Published in final edited form as:

Proc Congr Evol Comput. 2013 ; : 3236–3243. doi:10.1109/CEC.2013.6557966.

Integrating Clonal Selection and Deterministic Sampling for Efficient Associative Classification

Samir A. Mohamed Elsayed,

Computer Science Department, University of Connecticut, Storrs, CT 06269, Helwan University, Cairo, Egypt Samir@engineer.uconn.edu

Sanguthevar Rajasekaran, and

Computer Science Department, University of Connecticut, Storrs, CT 06269
Raj@engr.uconn.edu

Reda A. Ammar

Computer Science Department, University of Connecticut, Storrs, CT 06269
Reda@engr.uconn.edu

Abstract

Traditional Associative Classification (AC) algorithms typically search for all possible association rules to find a representative subset of those rules. Since the search space of such rules may grow exponentially as the support threshold decreases, the rules discovery process can be computationally expensive. One effective way to tackle this problem is to directly find a set of high-stakes association rules that potentially builds a highly accurate classifier. This paper introduces AC-CS, an AC algorithm that integrates the clonal selection of the immune system along with deterministic data sampling. Upon picking a representative sample of the original data, it proceeds in an evolutionary fashion to populate only rules that are likely to yield good classification accuracy. Empirical results on several real datasets show that the approach generates dramatically less rules than traditional AC algorithms. In addition, the proposed approach is significantly more efficient than traditional AC algorithms while achieving a competitive accuracy.

I. Introduction

The Natural Immune System (NIS) is a distributed, multi-layered, adaptive, dynamic, and life-long learning system. The Artificial Immune System (AIS) is a computational system inspired by the principles and processes of the NIS. The field of AIS has obtained some degree of success as a branch of computational intelligence since its emergence in the 1990s. There have been several successful applications of AIS in computer security, optimization, anomaly detection, and data mining.

Data mining is the process of discovering patterns from large data sets. One of the branches of data mining is Associative Classification (AC). AC algorithms integrate association rules discovery and classification to build a classifier from a training data for predicting the class of unforeseen test data. AC algorithms typically build a classifier by discovering the full set of Class Association Rules (CARs) from the training dataset and then select a subset to form a classifier. CARs are association rules of the form $iset \rightarrow c$, where $iset$ is an item-set and c is a class. Despite achieving high accuracy compared to other classification approaches such as C4.5 [1], the approach suffers from the overhead of exhaustive search through a large

pool of candidate rules. Moreover, the rule discovery process in traditional AC algorithms is not well integrated with the classification process.

Meanwhile, mining algorithms often require multiple passes over the full dataset which can be a performance challenge due to the ever-increasing volume of data. One way to tackle this scalability issue is to use only a sample of the data. However, sampling produces only approximate results. There is usually a trade-off between sampling rate and the desired accuracy. Naturally, the larger the sampling rate, the higher will be the accuracy.

Sampling can potentially make mining huge datasets feasible. Moreover, sampling algorithms may facilitate interactive mining [2]. When the goal is to obtain one or more interesting association rules as quickly as possible, a user might first mine a very small sample. If the results are unsatisfactory, the sample size can be iteratively increased until interesting rules are found. However, sampling may suffer from a couple of shortcomings. First is the presence of *missed item-sets* which are frequent in the entire dataset but infrequent in the sample. Second, there could be *false item-sets* which are infrequent in the entire dataset but frequent in the sample. These two issues could affect the accuracy of the sampling process.

The main contribution of this paper is AC-CS, an AC algorithm inspired by the Clonal Selection (CS) mechanism of the NIS. The algorithm employs deterministic data sampling for efficiency as well. Unlike traditional AC algorithms, the algorithm doesn't search for all association rules and then select a subset of them. Rather, AC-CS directly populates high quality rules through an evolutionary process. This evolutionary process follows the clonal selection process of cloning, mutating, and pruning populations of rules based on both their support and confidence measures. The process is repeated until the generated rules cover all the training data.

The rest of the paper is organized as follows. Section II gives a background about the immune system processes and algorithms as well as the associative classification problem. Section III describes the proposed algorithm in detail. Experimental results are reported in Section IV. Section V concludes the paper with some possible future work. I wish you the best of success.

