A Two-Armed Bandit Collective for Examplar Based Mining of Frequent Itemsets with Applications to Intrusion Detection

Vegard Haugland, Marius Kjølleberg,
Svein-Erik Larsen, and Ole-Christoffer Granmo

University of Agder, Grimstad, Norway

Abstract. Over the last decades, frequent itemset mining has become a major area of research, with applications including indexing and similarity search, as well as mining of data streams, web, and software bugs. Although several efficient techniques for generating frequent itemsets with a minimum support (frequency) have been proposed, the number of itemsets produced is in many cases too large for effective usage in real-life applications. Indeed, the problem of deriving frequent itemsets that are both compact and of high quality, remains to a large degree open.

In this paper we address the above problem by posing frequent itemset mining as a collection of interrelated *two-armed bandit* problems. In brief, we seek to find itemsets that frequently appear as subsets in a stream of itemsets, with the frequency being constrained to support granularity requirements. Starting from a randomly or manually selected examplar itemset, a collective of Tsetlin automata based two-armed bandit players aims to learn which items should be included in the frequent itemset. A novel reinforcement scheme allows the bandit players to learn this in a decentralized and on-line manner by observing one itemset at a time. Since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present itemset and the corresponding decisions of the bandit players, the resulting memory footprint is minimal. Furthermore, computational complexity grows merely linearly with the cardinality of the examplar itemset.

The proposed scheme is extensively evaluated using both artificial data as well as data from a real-world network intrusion detection application. The results are conclusive, demonstrating an excellent ability to find frequent itemsets at various level of support. Furthermore, the sets of frequent itemsets produced for network instrusion detection are compact, yet accurately describe the different types of network traffic present.

1 Introduction

Over the last two decades, frequent itemset mining has become a major area of research, with applications including indexing and similarity search, as well as mining of data streams, web, and software bugs [1].

P. Jędrzejowicz et al. (Eds.): ICCCI 2011, Part I, LNCS 6922, pp. 72–81, 2011.

The problem of finding frequent itemsets can be formulated as follows. Consider a set I of n items, $I = \{i_1, i_2, \ldots, i_n\}$. A *transaction* T_i, $1 \leq i \leq m$, is defined as a subset of I, $T_i \subseteq I$, collectively referred to as a transaction set: $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$. When an arbitrary set X is a subset of a transaction T_i, $X \subseteq T_i$, one says that T_i supports X. The *support* of X is then simply the number of transactions T_i in $\mathcal{T}$ that supports X, $support(X) = |\{T_i \in \mathcal{T} | X \subseteq T_i\}|$, with $|\cdot|$ denoting set cardinality. The notion of interest in this paper – the *frequency* of an itemset – can then be defined as follows:

Definition 1 (Itemset Frequency). *The frequency of itemset X, $freq(X)$, is defined as the fraction of transactions T_i in $\mathcal{T}$ that supports X:*

$$freq(X) = \frac{|\{T_i \in \mathcal{T} | X \subseteq T_i\}|}{|\mathcal{T}|}.$$

Although several efficient techniques for generating frequent itemsets with a minimum frequency have been proposed [1], the number of itemsets produced is in many cases too large for effective usage in real-life applications. Indeed, the problem of deriving frequent itemsets that are both compact and of high quality, so that they are tailored to perform well in specific real-life applications, remains to a large degree open.

1.1 Our Approach

In this paper we address the above problem by posing frequent itemset mining as a collective intelligence problem, modelled as a collection of interrelated *two-armed bandit* problems. The two-armed bandit problem [5] is a classical optimization problem where a player sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting in a random reward. The reward distributions are unknown, and thus, one must balance between exploiting existing knowledge about the arms, and obtaining new information.

Our proposed scheme can be summarized as follows. Starting from a randomly or manually selected examplar transaction, a collective of so-called Tsetlin automata [7] based bandit players – one automaton for each item in the examplar – aims to learn which items should be included in the mined frequent itemset, and which items should be excluded. A novel reinforcement scheme allows the bandit players to learn this in a decentralized and on-line manner, by observing transactions one at a time, as they appear in the transaction stream. Since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present transaction and the corresponding decisions of the bandit players, the resulting memory footprint is minimal. Furthermore, computational complexity grows merely linearly with the cardinality of the examplar transaction.

The above Tsetlin automata based formulation of frequent itemset mining provides us with three distinct advantages:

1. Any desired target itemset frequency can be achieved without any more memory than what is required by the Tsetlin automata in the collective (one byte per automaton).

