Tuning Hyperparameters of Classification Based on Associations (CBA)

Tomáš Kliegr[1] and Jaroslav Kuchař[2]

[1] Department of Information and Knowledge Engineering, Faculty of Informatics and Statistics
University of Economics, W. Churchill Sq. 1938/4, Prague 3, Czech Republic
tomas.kliegr@vse.cz
[2] Web Intelligence Research Group, Faculty of Information Technology
Czech Technical University in Prague, Thákurova 9, 160 00, Prague 6, Czech Republic
jaroslav.kuchar@fit.cvut.cz

Abstract: Classification models composed of crisp rules provide excellent explainability. The limitation of many conventional rule learning algorithms is the separate-and-conquer strategy, which may be slow on large data. Association Rule Classifiers (ARC) is an alternative approach that can be very fast on massive datasets but is highly susceptible to the correct choice of metaparameters. Most existing ARC algorithms use default thresholds of 50% for minimum confidence and 1% minimum support, which can result in excessively long rule generation or underperforming models. Due to the high-costs that can be associated with evaluation of single combination, it is impractical to use standard metaparameter optimization approaches. In this paper, we introduce two variant threshold tuning algorithms specifically designed for ARC. Evaluation on 22 standard UCI datasets shows promising results in terms of model size and accuracy in comparison with the default thresholds. The implementation of the proposed algorithms is made available in R packages rCBA and arc, which are available in the CRAN repository.

1 Introduction

Association rule classifiers (ARC) are formed by selecting a subset of rules from a high number of candidates, which are generated by association rule learning algorithms known for their excellent performance on big and sparse datasets. The large base of candidate rules or frequent itemsets provides opportunities for achieving a good balance between predictive performance and interpretability of the produced models.

An ARC algorithm has two fundamental steps: candidate generation, and building of a classifier by selecting a subset of the generated candidates. While most research has focused on the classifier building phase, the candidate generation phase has not received much attention. Most ARC algorithms including state-of-the-art approaches like Interpretable Decision Sets (IDS) [1], Scalable Bayesian Rule Lists (SBRL) [2], or Bayesian Rule Sets (BRS) [3] rely on simple heuristics for generating the candidates, such as step-wise increases in support threshold by 5% until a fixed desired number of candidate frequent itemsets is reached.

Candidate generation can fundamentally affect all facets of ARC models, including speed of model building, size of the generated models, and particularly the predictive performance. In this paper, we provide two alternative approaches to rule generation. We focus on approaches applicable to the rule generation step of the Classification-based on Associations (CBA) algorithm [4]. While there are newer approaches, CBA is still one of the best rule-based classification algorithms that concerns balance between comprehensibility of the model, predictive power and scalability [5].

The two tuning algorithms that we describe are based on different principles. The first approach is a heuristic, which aims to produce a user-set number of rules by varying minimum support, minimum confidence, and maximum antecedent length thresholds. The second approach is a supervised algorithm, in which each metaparameter setting is used to create a classifier. Next, it is evaluated through internal validation. As optimization algorithm we adopt simulated annealing.

This paper is organized as follows. Section 2 briefly introduces the CBA algorithm. Section 3 covers the two proposed threshold tuning algorithms. Section 4 presents evaluation and Section 5 summarizes limitations of the presented work and provides outlook for future extensions. The conclusions summarize the contributions of our proposal, briefly discussing possible applications.

2 Association Rule Classifiers

The first association rule classification algorithm was Classification based on Associations (CBA) [4]. While there were multiple follow-up algorithms providing marginal improvements in classification performance (e.g. CPAR [6], CMAR [7]), the structure of most ARC algorithms follows, with some deviations, that of CBA [8]:

1. learn classification association rules,
2. prune the set of rules,
3. classify new objects.

Rule learning In this phase, some algorithms such as CBA learn complete association rules of the form *antecedent* $\rightarrow$ *consequent*. The learning step returns all rules matching the minimum confidence and minimum support thresholds. The confidence of a rule is defined as $conf(r) = a/(a+b)$, where a is the number of correctly classified objects, i.e. those matching rule antecedent as well rule consequent, and b is the number of misclassified objects, i.e. those matching the antecedent, but not the consequent. The support of a rule is defined as $supp(r) = a/n$, where n is the number of all objects (relative support), or simply as a (absolute support). Additionally, the rule mining setup is constrained so that only the target class values can occur in the consequent of the rules.

