemble in case of CLIP4 or C5.0 with boosting.
The CLIP4 with ensemble performed always better than a single classifier, as is shown in Table 3. It achieved an average decrease of error rates ranging from 6% for *cmc* dataset to 19% for the *SPECT* dataset. Thus, the ensemble of classifiers generated by the CLIP4 algorithm satisfies necessary and sufficient condition that states that an ensemble of classifiers should be more accurate than any of its individual members [16]. The main reason why the ensemble of classifiers generated by the CLIP4 algorithm performs better than a single classifier is statistical [13]. The statistical problem appears when the amount of available training data is too small in comparison with the size of the state space. Without sufficient amount of data a machine learning algorithm can generate many different classifiers that give the same accuracy on the training data (CLIP4 generated different classifiers by injecting randomness when solving the SC problems). Thus, by constructing an ensemble out of the individual classifiers the CLIP4 algorithm "averaged" their votes to reduce the risk of choosing the wrong classifier.
The C5.0 using boosting performed worse then a single classifier for *bos* and *SPECT* datasets, and the highest error rate decrease was 19% for *sat* dataset, as shown in Table 4. The results show that the proposed approach always helps to increase accuracy, in contrary to boosting that can perform worse than a single classifier. Figure 5 summarizes the difference in the results achieved when using an ensemble of classifiers for CLIP4 and boosted C5.0, against the results achieved by a single classifier.

Table 3. Comparision of accuracy between the CLIP4 ensemble of classifiers method and a single CLIP4 classifier.

Dataset	**CLIP4 with ensemble accuracy**. [%]	CLIP4 with ensemble max. accuracy [%]	**CLIP4 single classifier accuracy** [%]	**Gained accuracy** [%]	**Other results** Algorithm (accuracy [%])
SPECT	**88.7**	90.4	**86.1**	**2.6**	CLIP3*** (84)
sat	**83.1**	84.0	**79.5**	**3.6**	Loh, Lim, and Shih* (60÷90)
wav	**78.8**	78.8	**74.5**	**4.3**	Loh, Lim, and Shih* (52÷85), [Optimal Bayes (86), CART (72), NN (78)]**
bos	**75.5**	N/A	**70.5**	**5.0**	Loh, Lim, and Shih* (69÷78)
bcw	**95.7**	N/A	**95.2**	**0.5**	Loh, Lim, and Shih* (91÷97), CLIP3*** (92.4)
cmc	**49.5**	N/A	**46.9**	**2.6**	Loh, Lim, and Shih* (40÷57)

* taken from [21], ** taken from [3], *** taken from [19], CLIP4 with no ensemble results taken from [8]

Table 4. T-statistics test comparison of accuracy between the CLIP4 ensemble classifiers method and a single CLIP4 classifier, and between C5.0 with boosting and C5.0 with no boosting.

Dataset	**Algorithm**	**mean accuracy with ensemble / boosting** [%]	**NO boosting/ ensemble mean accuracy** [%]	**mean gained accuracy** [%]	**t-stat accuracy comparison** [value of t]
bos	CLIP4	**75.5**	**70.5**	**5.0**	CLIP4 with ensemble is **significantly better** [4.7]
	C5.0	**74.7**	**75.6**	**- 0.9**	C5.0 with boosting is **somehow worse** [-0.5]
bcw	CLIP4	**95.7**	**95.2**	**0.5**	CLIP4 with ensemble is **somehow better** [1.2]
	C5.0	**94.0**	**94.0**	**0.0**	C5.0 with boosting is **the same good** [0.0]
cmc	CLIP4	**49.5**	**46.9**	**2.6**	CLIP4 with ensemble is **significantly better** [2.7]
	C5.0	**50.1**	**49.5**	**0.6**	C5.0 with boosting is **somehow better** [0.5]
SPECT	CLIP4	**88.7**	**86.1**	**2.6**	CLIP4 with ensemble is **significantly better** [7.3]
	C5.0	**73.6**	**74.3**	**- 0.7**	C5.0 with boosting is **significantly worse** [-2.3]
sat	CLIP4	**83.1**	**79.5**	**3.6**	CLIP4 with ensemble is **significantly better** [13.3]
	C5.0	**88.6**	**85.9**	**2.7**	C5.0 with boosting is **significantly better** [6.2]
wav	CLIP4	**78.8**	**74.5**	**4.3**	CLIP4 with ensemble is **significantly better** [11.8]
	C5.0	**79.2**	**74.5**	**4.7**	C5.0 with boosting is **significantly better** [5.9]

