Applying Bagging Techniques to the SA Tabu Miner Rule Induction Algorithm

Chapter · January 2010

DOI: 10.1007/978-3-642-10781-8_9

2 authors, including:

Ivan Chorbev
Ss. Cyril and Methodius University in Skopje
115 PUBLICATIONS **878** CITATIONS

SEE PROFILE

Applying Bagging Techniques to the SA Tabu Miner Rule Induction Algorithm

Ivan Chorbev[1], Mirjana Andovska[2],

[1]Faculty of Electrical Engineering and Information Technologies,
PO Box 574, MK-1001 Skopje, Republic of Macedonia
ivan@feit.ukim.edu.mk
[2]Netcetera. ul. Partizanski Odredi 72a, MK-1000 Skopje, Republic of Macedonia
m.andovska@gmail.com

Abstract. This paper presents an implementation of bagging techniques over the heuristic algorithm for induction of classification rules called SA Tabu Miner. The goal was to achieve better predictive accuracy of the derived classification rules. Bagging (Bootstrap aggregating) is an ensemble method that has attracted a lot of attention, both experimentally, since it behaves well on noisy datasets, and theoretically, because of its simplicity. It is directly related to bootstrap sampling, since it uses the bootstrap samples to train multiple predictors. The outputs of the predictors are then combined by various voting strategies. Bootstrap is a good solution when it is impossible, or too expensive, to get multiple samples. In this paper we present the experimental results of various bagging versions of the SA Tabu Miner algorithm. The SA Tabu Miner [1] algorithm is inspired by both research on heuristic optimization algorithms (Simulated Annealing and Tabu Search based Data Miner) and rule induction data mining concepts and principles. The algorithm creates rules incrementally, performing a sequential process to discover a list of classification rules covering as many as possible training cases with as big quality as possible. It uses a combination of Simulated Annealing and Tabu Search to perform the search for the optimal classification rule. Several bootstrap methodologies were applied to SA Tabu Miner, including reducing repetition of instances, forcing repetition of instances not to exceed two, using different percentages of the original basic training set. Various experimental approaches and parameters yielded different results on the compared datasets. In the paper we discuss the results and conclude that the best improvement in predictive accuracy was achieved by using only 10 voting classifiers derived from 90% of the basic training dataset.

Keywords: Bagging, Bootstrap, Simulated Annealing, SA Tabu Miner, Tabu Search, Data Mining, Rule Induction

1 Introduction

The SA Tabu Miner (Simulated Annealing and Tabu Search based Data Miner) is developed to perform the classification task of data mining in an integrated medical

expert system. It is a rule induction algorithm which creates rules incrementally, performing a sequential process to discover a list of classification rules to cover as many training cases as possible with the highest quality. It uses a combination of Simulated Annealing and Tabu Search to perform the search for the optimal classification rule. It has been compared with CN2 and Ant miner algorithms on public domain data sets. The results showed that it obtained similar and often better results than the other approaches [1].

A major concern with a rule induction algorithm is how to improve its predictability. One possible solution is to use ensemble methods [2, 3, 4]. The ensemble methods were designed to improve the predictive accuracy of a various learning algorithms. Given a set *S* of training examples, a learning algorithm outputs a classifier, which is a hypothesis about the true function *f*. An ensemble of classifiers is a set of individually trained classifiers whose predictions are combined in some way (typically by weighted or unweighted voting) to classify new examples. The resulting classifier is often much more accurate than any of the single classifiers consisted in the ensemble. Both theoretical [23, 24] and empirical [25, 26, 27] studies have demonstrated that a good ensemble is one where the individual classifiers in the ensemble are both accurate and diverse. An accurate classifier is one that has an error rate better than random guessing on new *x* values; and two classifiers are diverse if they make different errors on different parts of the input space.

Many methods for constructing ensembles have been developed [3]. One group of methods manipulates the training examples by using re-sampling techniques. These methods include bagging [3], boosting [7] and cross-validation committees [11]. Boosting generates the classifiers sequentially, increasing the weights of the misclassified examples which are provided as input to the next classifier, while the other two methods generate the classifiers in parallel. In this paper, we focus our attention towards the bagging method introduced by Brieman [5]. It uses bootstrap sampling, which is sampling with replacement, to manipulate the input data in order to get several different versions of learning sets. Then, the learning algorithm is applied to the learning sets in order to obtain a set of diverse classifiers. Finally, the resulting multiple classifiers are combined by majority voting to form a composite classifier. The classifier trained on trail *i* is denoted as C_i, while C^* is the bagged (the final, composite) classifier.