II. Background

In this section, a brief description of the immune system principles and processes that computer scientists draw inspiration from when designing AIS algorithms is presented. A brief summary of one of the major AIS algorithms, namely, the clonal selection algorithm, is also presented. In addition, the associative classification approach along with issues of traditional algorithms are discussed. Selected sampling algorithms are briefly discussed as well.

A. The Immune System

The Natural Immune System (NIS) is a complex network of tissues, organs, and chemicals. Its main function is to defend the body against foreign *pathogens* such as bacteria or viruses. NIS recognizes pathogens via a smaller portion on top of them called *antigens*. The main organs of the NIS where the immune cells develop, include the lymph system, the thymus, and the bone marrow. The immune system protects the body from infection with layered defenses. Physical barriers (e.g., skin) prevent pathogens from entering the body. If a pathogen breaches these barriers, the innate immune system provides an immediate but non-specific response. In many cases, it is able to successfully defend the body. If the innate immune system fails to handle the attack, then the adaptive immune system takes over. The

adaptive response is more specific yet slower, resulting in a more effective response. The adaptive immune system can remember past encounters with antigens such that the next time the antigen appears, a more specific and effective response is deployed (the main idea of vaccination).

The main component of the adaptive immune system is lymphocytes which are blood white cells. Lymphocytes consist of two cell types, namely: *B-cells* and *T-cells*. B-cells get produced and develop in the bone marrow. While the T-cells get produced in the bone marrow and then migrate to the thymus to develop.

There are two main types of B-cells, namely: plasma B-cells and memory B-cells. Plasma B-cells are cells that have been exposed to antigens and produce large amounts of *antibodies*. Antibodies are Y-shaped receptor molecules bound on the surface of a B-cell with the primary role of recognizing and binding, through a complementary match, with an antigen [3]. *Memory* B-cells, created in response to high affinity match with antigens, live for a very long time and provide faster future response. B-cells require co-stimulation from other immune cells (e.g., helper T-cells) in order to activate. B-cells undergo a process called *clonal selection*.

AIS is a relatively young field. The first paper by Farmer et al. [4] in the field was published in 1986. However, it was only in the mid-90's that AIS has become a subject area in its own right. There are several works that form the foundation of AIS. Examples of such works include the work by Forrest et al. [5] in 1994 as well as the work by Bersini [6]. There are five major AIS algorithms that have been popular in the literature, namely: negative selection [5], [7], clonal selection [8], [9], artificial immune networks [10], [11], dendritic cells [12], and danger theory algorithms [13].

B. The Clonal Selection (CS)

According to Burnet's 1959 clonal selection theory, activation of lymphocytes occurs on binding with a matching antigen [14]. Once activated, *clones* of the lymphocyte are produced in large numbers with identical receptors of the original lymphocyte that encountered the antigen. Any lymphocytes that bind to self's cells are set to die. During the clonal selection of B-cells, the average antibody affinity increases for the antigen that triggered the process. This phenomenon is called *affinity maturation* and is responsible for the fact that upon a subsequent exposure to the antigen, the immune response is more effective due to the antibodies having a higher affinity for the antigen. Affinity maturation is facilitated by a *somatic hyper mutation* and selection mechanisms that occur during the clonal expansion of B-cells. Somatic hyper mutation alters the specificity of antibodies by introducing random changes to them.

Researchers have tried to draw some inspiration from the clonal selection process, in particular, the antigen driven affinity maturation process of B-cells along with the hyper mutation mechanism. In [15], the authors highlight two features of affinity maturation in B-cells that can be exploited from a computational point. First, the proliferation of B-cells is proportional to the affinity of the antigen that binds it, the higher the affinity, the more clones are produced. Second, the mutations suffered by B-cells are inversely proportional to the affinity of the antigen it binds. Utilizing these two features, De Castro and Von Zuben [8] developed one of the most popular clonal selection inspired algorithms, CLONALG. There are different variations of clonal selection algorithms in the literature. [9] presents a comparative case study of a handful of such algorithms.

C. Sampling

One of the oldest sampling algorithms for rules mining is by Toivnen [16]. The algorithm selects *randomly* a fixed size sample from the dataset. It finds all *maximal* and *negative border* item-sets in the sample. Note that an item-set is maximal if it is frequent but none of its supersets is frequent. An item-set is in the negative border if it is not deemed frequent in the sample but all of its immediate subsets are. Upon checking with the remainder of the dataset (i.e., the dataset minus the sample), if no negative border item-set turns out to be frequent, then the algorithm has succeeded and ends. Otherwise, another pass of the algorithm to eliminate false item-sets is required which can be quite expensive.