In some newer methods, the first step involves generating frequent itemsets, rather than complete rules. An example of such method is IDS, which does not impose the minimum confidence threshold. It takes on the input already the result of frequent itemset mining (i.e. conjunctions of conditions). Rules are then formed within IDS by splitting the frequent itemset into an antecedent and consequent parts.

In both approaches, adaptations of standard frequent itemset generation association rule learning algorithms such as *apriori* [9] or FP-growth [10] are used.

Rule pruning What is performed during the pruning phase varies strongly from algorithm to algorithm. CBA uses a simple and fast heuristic, which first sorts the rules and then removes redundant rules. Rule is considered as redundant if it does not correctly classify any instances after instances covered by rules with higher priority. In contrast, the IDS algorithm uses computationally intensive submodular optimization, which provides guarantees in terms of the optimality of the selected subset of rules with respect to a chosen balance between predictive performance and interpretability.

Classification phase The way classification is performed depends primarily on whether the ARC algorithm produces rule lists or rule sets. Rule lists are ordered, and typically only the first matching rule in the rule list is used to classify an instance. CBA produces rule lists. In contrast, rule sets are unordered and typically all rules with matching antecedents contribute to classifying an instance. CPAR is an example of an algorithm that produces a rule set.

3 Automatic Tuning of Mining Parameters

The minimum support threshold is a mandatory hyperparameter of most, if not all, association rule learning approaches, yet even the latest algorithms do little to tune it algorithmically. Minimum confidence threshold is used in smaller number of algorithms, but when it is used, it is also not tuned. We suspect that the reason is that these thresholds are notoriously difficult to optimize due to exponential complexity of the search space [11]. Additionally, the classification performance is typically very sensitive to parameter setting. While generally lower values of confidence and support and higher values of rule length produce the best results, the side effect of such setting can be a disproportionally long time needed to build the classifier caused by a combinatorial explosion, and consequently extreme memory requirements.

We considered standard approaches such as pure random or grid search. Since they do not use any background knowledge of the algorithm, we found them to be unsuitable for optimizing the hyperparameters of association rule learning, because of the sudden steep increases in space state complexity that can be triggered by small changes in the value of the hyperparameter.

In the following, we introduce our two proposals for hyperparameter tuning for association rule classification.

3.1 Simulated Annealing Optimization

Algorithm 1: Automatic Parameter Tuning with Simulated Annealing.

```
input : Input dataset in a tabular form: X
output: CBA Classifier (a set of ordered rules): C
1  begin
2      repeat
3          bestSetting = currentSetting = randomSetting()
4          ruleSet = learnRuleset(X, currentSetting, TIME_LIMIT)
5      until ruleSet is not EMPTY
6      temp = INIT_TEMP // initialize with temperature >1
7      accBest = -1
8      while temp > 1 do
9          // compute accuracies using stratified split, status can
             either be timeout, success or empty rule set
10         (accCurrent, statusCurrent)=
             crossEvalAcc(currentSetting, X, TIME_LIMIT)
11         newSetting = perturbate(currentSetting, statusCurrent) //
             see Algorithm 2
12         (accNew, statusNew)= crossEvalAcc(newSetting, X,
             TIME_LIMIT)
13         // probability of acceptance for worse solution
14         Pr = min(1, exp(-(accNew - accCurrent))/temp)
15         if statusNew is success AND (accNew > accCurent OR
             Pr > rand(0, 1)) then
16             currentSetting = newSetting
17         if statusNew is success AND accNew > accBest then
18             (bestSetting, accBest) = (newSetting, accNew)
19             // remember best
20         temp = temp * (1-ALPHA) // cooling
21     C = learnRuleset(X, bestSetting)
22     return C
```

Algorithms 1 and 2 present our implementation for the hyperparameter optimization based on simulated annealing [12, 13, 14]. The objective criterion which is optimized against is the accuracy of the model.