At the time of its invention, only heuristic arguments were presented to explain why bagging would work. Breiman [3] presented empirical evidence that bagging can indeed reduce prediction error. He applied bagging over seven medium sized datasets and reported results that when the number of bootstrap replicates *T* is set at 50, the average error of the bagged classifier C^* ranges between 0.57 and 0.94 of the corresponding error when a single classifier is learned. In context of the number of bootstrap replicates, he examined 10 to 100 replicates and suggested that fewer replicates should be involved when the output is numerical. He also introduced the concept of an order-correct classifier-learning system as one which, over many training sets, tends to predict the correct class of the test instance more frequently than any other class. He explained that aggregating classifiers produced by an order-correct learner results in an optimal classifier, even though a single order-correct learner may not produce the optimal classifier. He also demonstrated that bagging is most effective for cart trees, due to their instability. Accordingly, he showed that

bagging works especially well for unstable learning algorithms – algorithms where with small changes in the training set the resulting classifier experiences large changes in its prediction. Decision-tree, neural network, and rule learning algorithms are all unstable. Linear regression, nearest neighbor, and linear threshold algorithms are generally very stable [3].

Recently bagging has attracted a lot of attention, both experimentally, since it behaves well on noisy datasets, and theoretically, due to its simplicity and also due to the popularity of the bootstrap methodology, which is an essential characteristic of bagging. Consequently, new researches have been conducted to explain the potential influence of training examples in the prediction process. In [9] experimental evidence is presented to support the hypothesis that bagging stabilizes prediction by equalizing the influence of training examples. Hence, influence equalization is mainly due to the absence of influential examples in 37% of bootstrap samples.

On the other hand, Buja and Stuetzie [18] observed bagging U-statistics and reported nearly identical results when using bootstrap sampling with or without replacement, in terms of bias, variance, and MSE, for resample size $M_{w/o}$ if $N/M_{with} = N/M_{w/o} - 1 = g$ (>0, $<\infty$), where N is the size of the original training set, and M_{with} and $M_{w/o}$ the resample size of the training set when using re-sampling with and without, respectively. Hall and Samworth [19] study the properties of bagged nearest neighbor classifiers and address how performance depends on the resample size. It is interesting to note that the benefits of the random sampling could be applied to other ensemble algorithms. For example in [10], Friedman proposes to incorporate uniform random sub-sampling, at each stage of the gradient boosting algorithm. Buhlmann and Yu [13] reported that bagging is unnecessary for MARS (multivariate adaptive regression splines), but works for low-dimensional predictors, such as stumps. Furthermore they conclude that subbagging (a subsample aggregating) is as accurate as bagging, but computationally cheaper. More on the effectiveness of other resample schemes can be found in the studies of [15, 13, 14, 22] and can be understood in the perspective of creating diverse classifiers.

Bagging and its variants have been applied to enhance many learning techniques such as decision trees, Bayesian classifier or discriminat function [3, 5, 12]. Its application in rule induction algorithms is seldom [6, 8]. Stefanowski [6] reported that bagging substantially improved the predictive accuracy of the MODLEM rule induction algorithm. Our goal was to examine the application of bagging to the SA Tabu algorithm, also an algorithm inducing decision rules. We tested whether the bagged classifier trained by the SA Tabu Miner could achieve a better predictive accuracy than the single SA Tabu Miner algorithm. Moreover, as a promising variant of bagging we analyzed the subagging (subsample aggregating) method. We choose the subsample size m to be a fraction $m = a*N$ with $0 < a < 1$ and N is the size of the original training set. In addition, we also tried a modified version of the bootstrapping technique, where the repetition of instances was forced not to exceed two.

The paper is organized as follows: in the next section we begin with a brief description of the SA Tabu algorithm. In section 3, we describe the ensemble methods, focusing on the bagging approach and its implementation over the rule induction SA Tabu Miner algorithm. Section 4 will cover our experimental results. The discussion of these results and conclusions are presented in the final section.