To tackle scalability issues of mining entire datasets, Zaki, et al. [17] argue that simple random sampling can reduce the I/O cost and computation time for association rule mining. The survey in [18] gives an overview of random sampling algorithms in databases. In [19], the authors propose two approaches to sample selection, namely, static sampling and dynamic sampling. With static sampling, a random sample is drawn from the large dataset, and hypothesis testing is used to establish whether it is sufficiently similar to the parent dataset. In dynamic sampling, a decision is made after each sampled element whether to continue sampling or not. If the current sample is considered sufficient for the required level of accuracy, the sampling stops. Similarly, the authors of [20] propose using progressively larger samples as long as the model accuracy improves.

FAST [2] is another sampling algorithm. It proceeds in two phases. In the first phase, a large initial sample of transactions is selected *randomly* to estimate the support of each item in the dataset. In the second phase, these estimated supports are used to trim outlier transactions or select representative transactions from the initial sample, thereby forming a smaller final sample that potentially more accurately approximates item-set supports. EASE [21] is similar to FAST in that the final sample is obtained by deterministic methods from a larger random sample. EASE, however, uses an ϵ -approximation approach to obtain the final sample by a process of repeated halving. The sampling method used in this paper shares some ideas with these algorithms.

D. Associative Classification (AC)

Agrawal, et al. [22] proposed association rules mining for market basket data patterns. Association rules identify correlations among a set of items found in transactions. The input to the association rules mining problem is a set of transactions where each transaction is a set of items. A set of items is also referred to as an *item-set*. A k -item-set is an item-set of size k . There are two measures proposed in [22] that quantify the significance of an association rule, namely, support and confidence ratio. Formally, an association rule is an implication $x \rightarrow y$ where x and y are item-sets in a dataset. The *support* of the rule is the ratio of the number of transactions containing both x and y to the total number of transactions. The *confidence* of the rule is the ratio of the number of transactions that contain y to the number of transactions containing x . The mining of association rules from a set of transactions is the process of identifying all rules having a pre-specified minimum support and confidence. This involves several phases in processing transactions. A k -item-set is said to be *frequent* if the ratio of the number of transactions containing all the items of the k -item-set to the total number of transactions is greater than or equal to the user specified minimum support. Algorithms for finding frequent item-sets typically require multiple passes over the entire dataset.

There are many algorithms for mining association rules in the literature. Apriori [22] is possibly the well-known and most cited algorithm. It uses a breadth-first search strategy and generates candidate item-sets and tests if they are frequent. The key to its success over older

algorithms such as AIS [23] and STEM [24] is the fact that it exploits an important property (commonly referred to as Apriori property or downward closure property). This property is the observation that no superset of an infrequent item-set can be frequent. However, generation of candidate item-sets can be expensive both in space and time. In addition, support counting involves multiple dataset scans which heavily impact performance. Apriori as well as Apriori-inspired algorithms (e.g., [25]) typically perform well on sparse (i.e., short frequent item-sets) such as the market basket data. However, they perform poorly on dense (i.e., long frequent item-sets) datasets such as census data. This degradation is due to the fact that these algorithms perform as many passes over the database (i.e., high I/O overhead) as the length of the longest frequent pattern [26].

While many algorithms including Apriori use the traditional horizontal data layout, Eclat [27] is probably the first algorithm to introduce a vertical data layout. Eclat is more efficient for long item sets than for short ones. In this algorithm, data is represented as lists of transaction identifiers (one per item). Support counting is performed by simply intersecting these lists. Compared to Apriori and other algorithms, Eclat often performs better on dense rather than sparse datasets. A variation of the algorithm depicted dEclat can be found in [26].

FP-Growth [28] allows frequent item-sets discovery without candidate item-set generation and adopts a divide and conquer strategy. It builds a compact data structure (i.e., FP-Tree) using only two passes over the dataset. The algorithm is more efficient as long as the full tree can fit in memory. However, the tree may need substantial memory in some cases. There are some variations of the algorithm such as H-mine [29].