The algorithm starts as a random search for one valid initial solution providing non-empty classifier. Each subsequent classifier is evaluated using nested cross-validation. Input data are internally divided into a train and a validation subset with a stratified split. The classifier is built with a setting generated on the train set. The accuracy is computed using the created classifier on the validation set. If the execution time of the evaluation is over a predefined threshold we stop the computation and mark the setting as invalid and set the computed accuracy to null.

Algorithm 2: Perturbate - Generating new setting for SA.

```
input  : Current setting: currentSetting
         Current setting status: resultStatus (timeout, success or
empty rule set)
output: New setting: newSetting
begin
    newSetting = currentSetting
    // with uniform probability select one parameter
    p = random("support", "confidence", "ruleLength")
    switch p do
        case "support" or "confidence" do
            if resultStatus is timeout then
                // increasing threshold can fasten execution
                newSetting[p] = newSetting[p] + rand(0, 1 -
                  newSetting[p])
            else
                if resultStatus is empty then
                    // threshold may have been too high
                    newSetting[p] = newSetting[p] - rand(0,
                      newSetting[p])
                else
                    newSetting[p] = random(0,1)
        case "ruleLength" do
            if resultStatus is timeout then
                // shorter rule length can fasten execution
                newSetting[p] = newSetting[p] - 1
            else
                newSetting[p] = rand(1, MAX_LENGTH)
    return newSetting
```

The evaluated new setting is accepted as a candidate for next iteration if: 1) it is a valid setting not leading to a timeout, 2) the accuracy is better than the current setting, or the computed probability of acceptance exceeds the random value. As an optimization we always remember the best solution found so far so that it can be used if the algorithm terminates at a sub-optimal place.

An important part of the algorithm is a generation of a new setting based on the previous one. Only one parameter is changed during generation of a new setting, which composes of support, confidence or rule length. If the current setting was labeled as invalid, the support or confidence are increased or rule length decreased to overcome long computation time and perform more restricted rule mining. If the setting does not generate any rule or no rule is applicable, the support or confidence are decreased. For remaining situations, a random value is generated.

3.2 Heuristic algorithm

As an alternative to the supervised evolutionary approach, we also introduce an unsupervised heuristic algorithm. While the search in the simulated annealing approach uses accuracy as objective function, the heuristic algorithm only aims to return a user-set number of rules. This approach is conceptually faster, since repeated evaluations of the classification model are not performed.

According to the recommendation in [4], CBA generates best results when the rule generation step returns at least 60.000 of rules. The experiments performed by [4] also provide recommended values for minimum confidence (50%) and support (1%) thresholds.

The problem that our *CBA-RG-auto* algorithm addresses is that on some datasets the combination of the values suggested in [4] fails. The principal reasons are either not enough rules generated or a combinatorial explosion generating high number of overly short (and thus general) rules.

The *CBA-RG-auto* algorithm (Alg. 3) takes on the input two principal parameters: the number of desired rules (*targetRuleCount*) and preferred time that can be spent with tuning (*totalTimeout*). The algorithm then iteratively refines the minimum support (*support*) and confidence (*conf*) thresholds. The mining time and risk of combinatorial explosion is controlled by adjusting the constraint on the minimum and maximum number of conditions that can appear in the antecedent of the rules (*minLen* and *maxLen*). To guide the search process, the algorithm takes on input several additional parameters. According to our experiments, their values can be typically left at their default values (we used the same defaults in all experiments reported in our evaluation).

4 Evaluation

In our benchmark, we aim to evaluate the performance of the two proposed tuning steps against CBA with default parameters as a baseline.

For simulated annealing, we report on two setups, one using default values of metaparameters of simulated annealing (denoted as *sa*). To investigate the effect of metaparameters introduced in the simulated annealing algorithm, we also involve approach denoted as *saopt*, which corresponds to the simulated annealing tuning algorithm with metaparameter values optimized with random search. For saopt the configurations were evaluated against test data to determine the upper bound of attainable accuracy. As a result, *saopt* cannot be directly compared with the remaining evaluated algorithms, which did not have access to test data during training.