2 The SA Tabu Miner Algorithm

In this section we describe the rule induction algorithm called SA Tabu Miner (Simulated Annealing and Tabu Search based Data Miner). The goal of the algorithm is induction of classification rules. The hybrid heuristic algorithms Simulated Annealing (SA) [Kirkpatrick 1982] and Short-term Tabu Search (TS) [28, 29, 30, 31] are used in the development process of this classification algorithm. It incrementally constructs and modifies a solution - a classification rule of the form:

IF < term1 AND term2 AND ...> THEN <class>

Each term is a triple <attribute, operator =, value>. Since the operator element in the triple is always "=", continuous (real-valued) attributes are discretized in a preprocessing step using the C4.5 algorithm [16]. Each attribute can occur only once in each rule to avoid invalid rules such as "IF (Sex = male) AND (Sex = female) ...".

A high level description of the SA Tabu Miner algorithm is shown in Algorithm 1. The algorithm creates rules incrementally, performing a sequential process to discover a list of classification rules covering as many as possible training cases with as big quality as possible. It uses a combination of Simulated Annealing and Tabu Search to perform the search for the optimal rule.

```
TrainingSet = {all training cases};
DiscoveredRuleList = [ ];/*init. with an empty list*/
While (TrainingSet > Max_uncovered_cases)
    Calculate entropy and hence probability
    Start with an initial feasible solution S ∈ Ω.
    Initialize temperature
    While (temperature > MinTemp)
          Generate neighborhood solutions V* ∈ N(S).
          Update tabu timeouts of recently used terms
          Sort by (quality/tabu) desc sol-s S*∈ V*
          S* = the first solution ∈ V*
          While(move is not accepted or V* is exhaust.)
          If metrop(Quality(S) - Quality(S*)) then
                Accept move and update best solution.
                Update tabu timeout of the used term
                break while
          End if
          S* = next solution ∈ V*
          End while
          Decrease temperature
    End While
    Prune rule S
    Add discovered rule S in DiscoveredRuleList
    TrainingSet = TrainingSet - {cases covered by S};
End while
```

Fig. 1. SA Tabu Miner, the rule induction algorithm.

Where
(i) Ω is the set of feasible solutions,
(ii) S is the current solution,
(iii) $S*$ is the best admissible solution,
(iv) *Quality*(S) is the objective function,
(v) *N(S)* is the neighborhood of solution S,
(vi) $V*$ is the sample of neighborhood solutions.

At the beginning, the list of discovered rules is empty and the training set consists of all the training cases. Each iteration of the outer WHILE loop of SA Tabu Miner, corresponding to a number of executions of the inner WHILE loop, discovers one classification rule. After the rule is completed, it is pruned from the excessive terms, to exclude terms that were wrongfully added in the construction process. The created rule is added to the list of discovered rules, and the training cases covered by this rule are removed from the training set. This process is iteratively performed while the following condition is satisfied

```
number of uncovered train cases > Max_uncovered_cases
```

where *Max_uncovered_cases* is a user-specified number of the unclassified training samples, usually 5% of all cases.

The selection of the term to be added to the current partial rule depends on both a problem-dependent heuristic function (entropy based probability), a tabu timeout for the recently used attribute values and the metropolitan probability function based on the Boltzman distribution of probability. The algorithm keeps adding one term at a time to its partial rule until one of the following two stopping criteria is met:

• Any term to be added to the rule would make the rule cover a number of cases smaller than a user specified threshold, called *Min_cases_per_rule*.

• The control parameter "temperature" has reached its lowest value.

Every time an attribute value is used in a term added to the rule, its tabu timeout is reset to the number of values of the particular attribute. In the same time, all other tabu timeouts of the other values for the particular attribute are decreased. This is done to enforce the use of various values rather than the most probable one, since often the difference in probability between the most probable one and the others is insignificant. Therefore the final solution might not include the most probable values in the rule terms, but a combination of less probable ones.