Liu et al. [30] introduced the Classification Based on Associations (CBAs) algorithm which laid the foundation of the AC approach and other algorithms as well (e.g., CMAR [31],[32]). The basic idea is to mine the full set of association rules from a training dataset into Class Association Rules (CARs) in the form of $iset \rightarrow c$ where $iset$ is an item-set (i.e, body of the rule) and c is a class (i.e, head of the rule). Rules with high confidence are then selected to form the classifier. During the classification process, all test data that match the rule with the highest confidence are assigned the class of that rule. CMAR generates and evaluates rules in a similar fashion but rather uses a more efficient FPtree structure [31]. One of the key differences is the use of multiple rules in prediction. Empirical results show that CMAR is more accurate than CBA.

AIS data mining applications include classification [33], [34], [35], clustering [3], and rule induction [36]. Most AISs for classification use an instance-based representation. This includes AIRS [37], [38], [39] and CLONALG [8], [40]. In this representation, the candidate solutions considered by the classification algorithm take the form of a subset of the original data instances, each of them with all of their attribute values [41]. Another type of representation is the rule-based one. This representation is used, for instance, in IFRAIS [36]: an AIS for discovering fuzzy classification rules. In this representation, the candidate solutions considered by the algorithm take the form of IF-THEN classification rules, where each rule typically contains a conjunction of a few attribute values [41]. Both representations have pros and cons. In [41], the authors argue that it all depends on the nature of the data being mined, the requirements of the application domain, and how important knowledge comprehensibility is to the user.

In [35] the authors propose an algorithm that employs the clonal selection of the immune system to directly find a subset of rules to form the classifier. Empirical results show that the algorithm is particularly efficient when the support threshold is low. In addition, reported

accuracy appears to be competitive with a traditional AC algorithm. The approach in this paper is similar to this work.

III. The AC-CS Algorithm

The AC-CS algorithm is inspired by the clonal selection process, in which rules are populated as if they are B-cells that can be cloned, mutated, and pruned as well. Affinity measurements are determined by the confidence of those rules. The objective is to avoid an exhaustive search for all possible rules in an explosive search space as it is the case in traditional AC algorithms.

Prior to describing the algorithm in detail, necessary notations and concepts are defined as follows:

- A class rule r can be defined as $iset \rightarrow c$ where $iset$ is an item-set (i.e, body of the rule) and c is a class (i.e, head of the rule).
- The support of r , $support(r)$, is the ratio of the number of transactions that contain all items in both the head and body of the rule, to the total number of transactions of the data set. This measure helps remove low-stakes rules.
- The confidence of r , $confidence(r)$, is the ratio of the number of transactions that contain the body item-set to the number of transactions that contain both the body and head item-sets of the rule.
- The coverage of r , $coverage(r)$, is the ratio of the number of transactions that are covered by the generated rules to the total number of transactions. The coverage measure has been previously used in [30] to filter the discovered rules. We use the coverage measure as a termination check to the algorithm. Indeed, if the generated rules managed to cover the whole training data set, then there would be no need to proceed further.
- The clonal rate of r is dictated by the following equation, borrowed from [35]:

$$cloneRate(r) = \frac{n \cdot clonalRate}{\sum_{i=1}^n confidence(r_i)} \cdot confidence(r)$$

where n denotes the number of rules at the current generation, and $clonalRate$ is a user defined parameter. The clone rate of the rule is proportional to its confidence (i.e., the higher the confidence, the higher will be the clone rate).

In [42], we introduced ML-DS (Multi-Level Deterministic Sampling), a deterministic sampling algorithm that attempts to improve accuracy without sacrificing the running time. ML-DS begins with a large sample deterministically selected from the dataset and then proceeds in levels. First, it divides the remaining data into disjoint groups of equal size. Each group in turn is recursively divided into smaller disjoint subgroups of equal size. A distance measure is then determined for each subgroup against the original group. Subgroups with minimum distance are retained while others are discarded. The process is repeated until the size of the remaining transactions is equal to a desired sampling threshold. We employ this sampling strategy to pick a good training sample to begin with.

The proposed algorithm AC-CS, depicted in figure 1, accepts a training sample of transactions and outputs a set of high quality rules that potentially yield a good accuracy classifier. The algorithm starts off by initializing a few data structures namely: the rule populations P , the frequent singleton items I , and finally the memory pool M that contains

the high quality rules found throughout the algorithm (line 1). User-defined parameters are initialized in line 2. In lines (3-11), rules with only a single item in the head and which have a higher support than the given *minSupport* are added to *P*. In addition, only items that turn out to be frequent are added to the new set of items *I*. From now on, AC-CS doesn't use the full set of singleton items *I*. Rather, it uses the refined set *I*. Note that *I* is a subset of *I*.