Algorithm 3: Automatic parameter tuning with heuristic algorithm (CBA-RG-auto)

```
input      : train training data
parameters: main: targetRuleCount, totalTimeout,
            supplementary: initSupport = 0.01, initConf = 0.5,
            confStep = 0.05, suppStep = 0.05, minLen = 2, initMaxlen = 3,
            iterTimeout = 2, maxIterations = 40
output     : rules - list of rules to be used as input for CBA-CB
1  begin
2      startTime ← currentTime(), supp ← initSupport, conf ←
         initConf, maxLen ← initMaxlen, iterations ← 0,
         maxLenDecreasedDueToTIMEOUT ← false,
         lastRuleCount ← -1
3      MAXRULELEN ← number of explanatory attributes
4      while true do
5          iterations ← iterations + 1
6          if iterations = maxIterations then
7              break
8          rulesCur ←
             apriori(minLen,maxLen,supp,conf,iterTimeout)
9          if apriori not finished within iterTimeout then
10             if currentTime()-startTime > totalTimeout then
11                 break
12             else if maxLen > minLen then
13                 maxLen ← maxLen - 1
14                 maxLenDecreasedDueToTIMEOUT ← true
15             else
16                 break // All options exhausted
17         else
18             rules ← rulesCur
19             if rulecount ≥ targetRuleCount then
20                 break // Target rule count satisfied
21             else if currentTime() - startTime > totalTimeout
                 then
22                 break // Max execution time exceeded
23             else if maxLen < MAXRULELEN and
                 lastRuleCount != count(rules) and
                 (maxLenDecreasedDueToTIMEOUT = false)
                 then
24                 maxLen ← maxLen + 1
25                 lastRuleCount ← count(rules)
26             else if maxLen < MAXRULELEN and
                 maxLenDecreasedDueToTIMEOUT = true and
                 supp ≤ (1-suppStep) then
27                 supp ← supp + suppStep
28                 maxLen ← maxLen + 1
29                 lastRuleCount ← rulecount
30                 maxLenDecreasedDueToTIMEOUT ← false
31             else if conf > confStep then
32                 conf ← conf - confStep
33             else
34                 break // All options exhausted
35     return first targetRuleCount rules from rules
```

4.1 Setup

Datasets The evaluation was performed on 22 datasets selected from the UCI repository [15]. All selected datasets were previously used in evaluation of rule learning or decision tree algorithms in one of the following seminal papers: [5, 16, 4, 17]. Numerical attributes with more than 3 values were binned with entropy-based discretization [18]. Ten-fold crossvalidation was used to generate train-test splits. The same splits were used for all evaluated configurations.

Implementation We made available under an open source licence implementations of all evaluated algorithms. We used R package `rCBA`[1] (available via CRAN) to obtain results for the baseline CBA run and for simulated annealing. R package `arc`[2] (also available via CRAN) was used to obtain results for the heuristic algorithm. Both implementations use the apriori algorithm for the rule learning phase.

Settings The classifier building phase of CBA does not have any metaparameters. The rule learning phase requires setting of rule mining parameters – minimum support, minimum confidence and maximum rule length. The starting parameters for the proposed threshold tuning methods (Algorithms 1 – 3) are also listed below.

- *Baseline CBA (base)*: 50% minimum confidence, 1% minimum support, maximal rule length 3.
- *Heuristic algorithm (heuristic)*: Default setting is *targetRuleCount*=60000, *initSupport* = 0.01, *initConf* = 0.5, *confStep* = 0.05, *suppStep* = 0.05, *minLen* = 2, *initMaxlen* = 3, *iterTimeout* = 2, *maxIterations* = 40
- *Simulated Annealing (sa)*: Default setting for the SA algorithm is $INIT\_TEMP = 100.0$, $ALPHA = 0.05$, $MAX\_LENGTH = 5$, $TIME\_LIMIT = 10$
- *Optimized Simulated Annealing (saopt)*: Random search from the following intervals $INIT\_TEMP = 10.0 - 100.0$, $ALPHA = 0.01 - 0.5$, $MAX\_LENGTH = 3 - 10$, $TIME\_LIMIT = 1 - 10$

4.2 Results

Results are reported in terms of accuracy (Table 1), rule count (Table 2), average number of conditions in rules in the model (Table 3), average model size computed as *average number of conditions* × *average rule count* (Table 4), and classifier build time (Table 5). Finally, Table 6 provides for each of the evaluated approaches an aggregate number of wins in each of the five criteria above.