2. Itemsets are found by the means of on-line collective learning, supporting processing of on-line data streams, such as streams of network packets.
3. An examplar transaction is used to focus the search towards frequent itemsets that are both compact and of high quality, tailored to perform well in real-life applications.

1.2 Example Application — Network Anomaly Detection

Network intrusion detection has been a particularly promising application area for frequent itemset mining [8,9]. In so-called network anomaly detection, huge amounts of network packet data needs to be mined so that the patterns of normal traffic can be found, and so that anomalous traffic can be distilled as deviations from the identified patterns. Although not based on frequent itemset mining, the packet byte based anomaly detection approach of Mahoney [3] is particularly fascinating in this perspective because it achieves state-of-the-art anomaly detection performance simply by inspecting 48 bytes from the header of network packets.

In order to investigate to what degree the properties of our bandit problem based approach to frequent itemset mining can be taken advantage of in network anomaly detection, we will propose a packet byte based anomaly detection system, formulated as a frequent itemset problem. Informally stated, each network packet i is seen as a transaction T_i and each byte value from the network packet is seen as an item belonging to the transaction. Thus, in this application we are looking for frequent itemsets consisting of byte-value pairs, such as $\{dstaddr1 : 24, dstaddr2 : 34, tcpflag : 12\}$, which is an itemset that identifies network packets with destination $24.34.*.*$ and tcp-flag 12.

1.3 Paper Organization

The paper is organized as follows. First, in Sect. 2 we present our decentralized Tsetlin automata based solution to frequent itemset mining, as well as a novel reinforcement scheme that guides the collective of Tsetlin automata towards a given target itemset frequency. Then, in Sect. 3 we demonstrate the performance advantages of the introduced scheme, including its ability to robustly identify compact itemsets that are useful for summarizing both artificial as well as real-life data. Finally, in Sect. 4 we offer conclusions as well as pointers to further work.

2 A Collective of Two-Armed Bandit Players for Examplar Based Frequent Itemset Mining

We here target the problem of finding frequent itemsets with a given support by *on-line* processing of transactions, taking advantage of so-called transaction *examplars*. To achieve this, we design a collective of Learning Automata (LA) that builds upon the work of Tsetlin and the linear two-action automaton [4,7].

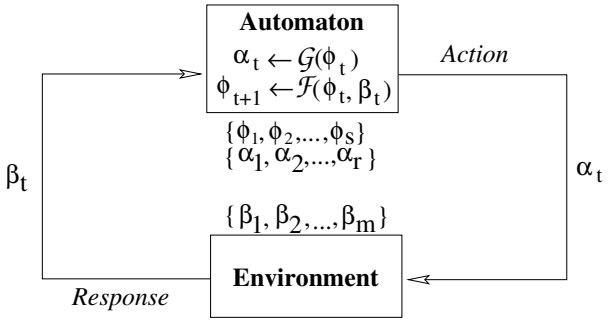

Fig. 1. A Learning Automaton interacting with an Environment

Generally stated, an LA performs a sequence of actions on an *Environment*. The Environment can be seen as a generic *unknown* medium that responds to each action with some sort of reward or penalty, generated *stochastically*. Based on the responses from the Environment, the aim of the LA is to find the action that minimizes the expected number of penalties received. Fig. 1 shows the interaction between a LA and the Environment.

As illustrated in the figure, an LA can be defined in terms of a quintuple [4]:

$$\{\underline{\Phi}, \underline{\alpha}, \underline{\beta}, \mathcal{F}(\cdot,\cdot), \mathcal{G}(\cdot,\cdot)\}.$$

$\underline{\Phi} = \{\phi_1, \phi_2, \ldots, \phi_s\}$ is the set of internal automaton states, $\underline{\alpha} = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of automaton actions, and, $\underline{\beta} = \{\beta_1, \beta_2, \ldots, \beta_m\}$ is the set of inputs that can be given to the automaton. An output function $\alpha_t = \mathcal{G}[\phi_t]$ determines the next action performed by the automaton given the current automaton state. Finally, a transition function $\phi_{t+1} = \mathcal{F}[\phi_t, \beta_t]$ determines the new automaton state from the current automaton state as well as the response of the Environment to the action performed by the automaton.

Based on the above generic framework, the crucial issue is to design automata that can learn the optimal action when interacting with the Environment. Several designs have been proposed in the literature, and the reader is referred to [4, 6] for an extensive treatment.