The algorithm proceeds in generations. In each generation, remaining rules from previous generations are cloned proportional to their confidence (i.e., affinity) which is calculated using the *cloneRate* function above (line 16). For each new cloned rule, the items in the body of the rule are extended (i.e., mutated) by adding one more *new* item (line 19). Only rules with a high support are retained (lines 20-21). Next, the remaining rules go through a pruning phase in which redundant rules are removed. For instance, given two rules $r_1 : iset_1$ and $r_2 : iset_2 \quad c$:

$$iset_2 \subseteq iset_1 \wedge confidence(r_1) > confidence(r_2) \Rightarrow prune(r_2)$$

Upon pruning all the redundant rules, only remaining rules with a high confidence are added to the memory pool *M* (lines 26-30). The process is repeated until either the rules in *M* cover the entire training dataset or the number of generations exceeds the maximum allowed (line 32).

Note that the whole process is repeated for each class in the dataset. Upon completion, all the rules in the memory pool *M* are sorted in decreasing order of confidence. Sorted rules are then applied one by one to a test data and the accuracy is reported accordingly.

IV. Experimental Results

This section demonstrates the potential of the proposed approach using several benchmarks. The experimental setup is first described followed by a discussion of both the capabilities and limitations of the approach.

A. Setup

Experiments are performed on real datasets of different sizes obtained from the UCI Machine Learning Repository [43]. These datasets are shown in table I along with their parameters. A few pre-processing steps were performed on the datasets as follows:

- Continuous data were discretized into intervals using the entropy methods among others with the MLC++¹ and the LUCS-KDD² software tools.
- Any transactions with missing fields were removed. Number of actual instances reflect this fact in table I.
- Data were converted into a vertical layout for efficient support counting.
- A hash table for item transactions was implemented for easy and efficient access.
- Items in the body of rules were implemented as sets.

The proposed algorithm AC-CS (with and without sampling) is compared against AC-Apriori, an Apriori based AC algorithm that generates the full set of possible association rules. We use a publicly available implementation of Apriori by Borgelt³. This

¹<http://www.sgi.com/tech/mlc/>

²<http://cgi.csc.liv.ac.uk/frans/KDD/Software/>

³<http://www.borgelt.net/apriori.html>

implementation is widely used for academic and commercial purposes. Additionally, the AC-CS is evaluated against the CMAR algorithm as well.

Execution times are reported in milliseconds, seconds, and minutes as appropriate. They include both the time to extract and apply the generated rules. The minimum support ratios used are 0.5%, 0.75%, 1%, 2%, 5%, 10%. A time out of 60 minutes is employed. All experiments were performed on a Windows 7 machine with a processor speed of 2.80GHz and a memory of 3.0GB.

B. Discussion

In this section, we discuss the performance of the proposed algorithm in terms of accuracy, execution time, and the discovered rules size. We also show how the algorithm compares against AC-Apriori as well as CMAR.

The bar charts in figure 2 illustrate the accuracy of both the AC-Apriori and AC-CS algorithms on several datasets with different support thresholds. Clearly, on both the Iris and the Nursery datasets, AC-CS has a higher accuracy than that of AC-Apriori in every single case. Meanwhile, for the larger dataset Adult, AC-CS's accuracy is approaching and sometimes rivaling that of AC-Apriori. Similarly, the line charts in the same figure show the execution times of the two algorithms. For the Adult and Iris datasets, AC-CS outperforms AC-Apriori in all cases. This is probably due to the significant reduction of rules required as a result of not discovering all possible rules as it is the case with AC-Apriori.

The bar charts in figure 3 illustrate the accuracy of both CMAR and AC-CS algorithms on the same datasets with the same support thresholds. Clearly, on Adult, Iris, as well as Nursery datasets, AC-CS has achieved a competitive accuracy compared to that of CMAR. On the letter dataset, the results are particularly interesting. For high support threshold, CMAR outperforms AC-CS in accuracy. However, for smaller support threshold, the accuracy gets more competitive and actually outperforms CMAR at low support thresholds. The line charts in the same figure show that AC-CS outperforms CMAR on all the datasets in running time. In particular, the CMAR algorithm times out on the letter dataset for smaller than 1% support threshold while AC-CS achieved a high accuracy efficiently.