Baseline CBA The results show that CBA with default parameter values performs surprisingly well, achieving best results in terms of overall size of the classifier on most datasets (14 out of 22), while obtaining the best results on

[1]Version 0.4.3
[2]Version 1.2

5 datasets in terms of predictive performance. Remarkably, there are three datasets (breast-w, credit-g, sonar) for which the default parameter values generate models that have best accuracy and at the same time are smallest in terms of combined rule count and rule length.

Despite the five wins, base CBA had the worst average and median accuracy. Detailed examination of Table 1 shows that the default thresholds result in either very low accuracy or excessive size on several datasets, the drop in accuracy is particularly strong on glass and letter datasets. The instability of results is reflected by high standard deviation for accuracy.

Heuristic The optimization heuristic provides best outcome in terms of predictive performance, both in terms of accuracy and the number of wins against other datasets. This comes at a cost of creating larger models than generated by other methods, also the build time is the highest. One dataset (letter) was not even processed. For accuracy, the heuristic approach provides the most stable results with lowest standard deviation.

Simulated annealing When it comes to compact models, very promising results were obtained by simulated annealing with default parameters (sa), which produced the smallest models in terms of rule count on 12 datasets. In two cases (australian, hepatitis), this algorithm produced much smaller models than the other methods with a small gap in terms of accuracy. On the ionosphere dataset, sa even generated a model which was most accurate and at the same time smallest.

The saopt algorithm generated almost consistently better results than sa. However, this approach is not fully comparable with the remaining two, because it used the test set to select the best combination of hyperparameters. It is included to show a possible effect of tuning hyperparameters of simulated annealing as opposed to only using the default values.

5 Limitations and Future Work

We acknowledge several limitations affecting our preliminary study:

- Our benchmark did not account for the tradeoff between rule count and accuracy. For example, 1% improvement in accuracy may need to be offset by much higher increase in number of rules, which are required to cover various specialized cases.

- We have not performed statistical testing on significance in differences between the algorithms.

- The baseline approaches could include some previously proposed approaches for metaparameter optimization, such as [11].

Table 1: Accuracy

	base	heuristic	sa	saopt
anneal	0.953	0.991	0.874	0.874
australian	0.862	0.841	0.854	0.856
autos	0.410	0.716	0.574	0.601
breast-w	0.960	0.948	0.956	0.957
colic	0.739	0.792	0.831	0.843
credit-a	0.753	0.861	0.854	0.867
credit-g	0.743	0.727	0.737	0.740
diabetes	0.750	0.753	0.760	0.757
glass	0.429	0.718	0.658	0.672
heart-statlog	0.804	0.807	0.804	0.826
hepatitis	0.786	0.793	0.774	0.807
hypothyroid	0.972	0.990	0.933	0.951
ionosphere	0.886	0.917	0.929	0.931
iris	0.933	0.933	0.927	0.933
labor	0.773	0.840	0.743	0.790
letter	0.241	NaN	0.533	0.610
lymph	0.757	0.781	0.707	0.759
segment	0.932	0.820	0.942	0.942
sonar	0.746	0.740	0.707	0.741
spambase	0.929	0.851	0.932	0.932
vehicle	0.663	0.682	0.678	0.701
vowel	0.414	0.628	0.434	0.588
mean	0.747	0.816	0.779	0.803
sd	0.203	0.100	0.144	0.118
median	0.765	0.807	0.789	0.816