2.1 The Item Selector Automaton (ISA)

Our LA based scheme for solving frequent itemset problems is centered around the concept of an *examplar* transaction $T_E \subset I$. With the examplar transaction T_E as a basis, the goal of our scheme is to identify an itemset $X \subseteq T_E$ whose frequency, $freq(X)$, is equal to a specific target frequency γ.

At the heart of our scheme we find an Item Selector Automaton (ISA). In brief, for each item i_j in T_E, a dedicated ISA, based on the Tsetlin automaton [7], is constructed, having:

- States: $\underline{\Phi} = \{-N-1, -N, \ldots, -1, 0, \ldots, N-2, N\}$.

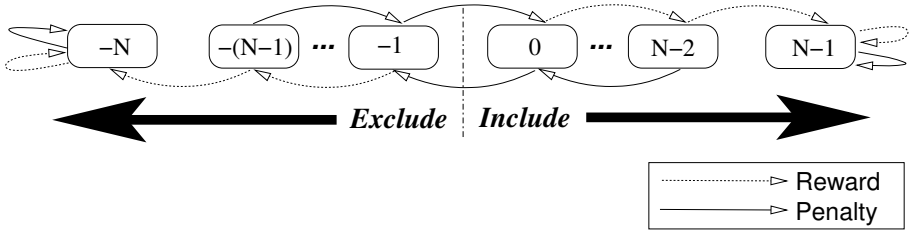

Fig. 2. An ISA choosing between including or excluding an item from candidate frequent itemsets

- Actions: $\underline{\alpha} = \{Include, Exclude\}$.
- Inputs: $\underline{\beta} = \{Reward, Penalty\}$.

Fig. 2 specifies the $\mathcal{G}$ and $\mathcal{F}$ matrices.

The $\mathcal{G}$ matrix can be summarized as follows. If the automaton state is positive, then action *Include* will be chosen by the automaton. If on the other hand the state is negative, then action *Exclude* will be chosen. Note that since we initially do not know which action is optimal, we set the initial state of the ISA randomly to either *'-1'* or *'0'*.

The state transition matrix $\mathcal{F}$ determines how learning proceeds. As seen in the graph representation of $\mathcal{F}$ found in the figure, providing a *reward* input to the automaton *strengthens* the currently chosen action, essentially by making it less likely that the other action will be chosen in the future. Correspondingly, a *penalty* input *weakens* the currently selected action by making it more likely that the other action will be chosen later on. In other words, the automaton attempts to incorporate past responses when deciding on a sequence of actions.

Note that our ISA described above deviates from the traditional Tsetlin automaton in one important manner: State $-N$ and state $N-1$ are absorbing. This allows the ISA to converge to a single state, rather than to a distribution over states, thus artificially introducing an unambiguous convergence criterion.

2.2 Reinforcement Scheme

Since each item i_j in the transaction examplar T_E is assigned a dedicated ISA, ISA_j, we obtain a collective of ISA. The reinforcement scheme presented here is incremental, processing one transaction at a time at discrete time steps. At each time step s, a transaction $T_i \in \mathcal{T}$ is presented to the collective of ISA, whose responsibility is to propose a candidate itemset $X(s)$ for that time step. By on-line processing of the transactions, the goal of the ISA is to converge to proposing an itemset X^* that is supported with frequency, $freq(X^*) = \gamma$, with probability arbitrarily close to 1.

To elaborate, each automaton, ISA_j, chooses between two options at every time step s: shall its own item i_j be included in $X(s)$ or shall it be excluded? Based on the decisions of the ISAs as a collective, a candidate itemset $X(s)$ for

time step s is produced. A response from the Environment is then incurred as follows. First it is checked whether the present transaction T_i supports $X(s)$, and based on the presence or absence of support, each ISA_j is rewarded/penalized according to the following novel reinforcement scheme.

The novel reinforcement scheme that we propose rewards an automaton ISA_j based on the decision of the automaton at time step s and based on whether the present transaction T_i supports the resulting candidate itemset $X(s)$. In brief, if ISA_j decides to include item i_j in $X(s)$, we have two possibilities. If T_i supports $X(s)$, ISA_j is rewarded. On the other hand, if T_i does not support $X(s)$, then ISA_j is randomly penalized with probability $r = \frac{\gamma}{1-\gamma}$. The other decision ISA_j can make is to exclude item i_j from $X(s)$. For that decision, ISA_j is randomly rewarded with probability $r = \frac{\gamma}{1-\gamma}$ if T_i does not support $X(s) \cup \{i_j\}$. On the other hand, if T_i supports $X(s) \cup \{i_j\}$, then the ISA is penalized.