In some cases, when two or more attribute values have close probabilities which are significantly greater than the probabilities of the other attributes, a dead lock race starts among the algorithms. In the next iterations all attributes with close probabilities are switching places and this continues until the minimal SA temperature is reached. In order to avoid this behavior, there is a tabu list for preventing dead lock cycles, which contains recently appeared qualities of proposed solutions and their appearing frequency. If a solution from the tabu list appears given number of times

(*DeadLockTimes*), then the iteration cycle ends, giving the best found rule as a solution.

The entropy based probability guides and intensifies the search into promising areas (attribute values that have more significance in the classification), therefore intensifying the search. The tabu timeouts of the recently used attribute values discourage their repeated use, therefore diversifying the search. The metropolitan probability function controls the search, allowing greater diversification and a broader search at the beginning, while the control parameter temperature is big, and later in the process, when the control parameter temperature is low, it intensifies the search only in promising regions.

The appropriate initial temperature needs to be high enough for all conditions to have equal chances to be visited. However, it should not be too high, as necessary searches are then done. The initial proposition rule starts using the most influential value of the most influential attribute in the training set. The usage probability of the selected attribute f_{best} is calculated and this probability is set as probability of the best solution. For the worst solution the probability is set to zero f_{worst}=0. If we presume that the probability of choosing the worst solution $Pr \in (0, 1)$ over the best solution is 0.5, then the initial temperature T_0 is

$$T_0 = - (f_{worst} - f_{best})/ln\ (Pr).$$

According to the metropolitan probability function, when there are several possibilities, simulated thermodynamic system changes its state from energy E_1 to E_2 with probability

$$P_T(\Delta E(X)) = \begin{cases} 1, \Delta E(X) <= 0 \\ \exp(-\dfrac{\Delta E(X)}{T}), \text{otherwise} \end{cases}$$

Where $\Delta E = E_2 - E_1$, and T is the control temperature parameter.

3 Bagging

Ensemble methods are widely and effectively used in machine learning to improve the predictive accuracy of a given learning algorithm. The basic idea behind these methods is to generate an ensemble of classifiers from some base training set and then combine their individual decisions into a final decision, constructing a final classifier.

Weaknesses of the ensemble methods, due to the process of combining classifiers are:

- increased memory space
- increased computation
- decreased comprehensibility of the final classifier.

The first weakness, the increased memory space, originates from the requirement that all subclassifiers (members of the ensemble) need to be stored after the training. The total used memory depends on the size of each subclassifier itself and the size of

the ensemble. Regarding this weakness, the SA Tabu Miner does not need much memory, since it produces classifier rules that use less memory.

The second weakness, the increased computation is due to two aspects. The first one addresses the training process; while training the ensemble subclassifiers, the training algorithm needs to be executed many times. The second aspect is about the classification process; in order to classify an input query, the answers of all subclassifiers need to be gathered, and afterwards the final decision is made. The training process in the SA Tabu Miner executes once and it repeats in longer time intervals when new training data arrives. Additionally, the classification process executes exceptionally quick, due to the fact that operations on classification rules use insignificant computation.

The third weakness, the decrease of comprehensibility of the classifier, is the most important weakness of the ensemble methods when they are used on rule induction algorithms. Since the SA Tabu Miner algorithm is used in a medical expert system, the expressiveness and human-readability of the classification rules is of great importance. Combining only ten groups of rules, similar with each other, a better predictive accuracy is achieved, and still the human-readability and simplicity of the classifier are preserved. In order to get any advantage from using multiple learners, it is necessary that the group exhibits a sort of heterogeneity or diversity, that is, the group classifiers produce different predictions. This allows individual errors to be compensated and a better prediction to be reached. Several diversity creation methods have been explored in the literature [4], including manipulation of the architecture of the learners, modification of the objective function each learner optimizes and sampling methods both on the training patterns and the training features.