Table 2: Average rule count

	base	heuristic	sa	saopt
anneal	22.7	36.7	16.3	16.3
australian	21.9	89.3	8.0	16.2
autos	51.7	41.2	32.2	33.8
breast-w	30.2	56.6	24.4	32.6
colic	77.8	106.8	20.3	10.6
credit-a	42.1	119.2	22.9	23.8
credit-g	62.2	135.1	62.0	75.1
diabetes	21.8	63.9	24.6	26.4
glass	24.3	31.5	15.5	18.1
heart-statlog	12.9	54.8	16.1	22.7
hepatitis	22.1	32.0	9.7	13.0
hypothyroid	20.9	45.8	11.5	13.4
ionosphere	46.4	46.0	18.5	20.8
iris	7.1	5.9	4.6	3.9
labor	11.7	11.3	6.5	7.8
letter	41.3	NaN	1005.4	1347.4
lymph	26.8	37.1	18.7	14.3
segment	116.9	62.4	148.1	148.1
sonar	29.2	42.8	17.5	23.3
spambase	284.8	7.5	293.3	301.3
vehicle	70.0	86.3	88.7	105.8
vowel	50.2	144.4	58.4	122.0
mean	49.8	59.8	87.4	108.9
sd	58.4	39.9	215.0	285.0
median	29.7	46.0	19.5	23.0

- Since some datasets are imbalanced, the evaluation metrics should be complemented by appropriate measure, such as Cohen's Kappa. This measure could also be supported as fitness measure in the sa algorithm.

Table 3: Average antecedent length

	base	heuristic	sa	saopt
anneal	1.737	1.749	2.051	2.051
australian	1.899	3.759	1.532	2.146
autos	1.847	3.409	2.739	2.514
breast-w	1.893	2.857	2.100	2.613
colic	1.973	2.240	2.134	2.102
credit-a	1.952	4.653	2.293	1.973
credit-g	1.967	4.272	2.029	2.236
diabetes	1.823	3.120	1.987	2.040
glass	1.876	2.763	1.620	2.045
heart-statlog	1.841	4.054	2.518	3.018
hepatitis	1.836	2.646	1.501	1.975
hypothyroid	1.811	2.126	1.726	1.632
ionosphere	1.726	2.231	1.272	1.298
iris	1.326	1.290	1.008	0.730
labor	1.478	1.559	1.035	1.379
letter	1.901	NaN	2.158	2.711
lymph	1.854	3.181	1.669	1.770
segment	1.938	2.529	2.213	2.213
sonar	1.922	2.786	1.936	2.059
spambase	1.984	2.234	2.451	2.635
vehicle	1.970	1.988	2.754	3.269
vowel	1.931	2.031	2.080	3.702
mean	1.840	2.737	1.946	2.187
sd	0.160	0.903	0.486	0.660
median	1.885	2.646	2.040	2.080

Table 4: Model size

	base	heuristic	sa	saopt
anneal	3.017	3.059	4.206	4.206
australian	3.604	14.133	2.346	4.607
autos	3.411	11.619	7.500	6.319
breast-w	3.583	8.162	4.411	6.826
colic	3.892	5.016	4.555	4.417
credit-a	3.812	21.652	5.257	3.891
credit-g	3.871	18.252	4.119	4.999
diabetes	3.323	9.731	3.947	4.164
glass	3.520	7.632	2.623	4.183
heart-statlog	3.391	16.436	6.341	9.106
hepatitis	3.370	7.000	2.254	3.901
hypothyroid	3.279	4.520	2.979	2.664
ionosphere	2.980	4.976	1.617	1.686
iris	1.759	1.665	1.017	0.533
labor	2.185	2.431	1.072	1.901
letter	3.613	NaN	4.658	7.351
lymph	3.436	10.116	2.786	3.133
segment	3.757	6.395	4.896	4.896
sonar	3.693	7.761	3.749	4.239
spambase	3.937	4.991	6.008	6.945
vehicle	3.880	3.951	7.585	10.686
vowel	3.730	4.125	4.326	13.706
mean	3.411	8.268	4.012	5.198
sd	0.540	5.428	1.845	3.036
median	3.552	7.000	4.162	4.328

- For the baseline CBA algorithm, we evaluated only setting with maximum length of antecedent set to 3, as higher thresholds sometimes led to combinatorial explosion.