In general, the accuracy of AC-CS with sampling is very close to that without sampling. However, there is a clear reduction in the running time on all datasets. This indicates that the sampling approach is very effective in producing a good representative sample of the original full dataset. In terms of the number of rules generated, there is a significant reduction in size for the case of AC-CS. The reduction is particularly apparent in the case of the Adult dataset. While the rules used by AC-CS grow slowly with lower minSupport, the rules generated by AC-Apriori and CMAR grow exponentially in some cases. However, when the rules don't differ much like in the case of the dataset Letter, AC-CS loses its efficiency advantage. It is worth mentioning that the AC-CS algorithm appears to be more efficient than the algorithm proposed in [35] as well.

AC-CS algorithm is able to integrate the rule discovery along with the rule selection processes, a clear advantage that traditional AC algorithms lack. Moreover, unlike Genetic Algorithms (GA), the employed CS algorithm performs proportionate selection as well as affinity inversely proportional hyper-mutation with no crossover operation [44]. The authors in [8] show that unlike GA, CS algorithm can reach a diverse number of local optimal solutions. The algorithm has two influential parameters, namely, *clonalRate* and *maxGens*. The former dictates the rate at which items in rules at a given generation are extended. The later dictates how many generations the evolutionary process may go through. The algorithm ends if the number of generations exceeds this limit or the whole training set has been

already covered by the current rules, whichever occurs sooner. We chose a value of 10 for both the *clonalRate*, and the *maxGens* since this was shown to be a reasonable choice in terms of both time and accuracy.

V. Conclusions and Future Work

In this paper, we have introduced AC-CS, an AC algorithm inspired by the clonal selection algorithm. The algorithm begins by picking a representative sample from the full dataset. It then begins with a small population of frequent single item rules. These rules then go through a process of cloning, mutating, and pruning for several generations. Only high quality rules are added to the memory pool. These rules are applied in turn to classify a testing dataset. Empirical results show that the approach outperforms traditional AC algorithms in running time and quite competitive in accuracy as well. In addition, our sampling approach significantly improve running time while maintaining high accuracy. Possible future research directions for this work include: drawing more inspiration from the NIS processes; testing different clonal rates of rules; and improving the accuracy while keeping the run time minimal.

Acknowledgments

The first author is grateful to the Computer Science Department, Helwan University, Cairo, Egypt for supporting his Ph.D. program at the University of Connecticut.

This work has been partially supported by the following grants: NSF0829916 and NIH-R01-LM010101. The first author is partially supported by Helwan University, Cairo, Egypt.

References

1. Quinlan, J. C4 5: programs for machine learning. Morgan kaufmann; 1993.
2. Chen, B.; Haas, P.; Scheuermann, P. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, ser KDD '02. New York, NY, USA: ACM; 2002. A new two-phase sampling based algorithm for discovering association rules; p. 462-468.
3. Castro LND, Zuben FJV. An evolutionary immune network for data clustering. Brazilian Symposium on Neural Networks. 2000:84–89.
4. Farmer J, Packard N, Perelson A. The immune system, adaptation, and machine learning. Physica D: Nonlinear Phenomena. 1986; 22(no. 1-3):187–204.
5. Forrest, S.; Perelson, A.; Allen, L.; Cherukuri, R. Research in Security and Privacy, 1994 Proceedings 1994 IEEE Computer Society Symposium on. IEEE; 1994. Self-nonself discrimination in a computer; p. 202-212.
6. Bersini H, Varela F. Hints for adaptive problem solving gleaned from immune networks. Parallel problem solving from nature. 1991:343–354.
7. Ji Z, Dasgupta D. Revisiting negative selection algorithms. Evolutionary Computation. 2007; 15(no. 2):223–251. [PubMed: 17535140]
8. De Castro L, Von Zuben F. The clonal selection algorithm with engineering applications. Proceedings of GECCO'00, workshop on artificial immune systems and their applications. 2000; 3637
9. Cutello, V.; Narzisi, G.; Nicosia, G.; Pavone, M. Clonal selection algorithms: A comparative case study using effective mutation potentials. International Conference on Artificial Immune Systems; 2005. p. 13-28.
10. Ishida, Y. Fully distributed diagnosis by pdp learning algorithm: towards immune network pdp model. Neural Networks, 1990., 1990 IJCNN International Joint Conference on; IEEE; 1990. p. 777-782.
11. Timmis J, Neal M, Hunt J. An artificial immune system for data analysis. Biosystems. 2000; 55(no. 1-3):143–150. [PubMed: 10745118]