Most widely investigated methods of ensemble training are the ones that manipulate the training data, where different ensemble members can be given different parts of this set, so they will hopefully learn different things about the same task. Bagging (derived from *b*ootstrap *agg*regat*ing*) approach belongs to the aforementioned methods. From a training set of size n $S = \{x_i \in X|_i = 1, \ldots, n\}$, a bootstrap sample (a new training set S_j) is created by drawing with replacement n examples, using the uniform distribution $P(x_i) = 1/n \ \ \forall i$. In general, T bootstrap samples are built in the sampling process, and T different classifiers are generated based on these bootstrap samples. The output of all T subclassifiers is combined into a final classifier using various voting strategies. Since Bagging resamples the training set with replacement, some examples are represented multiple times while others are left out. For a training set with n examples, the probability of an example being selected at least once is $1 - (1 - 1/n)\ n$. For a large n, the formula can be simplified to 1 – 1/e. Each bootstrap sample contains, on the average, 63.2% unique examples from the training set. The classifier trained on trail i is denoted as C_i, while C^* is the bagged (the final, composite) classifier. For any instance x, $C_i(x)$ and $C^*(x)$ are the classes predicted by the classifiers C_i и C^*, respectively. To classify an instance x, a vote for class k is noted by every classifier for which $C_i(x) = k$, so that $C^*(x)$ is the class with the top most votes. The algorithm is shown on Figure 2.

(input S learning set; T number of bootstrap samples; LA learning algorithm output C∗ classifier)
begin

```
        for i = 1 to T do
        begin
                Si := bootstrap sample from S; {sample with replacement} Ci := LA(Si); {
                generate a sub-classifier }
        end; { end for }
        C∗(x) = arg maxy∈Kj T i=1
        (Ci (x) = y)  {the most often predicted class}
end
```

Fig. 2. The bagging algorithm

3.1. Bagging the SA Tabu Miner

In this section, we describe the application of the bagging method over the SA Tabu Miner algorithm. A detailed description of the bagged algorithm is shown in pseudo code in Figure 3. The training data is first resampled or subsampled and bootstrap samples are created. Then using each bootstrap sample, a classifier is created by the SA Tabu Miner algorithm. The final classifier is formed by performing a majority voting scheme over the generated classifiers, i.e. it is signed to the most often predicted class.

```
  DiscoveredClassifiersList = [ ];  /*initialized with an
empty list*/

  While (number of classifiers planned are not generated)
  TrainingSet = {Determined % of all training cases};
  DiscoveredRuleList = [ ];   /*initialized with an empty
list*/
  While (TrainingSet > Max_uncovered_cases)
  Calculate entropy and hence probability
  Start with an initial feasible solution S ∈ Ω.
  Initialize temperature
  While ((temperature > MinTemp) && (noDeadlock))
       Generate neighborhood solutions V* ∈ N(S).
       Update tabu timeouts of recently used terms
       Sort by (quality/tabu order) desc sol-s S* ∈ V*
       S* = the first solution ∈ V*
     While (move is not accepted or V* is exhausted)
          If  metrop(Quality(S) - Quality(S*)) then
             Accept move and update best solution.
             Update tabu timeout of the used term
              break while
          End  if
          S* = next solution ∈ V*
       End while
```

```
     Decrease temperature
End  While
Prune rule S
Add discovered rule S in DiscoveredRuleList
TrainingSet = TrainingSet - {cases covered by S};
End  While

Calculate DiscoveredRuleList predictive ability
Add DiscoveredRuleList in DiscoveredClassifiersList
End While
```

Where
(i) Ω is set of possible solutions,
(ii) S is current solution,
(iii) S∗ is the best found solution yet,
(iv) Quality(S) is cost function, which values the quality of the rule,
(v) N(S) is neighbor of the solution S,
(vi) V∗ is sample of the neighbor solutions.

Fig. 3. Bagged SA Tabu algorithm

4 Experimental results

In this paper we examined the application of bagging over the SA Tabu Miner algorithm, which is a rule induction algorithm, and checked whether the bagged classifier trained by the SA Tabu Miner could achieve better predictive accuracy than the single SA Tabu Miner algorithm. Moreover, we wanted to identify conditions under which the bagging method may improve the final prediction accuracy. Thus, we extended the bagging method with different resampling techniques:

- the resampling without replacement technique, where the subsample size m is $m = a*N$ with $0 < a < 1$ and N i is the size of the original training set and,
- the resampling with replication (bootsraping) technique, but the repetition of instances is forced to be not more than two.