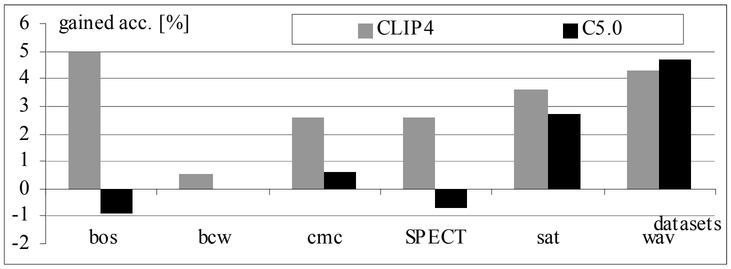

Fig. 5. Gained accuracy comparison of single classifier and ensemble of classifiers for CLIP4 and boosted C5.0.

We also investigated the overall goodness of the ensemble of classifiers in terms of the relation between the gained accuracy vs. the number of rules in the entire ensemble of classifiers for different numbers of generated classifiers. The goodness measure of the ensemble of classifiers was defined as a gained accuracy value divided by the number of rules in the ensemble. This value is then scaled to the value of one. The formula is:

$$goodness_{ensemble\ i} = \frac{gained\ accuracy_i}{number\ of\ rules_i} \cdot normalization\ factor$$

The goodness of the ensemble of classifiers generated by the CLIP4 algorithm for different datasets and different number of classifiers is summarized in Figure 6.

The optimal number of classifiers in the ensemble is 3 or 4. The peak at this value represents the best "trade-off" between gained accuracy and the size of the ensemble of classifiers.

If the user is interested in maximum accuracy gain then the bigger the number of classifiers in the ensemble the better the result. However, the accuracy gain value increases up to the certain point and then it saturates. After that value adding more classifiers will not significantly improve the accuracy, instead it will make the ensemble unnecessary complex.

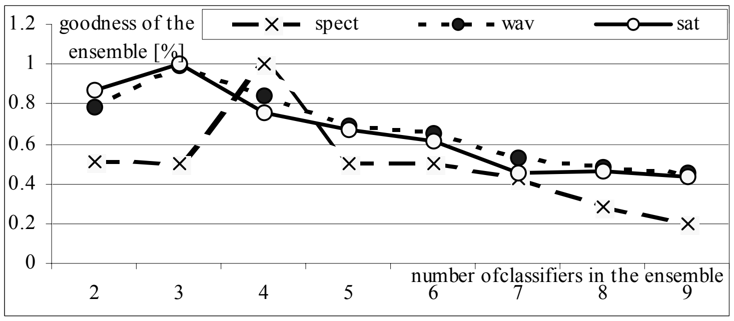

Fig. 6. Goodness of the ensemble as function of the number of classifiers in the ensemble.

4. SUMMARY AND CONCLUSIONS

In this paper we presented the new method for improving the accuracy of inductive machine learning algorithm CLIP4. An ensemble of classifiers was generated by injecting the randomness into the CLIP4 algorithm. The classifiers were combined using weighted voting scheme. As the result significant improvement in the accuracy of the CLIP4 algorithm was achieved. The test results show that the average achieved gain of accuracy was about 3.1%, which corresponds to about 15% decrease of the error rate when compared to the accuracy achieved by a single classifier. Because the CLIP4 algorithm generates strong classifiers in the first place the achieved improvement is significant. The results also show that the increase of accuracy for the CLIP4 ensemble of classifiers is achieved for all datasets, in contrary to boosting technique that can perform worse then a single classifier. Testing results show that a significant gain of accuracy is achieved for the ensemble of 3 to 4 classifiers. This fact stresses the usefulness of the proposed method. Namely, we can achieve a significant increase of the accuracy at the cost of generating only 3 to 4 classifiers instead of a single classifier.