12. Greensmith, J.; Aickelin, U.; Cayzer, S. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. *International Conference on Artificial Immune Systems*; 2005. p. 153-167.
13. Matzinger P. The danger model: a renewed sense of self. *Science*. 2002; 296(no. 5566):301–305. [PubMed: 11951032]
14. <http://www.artificial-immune-systems.org>
15. De Castro, L.; Timmis, J. *Artificial immune systems: a new computational intelligence approach*. Springer Verlag; 2002.
16. Toivonen, H. Sampling large databases for association rules. *Proceedings of the 22th International Conference on Very Large Data Bases, ser VLDB '96*; San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.; 1996. p. 134-145.
17. Zaki, MJ.; Parthasarathy, S.; Li, W.; Ogihara, M. *Proceedings of the 7th International Workshop on Research Issues in Data Engineering, ser RIDE '97*. Washington, DC, USA: IEEE Computer Society; 1997. Evaluation of sampling for data mining of association rules; p. 42
18. Olken F, Rotem D. Random sampling from databases: a survey. *Statistics and Computing*. 1995; 5(no. 1):25–42.
19. John, G.; Langley, P. Static versus dynamic sampling for data mining. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*; 1996. p. 367-370.
20. Provost, F.; Jensen, D.; Oates, T. Efficient progressive sampling. *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, ser KDD '99*; New York, NY, USA: ACM; 1999. p. 23-32.
21. Brönnimann, H.; Chen, B.; Dash, M.; Haas, P.; Scheuermann, P. Efficient data reduction with ease. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ser KDD '03*; New York, NY, USA: ACM; 2003. p. 59-68.
22. Agrawal, R.; Srikant, R., et al. Fast algorithms for mining association rules. *Proc 20th Int Conf Very Large Data Bases, VLDB*; 1994. p. 487-499.
23. Agrawal, R.; Imielski, T.; Swami, A. Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD international conference on Management of data, ser SIGMOD '93*; New York, NY, USA: ACM; 1993. p. 207-216.
24. Houtsma, M.; Swami, A. Set-oriented mining of association rules. *International Conference on Data Engineering*; 1993.
25. Bayardo, RJ, Jr. Efficiently mining long patterns from databases. *Proceedings of the 1998 ACM SIGMOD international conference on Management of data, ser. SIGMOD '98*; New York, NY, USA: ACM; 1998. p. 85-93.
26. Zaki, MJ.; Gouda, K. Fast vertical mining using diffsets. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ser KDD '03*; New York, NY, USA: ACM; 2003. p. 326-335.
27. Zaki MJ, Parthasarathy S, Ogihara M, Li W. New algorithms for fast discovery of association rules. *Knowledge Discovery and Data Mining*. 1997:283–286.
28. Han, J.; Pei, J.; Yin, Y. Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data, ser SIGMOD '00*; New York, NY, USA: ACM; 2000. p. 1-12.
29. Pei, J.; Han, J.; Lu, H.; Nishio, S.; Tang, S.; Yang, D. H-mine: hyperstructure mining of frequent patterns in large databases. *Proceedings IEEE International Conference on Data Mining, ICDM'2001*; 2001. p. 441-448.
30. Liu B, Hsu W, Ma Y. Integrating classification and association rule mining. *Knowledge Discovery and Data Mining*. 1998:80–86.
31. Li W, Han J, Pei J. Cmar: Accurate and efficient classification based on multiple class-association rules. *IEEE International Conference on Data Mining*. 2001:369–376.
32. Liu B, Ma Y, Wong C. Classification using association rules: weaknesses and enhancements. *Data mining for scientific applications*. 2001:1–11.
33. Zheng, H.; Du Jiaying, Z.; Wang, Y. Research on vehicle image classifier based on concentration regulating of immune clonal selection. *Fourth International Conference on Natural Computation*; IEEE; 2008. p. 671-675.