The predictive accuracy is evaluated using the ten-fold cross validation technique [21]. The whole dataset is partitioned to 10 equal-sized blocks with similar class distributions, which are then grouped in a 9 to 1 ratio blocks. Each block is in turn used as a test set, while the classifier is trained over the remaining nine blocks. The whole process is executed 10 times. The final result is then averaged. An ensemble of T sub-classifiers is created over each fold. All subclassifiers (single classifiers inside an ensemble) are induced by the SA Tabu Miner algorithm. The maximal temperature in the algorithm is set to the product of the initial number of samples and the number of attributes

*MaxTemperature = InitialNumberOfSamples * NumberOfAttributes*

The aforementioned value has been shown to give best results in single usage of our algorithm in comparison with the other rule induction algorithms.

All the experiments were performed using seven public-domain datasets from the UCI (University of California at Irvine) machine learning repository [20]. A brief overview of the datasets features is given in Table 1.

Table 1. Brief description of datasets

Data Set	Number of examples (instances)	Number of attributes
Hepatitis	155	19
Haberman's Survival Data Set	306	3
Bupa liver disorders	345	7
New thyroid gland data	215	5
Wisconsin breast cancer	569	32
Ljubljana breast cancer	286	10
Echocardiogram	132	12

When creating the bootstrap samples we used resampling without replacement with different resample sizes; we tried *m* being {95%, 90%, 75%} of *n*, the original training set size. B

The number of bootstrap samples is an important parameter, not only for the predictive accuracy, but for computational considerations, as well. Quinlan obtained good results for small numbers such as 3, 7 and 10 [12], but Brieman tested bagging using 10 to 100 bootstrap samples and Fei Xia [17] reported best results using only 10 bootstraps. Therefore, we choose T, the number of classifiers inside the ensemble, to be 10 and 50. In addition, we also tried a modified version of the resampling with replacement (bootstrapping) technique, where the repetition of instances would not exceed two.

The results are given in Table 2. For each data set, the first column shows the predictive accuracy obtained by a single classifier, while the next two columns contain the results of the basic version of the composite bagged classifier for different numbers of bootstrap samples. The following column group holds the predictive accuracy of the bagging method where one instance may appear twice at most.

Table 2. Results of different bagging approaches

Data set	Single SA Tabu Miner	Bagging with different *T*		Bagging where repetition does not exceed two	
		T = 10	*T* = 50	*T* = 10	*T* = 50
Hepatitis	89.2	76	81.33	82.22	84
Haberman's Survival	74.86	74.49	74.65	72.33	73.5
Bupa liver disorders	67.7	62.65	62.06	66.67	58.82
New thyroid gland data	92.44	87.14	88.57	90.79	89.76
Breast cancer Wisconsin	90.3	87.16	89.4	89.55	89.78

Ljubljana breast cancer	65.1	74.29	71.79	72.14	72.32
Echocardiogram	54.4	51.67	52.5	52.08	50.83

It is evident from the results that the bagging method applied to SA Tabu Miner achieves certain improvements to the predictive accuracy. Best improvement is obtained for the Ljubljana breast cancer data set. However, there are cases where due to the small number of examples in the training set, bagging gives no improvement at all. For the Hepatitis and New thyroid gland data it even decreases the predictive accuracy.

Comparing the results obtained using plain resampling with repetition and resampling with repetition where the repetition of instances does not exceed two, we conclude that better accuracy is obtained when the instance repetition is limited.

In order to further analyze the effect of the bagging ensemble method, we conducted experiments with different sizes of the training data. When creating the bootstrap samples we used resampling without replacement with different resample sizes. We tried m being {95%, 90%, 75%} of N, the original training set size. Buja and Stuetzle have shown that bagging is beneficial for $M_{\text{with\_replacement}} > N/6$ and $M_{w/o} > N/7$, but optimal is $M_{\text{with\_replacement}} = N/3$ and $M_{w/o} = N/4$. The results are shown on Table 3.

Table 3. Bagging without repetition with different T and different m

Data set	$T = 10$ $m = 75\%$	$T = 50$ $m = 75\%$	$T = 10$ $m = 90\%$	$T = 50$ m = 90%	$T = 10$ m=95%
Hepatitis	82.67	82.67	81.33	79.33	82.67
Haberman's Survival	72.67	73.67	73	73	72.33
Bupa liver disorders	57.65	61.18	66.18	64.12	62.35
New thyroid gland data	91.67	90.48	90	91.9	90.48
Breast cancer Wisconsin	90.75	88.36	91.49	91.19	90.45
Ljubljana breast cancer	72.86	71.43	72.5	72.14	72.5
Echocardiogram	53.33	51.67	50.83	52.5	50.83

The results of the experiments in Table 3 illustrate that best achievements are gained when using 90% of the base training set, while the number of bootstrap samples, that is voting classifiers, is only 10.