- The included datasets are of small or moderate size, evaluation on large datasets was not performed.

Table 5: Build time in seconds

	base	heuristic	sa	saopt
anneal	0.73	24.56	29.08	29.02
australian	0.48	101.88	10.76	34.08
autos	0.61	94.44	21.03	77.25
breast-w	0.48	21.11	13.01	19.97
colic	0.63	15.40	16.18	20.00
credit-a	0.52	120.01	15.60	47.33
credit-g	0.68	443.78	20.95	61.15
diabetes	0.53	44.58	13.35	32.56
glass	0.51	16.91	11.40	14.35
heart-statlog	0.46	60.58	13.74	14.29
hepatitis	0.53	14.12	11.27	5.67
hypothyroid	1.25	61.49	48.24	152.60
ionosphere	0.99	36.91	16.09	49.77
iris	0.44	0.52	7.76	1.69
labor	0.48	2.06	9.12	6.10
letter	0.82	NaN	291.99	229.86
lymph	0.19	27.39	14.81	5.22
segment	0.62	172.69	44.28	40.58
sonar	0.67	36.73	32.41	34.96
spambase	3.47	475.76	160.90	228.13
vehicle	0.26	25.52	24.38	57.17
vowel	0.14	22.22	16.22	7.07
mean	0.704	86.603	38.299	53.128
sd	0.664	131.364	65.041	65.847
median	0.530	36.730	16.135	33.320

Table 6: Number of wins for individual evaluation criteria across the 22 evaluation datasets: *acc* denotes accuracy, *#rules* average number of rules, *length* denotes average antecedent length, *time* average build time, *size* is computed as *rules* $\times$ *length*

	acc	#rules	length	time	size
base	5	5	14	22	14
heuristic	7	2	0	0	0
sa	3	12	6	0	6
saopt	7	3	2	0	2

We plan to address some of the limitations noted above in a larger follow-up study. In future work, it would also be interesting to adapt the proposed rule tuning heuristics to the recent generation of association rule classification algorithms. Unlike CBA, which uses a computationally lightweight approach to selecting rules for the final classifier, these algorithms typically subject the input rule set to much more sophisticated selection process, involving optimization techniques such as Markov Chain Monte Carlo (in SBRL), submodular optimization (in IDS) or simulated annealing (in BRS).

This adaptation may require experimentation with other metaparameter optimization algorithms, such as sequential model based optimization approaches (SMBO) [19] or other types of nature-inspired algorithms, e.g. F-race (Irace) [20], which were experimentally showed to outperform SMBO on tasks with mixed types of parameters [21].

6 Conclusions

In this paper, we have shown how thresholds used in rule generation can be tuned in both unsupervised and supervised way to improve results of association rule classification algorithms in terms of predictive performance and size of the resulting model. Our results showed, somewhat surprisingly, that the default thresholds recommended for the CBA algorithm (1% minimum support and 50% minimum confidence thresholds) provide on many datasets results highly competitive to the best configuration found with any of the proposed tuning algorithms. Despite this, using these defaults cannot be unanimously recommended as the default settings works well on some datasets, but has abysmal results on others. The proposed unsupervised heuristic tuning algorithm provides best predictive accuracy and relatively stable results. The supervised approach based on simulated annealing has promising results in terms of generating compact models.

Possible applications include not only general classification problems, but particularly the use of associative classification for anomaly detection, where the results are known to be very sensitive to the choice of the support threshold [22].

The implementation of the proposed algorithms is made available in R packages rCBA and arc, which are available in the CRAN repository.

Acknowledgments

The authors would like to thank the three anonymous reviewers for insightful comments that helped to improve the final version of the paper. This research was supported by Faculty of Informatics, Czech Technical University in Prague and by the Faculty of Informatics and Statistics, University of Economics, Prague by institutional support for research and grant IGA 12/2019.

References

[1] Lakkaraju, H.; Bach, S. H.; Leskovec, J.: Interpretable Decision Sets: A Joint Framework for Description and Prediction. In *Proceedings of KDD '16*, New York, NY, USA: ACM, 2016, ISBN 978-1-4503-4232-2, s. 1675–1684.