The above reinforcement scheme is designed to guide the collective of learning automata as a whole towards converging to including/excluding items in $X(s)$ so that the frequency of $freq(X(s))$ converges to γ, with probability arbitrarily close to 1.

Note that because multiple variables, and thereby multiple ISA, may be involved when constructing the frequent itemset, we are dealing with a game of LA [4]. That is, multiple ISA interact with the same Environment, and the response of the Environment depends on the actions of several ISA. In fact, because there may be conflicting goals among the ISA involved, the resulting game is competitive. The convergence properties of general competitive games of LA have not yet been successfully analyzed, however, results exists for certain classes of games, such as the Prisoner's Dilemma game [4].

In order to maximize speed of learning, we initialize each ISA randomly to either the state *'-1'* or *'0'*. In this initial configuration, the actions will be switched relatively quickly because only a single state transition is necessary for a switch. Accordingly, the joint state space of the ISA is quickly explored in this configuration. However, as learning proceeds and the ISA move towards their boundary states, i.e., states *'-N'* and *'N-1'*, the exploration calms down. Accordingly, the search for a solution to the frequent itemset problem at hand becomes increasingly focused.

Furthermore, note that we keep a time step counter for each ISA. When a certain cut off threshold has been achieved, we force one of the ISA to converge if it has not yet done so. This enforcement resets the counters of the other ISA, allowing them to adapt to the new configuration. The purpose of this mechanism is to increase convergence speed in ambiguous decision making cases where two different actions provide more or less the same feedback.

3 Empirical Results

In this section we evaluate our proposed scheme using both artificial data as well as data from a real-world network intrusion detection application.

3.1 Artificial Data

For the evaluation on artificial data we constructed a collection of transactions in a manner that by selecting the correct itemset X, one can achieve a frequency, $freq(X)$, of either $0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0$. The purpose is to challenge the scheme by providing a large number of frequency levels to choose among, with only one of these being the target frequency γ that our scheme must converge to. By varying the target frequency γ in the latter range, we also investigate the robustness of our scheme towards low, medium, and high frequency itemsets.

We here report an ensemble average after conducting 100 runs of our scheme. Given a target frequency γ, each run produces an itemset X^*, with X^* being supported by an actual frequency $freq(X^*)$. By comparing the sought target frequency γ with the achieved frequency $freq(X^*)$, the convergence accuracy of our scheme is revealed.

We first study convergence accuracy when any subset $X \subset T_E$ of the examplar transaction T_E have a frequency, $freq(X)$, that is either equal to the target frequency γ, unity (1), or zero (0). Then the goal of the ISA collective is to identify the subset $X \subset T_E$ with frequency γ. As seen in Fig. 3, our scheme achieves this goal with remarkable accuracy.

We observe that for any of the target frequencies γ listed in the figure, on average our ISA collective identifies itemsets X^* with frequencies $freq(X^*) \in \{0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0\}$ that either equals γ or surpasses γ with the least possible amount: $freq(X^*) \geq \gamma \wedge freq(X^*) - 0.125 < \gamma$.

When using a generic transaction examplar T_E instead — one that contains item subsets $X \subseteq T_E$ of any arbitrary frequency level $freq(X) \in \{0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0\}$, the challenge increases. The ISA collective then also have the option to produce frequencies in close vicinity of the target

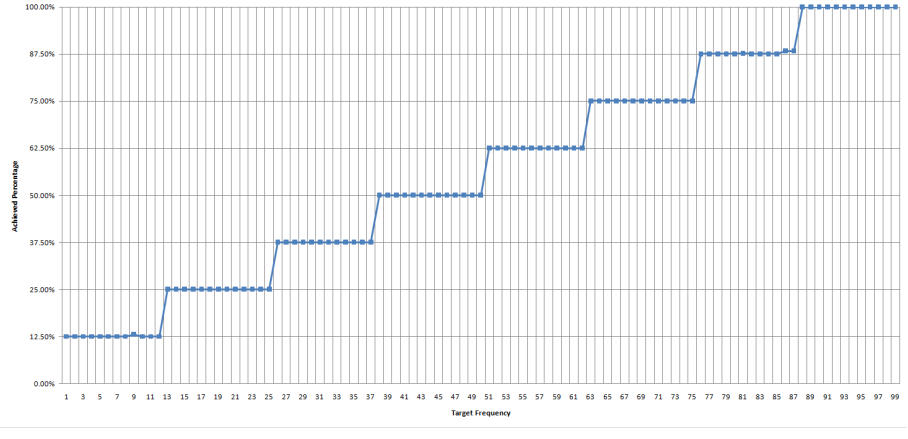

Fig. 3. Achieved percentage of transactions supported by produced itemset (y-axis) using a specific examplar transaction, for varying target frequencies γ (x-axis)