By comparing the results in Table 2 and Table 3, it can be concluded that bagging and its extensions does not affect the predictive accuracy significantly in every data set. Additionally, due to the increased number of voting classifiers, there is a loss in the comprehensible, simple and interpretable structure of the generated rules. In the Table 4, for each dataset, the average number of rules is listed. As expected, the number of rules increases as the number of classifiers rise.

Table 4. Average number of rules for each dataset

Data set	Hepatitis	Haberman Survival	Bupa	New thyroid	Breast cancer Wiscon.	Ljublj. breast cancer	Echocard.
No bag	3.21	6.4	9.9	5.8	6.1	8.55	8.3
$T = 10$	7.92	7.53	12.91	10.35	12.75	7.2	9.05
$T = 50$	9.09	9.31	13.92	12.15	15.37	8.71	10.5
$T = 10$	8.59	7.68	11.74	9.74	11.87	7	9.26
$T = 50$	9.38	8.66	13.13	11.71	14.07	7.69	10.76
$T = 10$ m=75%	8.3	7.62	11.54	9.58	12.24	6.91	9.7
$T = 50$ $m = 75\%$	9.72	8.881	14.23	11.69	14.34	7.8	10.45
$T = 10$ $m = 90\%$	8.41	7.44	12.31	9.41	11.54	6.62	9.22
$T = 50$ $m = 90\%$	9.29	8.82	13.87	11.64	14.04	7.73	10.99
$T = 10$ $m = 95\%$	8.55	7.16	11.64	9.14	11.95	6.56	9.79

5 Conslusion

This paper presented the implementation of bagging techniques to the SA Tabu Miner algorithm for classification rule induction. We have compared the performance of various version of bagging techniques applied to SA Tabu Miner on public domain data sets. The results showed that certain bagging approaches with particular parameters can significantly increase the predictive accuracy of the algorithm. However, in some fewer datasets, due to their statistical nature and the number of instances, bagging does not bring any benefits. Thanks to the resampling component, bagging can be useful in poor training sets, or when it is impossible, or too expensive, to get multiple samples. On the other hand, it cannot be applied in cases where there are small data sets, or the original sample is not a good approximation of the population, or there is dirty data.

Since comprehensibility is important whenever discovered knowledge will be used for supporting a decision made by a human user, SA Tabu Miner often discovered simpler rule lists. However, with the application of bagging the advantage of comprehensibility is lost for the sake of greater predictive ability. In some scenarios, when human verification of the discovered knowledge is not important and maximized predictive accuracy is the goal, the approaches proposed in the paper can be applied.

References

[1] Chorbev I., Mihajlov D., Jolevski I., Web Based Medical Expert System with a Self Training Heuristic Rule Induction Algorithm, Proc. of The First