[2] Yang, H.; Rudin, C.; Seltzer, M.: Scalable Bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, JMLR. org, 2017, s. 3921–3930.

[3] Wang, T.; Rudin, C.; Doshi-Velez, F.; aj.: A bayesian framework for learning rule sets for interpretable classification. *The Journal of Machine Learning Research*, ročník 18, č. 1, 2017: s. 2357–2393.

[4] Liu, B.; Hsu, W.; Ma, Y.: Integrating classification and association rule mining. In *Proceedings of KDD'98*, 1998, s. 80–86.

[5] Alcala-Fdez, J.; Alcala, R.; Herrera, F.: A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy Systems*, ročník 19, č. 5, 2011: s. 857–872.

[6] Yin, X.; Han, J.: CPAR: Classification based on Predictive Association Rules. In *Proceedings of the SIAM International Conference on Data Mining*, San Franciso: SIAM Press, 2003, s. 369–376.

[7] Li, W.; Han, J.; Pei, J.: CMAR: Accurate and Efficient Classification Based on Multiple Class-Association Rules. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, ICDM '01, Washington, DC, USA: IEEE, 2001, ISBN 0-7695-1119-8, s. 369–376.

[8] Vanhoof, K.; Depaire, B.: Structure of association rule classifiers: a review. In *2010 International Conference on Intelligent Systems and Knowledge Engineering (ISKE)*, November 2010, s. 9–12.

[9] Agrawal, R.; Imielinski, T.; Swami, A. N.: Mining Association Rules between Sets of Items in Large Databases. In *SIGMOD*, 1993, s. 207–216.

[10] Han, J.; Pei, J.; Yin, Y.; aj.: Mining Frequent Patterns Without Candidate Generation: A Frequent-Pattern Tree Approach. *Data Mining and Knowledge Discovery*, ročník 8, č. 1, Leden 2004: s. 53–87, ISSN 1384-5810.

[11] Coenen, F.; Leng, P.; Zhang, L.: Threshold tuning for improved classification association rule mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2005, s. 216–225.

[12] Černý, V.: Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, ročník 45, č. 1, 1984: s. 41–51, ISSN 1573-2878.

[13] Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P.: Optimization by Simulated Annealing. *Science*, ročník 220, č. 4598, 1983: s. 671–680, ISSN 00368075, doi:10.1126/science.220.4598.671.

[14] Johnson, D. S.; Aragon, C. R.; McGeoch, L. A.; aj.: Optimization by Simulated Annealing: An Experimental Evaluation. Part I, Graph Partitioning. *Oper. Res.*, ročník 37, č. 6, Říjen 1989: s. 865–892, ISSN 0030-364X, doi:10.1287/opre.37.6.865.

[15] Dua, D.; Graff, C.: UCI Machine Learning Repository. 2017. Available on: `http://archive.ics.uci.edu/ml`

[16] Hühn, J.; Hüllermeier, E.: FURIA: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, ročník 19, č. 3, 2009: s. 293–319.

[17] Quinlan, J. R.: Improved use of continuous attributes in C4. 5. *Journal of artificial intelligence research*, ročník 4, 1996: s. 77–90.

[18] Fayyad, U. M.; Irani, K. B.: Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. In *13th International Joint Conference on Uncertainly in Artificial Intelligence (IJCAI93)*, 1993, s. 1022–1029.

[19] Bergstra, J. S.; Bardenet, R.; Bengio, Y.; aj.: Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, 2011, s. 2546–2554.

[20] Birattari, M.; Yuan, Z.; Balaprakash, P.; aj.: F-Race and iterated F-Race: An overview. In *Experimental methods for the analysis of optimization algorithms*, Springer, 2010, s. 311–336.

[21] Mantovani, R. G.; Horváth, T.; Cerri, R.; aj.: An empiri-

cal study on hyperparameter tuning of decision trees. *arXiv preprint arXiv:1812.02207*, 2018.

[22] Brauckhoff, D.; Dimitropoulos, X.; Wagner, A.; aj.: Anomaly extraction in backbone networks using association rules. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, ACM, 2009, s. 28–34.