34. Gu, F.; Feyereisl, J.; Oates, R.; Reys, J.; Greensmith, J.; Aickelin, U. *Artificial Immune Systems*. Springer; 2011. Quiet in class: Classification, noise and the dendritic cell algorithm.
35. Do TD, Hui SC, Fong ACM, Fong B. Associative classification with artificial immune system. *IEEE Transactions on Evolutionary Computation*. 2009; 13:217–228.
36. Alves, R.; Delgado, M.; Lopes, H.; Freitas, A. *Parallel Problem Solving from Nature-PPSN VIII*. Springer; 2004. An artificial immune system for fuzzy-rule induction in data mining; p. 1011-1020.
37. Watkins A, Timmis J, Boggess L. Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm. *Genetic Programming and Evolvable Machines*. 2004; 5(no. 3):291–317.
38. Watkins, A.; Timmis, J. *Artificial Immune Systems*. Springer; 2004. Exploiting parallelism inherent in airs, an artificial immune classifier; p. 427-438.
39. Watkins, A. Ph D dissertation. University of Kent, Computing Laboratory; 2005. Exploiting immunological metaphors in the development of serial, parallel and distributed learning algorithms.
40. De Castro L, Von Zuben F. Learning and optimization using the clonal selection principle. *Evolutionary Computation, IEEE Transactions on*. 2002; 6(no. 3):239–251.
41. Freitas, AA.; Timmis, J. Revisiting the foundations of artificial immune systems: a problem-oriented perspective. In: Hart, editor. *Artificial Immune Systems (Proc ICARIS-2003)*, LNCS 2787. Springer; 2003. p. 229-241.
42. Elsayed, SAM.; Rajasekaran, S.; Ammar, RA. MI-ds: A novel deterministic sampling algorithm for association rules mining. In: Perner, P., editor. *Advances in Data Mining Applications and Theoretical Aspects*, ser Lecture Notes in Computer Science. Vol. 7377. Springer; Berlin Heidelberg; 2012. p. 224-235.[Online]. Available: http://dx.doi.org/10.1007/978-3-642-31488-9_18
43. Blake, C.; Merz, C. *UCI Repository of machine learning databases*. University of California, Department of Information and Computer Science; Irvine, CA: 1998. <http://www.ics.uci.edu/mlearn/MLRepository.html>
44. Castro L, Timmis J. Artificial immune systems as a novel soft computing paradigm. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*. 2003; 7(no. 8):526–544.

```

1  $P = \{\}; I = \{\}; M = \{\};$ 
2 initialize clonalRate, currentGen, maxGens;
3 foreach item  $i \in \mathcal{I}$ 
4 {
5     set rule  $r = \{i \Rightarrow c\};$ 
6     if(support( $r$ )  $\geq$  minSupport)
7     {
8         set  $P = P \cup r;$ 
9         set  $I = I \cup i;$ 
10    }
11 }
12 do
13 {
14     foreach rule  $r \in P$ 
15     {
16         repeat cloneRate( $r$ ) times
17         {
18             set  $\bar{r} = r;$ 
19             set  $\bar{r}.items = r.items \cup i$  where  $i \in I \wedge i \notin r.items;$ 
20             if(support( $\bar{r}$ )  $\geq$  minSupport)
21                 set  $P = P \cup \bar{r};$ 
22         }
23     }
24     prune  $P$ 
25     foreach rule  $r \in P$ 
26     {
27         if(confidence( $r$ )  $\geq$  minConfidence)
28              $M = M \cup r;$ 
29     }
30     currentGen ++;
31 } while(coverage( $M$ )  $< 100\% \wedge$  currentGen  $< maxGens$ )
32 return  $M;$ 

```

Fig. 1.
The AC-CS Algorithm

Fig. 2. AC-CS vs. AC-Apriori: Accuracy and execution time results for datasets (from top left) Adult, Iris, Letter, and Nursery.

Fig. 3. AC-CS vs. CMAR: Accuracy and execution time results for datasets (from top left) Adult, Iris, Letter, and Nursery.

TABLE I

Datasets & Parameters

Dataset	Instances	Classes	Training	Testing
Adult	45,222	2	30,162	15,060
Iris	150	3	100	50
Letter	20,000	26	13,333	6,667
Nursery	12,960	5	8,640	4,320