International Conference on Advances in Databases, Knowledge, and Data Applications, DBKDA 2009, Cancun, Mexico, March 2009, page 143-148.
[2] Thomas G. Dietterich: Machine Learning Research: Four Current Directions. *AI Magazine*, **18** (4), 97–136 (1997).
[3] Thomas G. Dietterich, *Ensemble Methods in Machine Learning*, Oregon State University, Corvallis, Oregon, USA, tgd@cs.orst.edu, WWW home page: http://www.cs.orst.edu/tgd
[4] J. E. Gentle, W. Härdle, Y. Mori: Handbook of Computational Statistics, ISBN-10:3540404643. http/fedc.wiwi.hu.berlin.de/xplore/ebooks/html/csa, Springer, chp 16.
[5] L. Breiman: Bagging predictors. *Machine Learning*, **24** (2), 123–140 (1996).
[6] Jerzy Stefanowski, Bagging and Introduction of Decision Rules. In. Mieczyslaw Klopotek, et al.: *Intelligent information systems* (2002).
[7] Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In Machine Learning: *Proc. of the Thirteenth International Conference*, 148–156 (1996).
[8] Rafael Pino-Mejías, et al.: Bagging Classification Models with Reduced Bootstrap, In. Structural, Syntactic, and Statistical Pattern Recognition, pp 966–973 (2004) http://www.springerlink.com/content/6r0b2payc24fj93e/
[9] Yves Grandvalet: Bagging equalizes influence, *Machine Learning,* **55** (3) 251–270 (2004).
[10] J. Friedman. Stochastic gradient boosting. *Computational Statistics and Data Analysis*, **38** (4):367–378 (2002).
[11] B. Parmanto, P. Munro, H. Doyle: Improving Committee Diagnosis with Resampling Techniques. In: D. Touretzky, , M. Mozer, M. Hasselmo (Ed.): *Advances in Neural Information, Processing Systems,* **8**, 882–888 (1996).
[12] J. R. Quinlan: Bagging, boosting and C4.5. In: *Proceedings of the 13th National Conference on Artifitial Intelligence*, 725–730 (1996).
[13] P. Bühlmann, B. Yu: 2000, '*Explaining Bagging*'. Technical Report 92, Seminar für Statistik, ETH, Zürich.
[14] A. Buja, W. Stuetzle: T*he Effect of Bagging on Variance, Bias and Mean Squared Error'.* Technical report, AT&T Labs-Research (2000)
[15] J. H. Friedman, P. Hall: *On Bagging and Non-linear Estimation*. Technical report, Stanford University, Stanford, CA (2000).
[16] J. R. Quinlan: *C4.5: Programs for Machine Learning*, San Francisco, CA: Morgan Kaufmann (1993).
[17] Fei Xia, *Bagging*, http://faculty.washington.edu/fxia/courses/LING 572/pagging.ppt, (2006)
[18] Andreas Buja, Werner Stuetzle: Observation of Bagging, *Statistica Sinica*, **16**, 323–351 (2006).
[19] P. Hall, R. J. Samworth: Properties of Bagged Nearest-neighbor Classifiers, *J. Roy. Statist. Soc.*, Ser. B, **67**, 363–379 (2005).
[20] http://archive.ics.uci.edu/ml/
[21] S. M. Weiss: Small Sample Error Rate Estimation for k-neares Neighbor Classifiers, *IEEE, Transaction of pattern analysis and Machine Intelligent*, **13** (3) 285–289 (1991).

[22] Gavin Brown, Jeremy Wyatt, Rachel Harris, Xin Yao: Diversity Creation Methods: A Survey and Categorisation, *Information Fusion*, **6** (1), 5–20 (2005).
[23] L. Hansen, P. Salamon: Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 993–1001 (1990).
[24] A. Krogh, J. Vedelsby: Neural network ensembles, cross validation, and active learning. In: G. Tesauro, D. Touretzky, T. Leen: *Advances in Neural Information Processing Systems*, **7**, 231–238 Cambridge, MA. MIT Press (1995).
[25] S. Hashem: Optimal linear combinations of neural networks. *Neural Networks*, **10** (4), 599–614 (1997).
[26] D. Opitz, J. Shavlik: Actively searching for an effective neural-network ensemble, *Connection Science*, **8** (3/4), 337–353 (1996a).
[27] D. Opitz, J. Shavlik: Generating accurate and diverse members of a neural-network ensemble. In D. Touretsky, M. Mozer,, M. Hasselmo: *Advances in Neural Information Processing Systems*, Cambridge, MA. MIT Press. **8**, 535–541 (1996b).
[28] S. Kirkpatrick, C. D. Jr. Gelatt, M. P. Vecchi: Optimization by Simulated Annealing, In: J Thomas: Technical Report Research Report RC 9335, IBM Watson Center, Yorktown Heights, N.Y. (1982).
[29] H. Zhang, G. Sun: Feature selection using tabu search method, *Pattern Recognition*, **35** (3) 701–711 (2002).
[30]. S. M. Sait, H. Youssef: General Iterative Algorithms for Combinatorial Optimization, *IEEE Computer Society*, Los Alamitos, Calif, USA (1999).
[31]. F. Glover: Tabu search I, *ORSA Journal on Computing*, **1** (3) 190–206 (1989).