5. REFERENCES

1. Ali, K. M., and Pazzani, M. J., Error reduction through learning multiple decisions, *Machine Learning*, 24 (3), pp. 173-202, 1996.
2. Bellmore, M., and Ratliff, H. D., Set Covering and Involuntary Bases, *Management Science*, 18 (3), 1971.
3. Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J., *Classification and Regression Trees*, Wadsworth International Group: Belmont, California, pp. 43-55, 1984.
4. Cios, K. J., and Liu, N., An algorithm which learns multiple covers via integer linear programming. Part I - The CLILP2 Algorithm, *Kybernetes*, 24 (2), pp. 29-50, 1995.
5. Cios, K. J., and Liu, N., An algorithm which learns multiple covers via integer linear programming. Part II – experimental results and conclusions, *Kybernetes*, 24 (3), pp. 24-36, 1995.
6. Cios, K. J., Wedding, D. K., and Liu, N., CLIP3: cover learning using integer programming. *Kybernetes*, 26 (4-5), pp. 513-536 (The Norbert Wiener 1997 Outstanding Paper Award), 1997.
7. Cios, K. J., Pedrycz, W., and Swiniarski, R.: *Data Mining Methods for Knowledge Discovery*, Kluwer, 1998.
8. Cios, K.J., & Kurgan, L., Hybrid Inductive Machine Learning Algorithm that Generates Inequality Rules, *Information Sciences*, Special Issue on *Soft Computing Data Mining*, in review, 2002.
9. Cios K. J. & Kurgan L., Hybrid Inductive Machine Learning: An Overview of CLIP Algorithms, In: Jain L.C., and Kacprzyk J. (Eds.) *New Learning Paradigms in Soft Computing*, Physica-Verlag (Springer), pp. 276-322, 2001.
10. Chvatal, V., A Greedy-Heuristic for The Set-Covering Problem, *Math. Operations Research*, 4 (3), 1979.
11. Dietterich, T. G., Machine Learning Research: Four Current Directions, *AI Magazine*, 18 (4), pp. 97-136, 1997.
12. Dietterich, T. G., and Kong, E. B., Machine Learning bias, statistical bias, and statistical variance in decision tree algorithms (Tech Report), Department of Computer Science, Oregon State University, Corvallis, Oregon, 1995.
13. Dietterich, T. G., Ensemble methods in machine learning. In Kittler, J., Roli, F. (Eds.) *Proceedings to First International Workshop on Multiple Classifier Systems*, Lecture Notes in Computer Science, vol. 1857, Springer-Verlag, Germany, 2000.
14. Data Mining Tools, http://www.rulequest.com/see5-info.html, 2000.
15. Grafinkel, R. S., and Nembauser, G. L., *Integer Programming*, John Wiley and Sons, New York, 1972.

16. Hansen, L., and Salamon, P., Neural Network Ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12, pp. 993-1001, 1990.
17. Hochbaum, D. .S., Approximation Algorithm for The Weighted Set-Covering and Vertex Cover Problems, *Siam J. Comput*., 11 (3), 1982.
18. Kolen, J. F., and Pollack, J.B., Back propagation is sensitive to initial conditions, *Advances in Neural Information Processing Systems*, 3, pp. 860-867, San Francisco, CA. Morgan Kaufmann, 1991.
19. Kurgan, L.A., Cios, K.J., Tadeusiewicz, R., Ogiela, M. & Goodenday, L.S., Knowledge Discovery Approach to Automated Cardiac SPECT Diagnosis, *Artificial Intelligence in Medicine*, 23 (2), pp.149-169, 2001.
20. Kwok, S. W., and Carter, C., Multiple decision trees. In Schachter, R. D., Levitt, T. S., Kannal, L. N., and Lemmer, J. F. (Eds.), *Uncertainty in Artificial Intelligence,* 4, pp. 327-335, Elsevier Science, Amsterdam, 1990.
21. Loh, W. Y., Lim, T. S., and Shih, Y. S., A comparison of prediction accuracy, complexity, and training time of thirty-three old and new classification algorithms, *Machine Learning*, 40, pp. 203-228, 2000.
22. Michalski, R. S., Discovering classification rules using variable-valued logic system VL1, *Proceedings of the Third International Joint Conference on Artificial Intelligence*, pp. 162-172, 1973.
23. Michalski, R. S., A theory and methodology of inductive learning. In R. Michalski., J. Carbonell, and T. M. Mitchell (Eds.), *Machine Learning*. Tioga Press, 1983.
24. Parmanto, B., Munro, P. W., and Doyle, H. R., Improving committee diagnosis with resampling techniques, In Touretzky, D. S., Mozer, M. C., and Hesselmo, M. E. (Eds.), *Advances in Neural Information Processing Systems*, 8, pp. 882-888, Cambridge, MA. MIT Press, 1996.
25. Ravindran, A., Phillips, D.T., and Solberg, J.J., *Operations Research, Principles and Practice*, second addition, pp. 1-69, John Wiley and Sons, 1987.
26. Schapire, R. E.: A brief introduction to boosting**.** *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pp. 1401-1406, 1999.
27. Witten I.H., Frank E.: *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation*, Morgan Kaufmann, San Francisco, 2000.