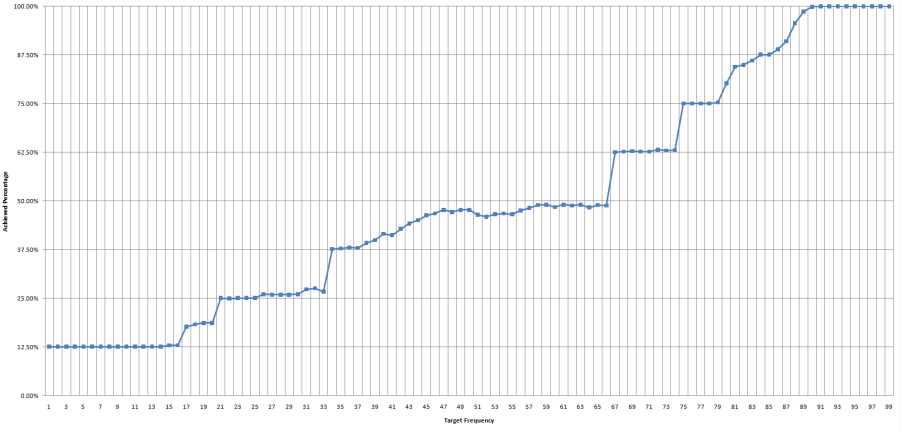

Fig. 4. Achieved percentage of transactions supported by produced itemset (y-axis) using a generic examplar transaction, for varying target frequencies γ (x-axis)

frequency γ. Fig. 4 reports the resulting convergence accuracy, and as seen, it is now more difficult for the collective of ISA to always produce an itemset X with a transaction support frequency exactly equal to γ. Still, the itemsets produced are always close to a nearby neighbor of γ in $\{0.0, 0.125, 0.25, \ldots, 0.75, 0.875, 1.0\}$

3.2 DARPA Intrusion Detection Evaluation Data Set

To evaluate the ISA collective scheme on a real life application, we have implemented a network intrusion detection system, with the ISA collective at its core. Briefly explained, we analyze the last 40 bytes of each network packet header in combination with the first 8 bytes of the transport layer payload, as also done in NETAD [3].[1] Essentially, we see each network packet as a transaction, and byte-value pairs from a network packet are seen as items.

We intend to detect network attacks by first learning a collection of frequent itemsets that describe the key features of normal network traffic – and based on these frequent itemsets, reporting network packets as anomalous when they do not support any of the learned frequent itemsets.

We use the 1999 DARPA Intrusion Detection Evaluation data set [2] for training and testing our system. During training, we use one week of normal traffic data, learning one frequent itemset at a time by randomly picking examplar transactions (network packets) from the normal traffic data. Each time the collective of ISA converges to a new frequent itemset, all network packets that support this itemset are removed from the normal traffic data, and the procedure is repeated to learn each kind of traffic. Note that the granularity of the learned frequent itemset is controlled by γ.

[1] Note that in contrast to NETAD, we analyze both ingoing and outgoing network packets, for greater accuracy.

Table 1. Transaction Examplars

Rule	Attacks
ver+ihl:0x45, frag1:0x40, frag2:0x00, proto:0x06, src-port1:0x00, tcphl:0x50, urgptr1:0x00, urgptr2:0x00	ps
ver+ihl:0x45, dscp:0x00, frag1:0x00, frag2:0x00, proto:0x06, tcphl:0x50, urgptr1:0x00, urgptr2:0x00	ps
ver+ihl:0x45, dscp:0x00, len:0x00, frag1:0x40, frag2:0x00, ttl:0x40, proto:0x06, dstaddr1:0xac, dstaddr2:0x10, dstport1:0x00, dstport2:0x17, tcphl:0x50, recwd1:0x7d, recwd2:0x78, urgptr1:0x00, urgptr2:0x00, pld1:0x00, pld5:0x00, pld7:0x00, pld8:0x00, pld9:0x00	ps, guesstelnet, sendmail

For testing, the second week of the DARPA data set is used. The network packets from this week also contain attacks. If a network packet in the second week of data does not support any of the learned frequent itemsets, it is reported as an anomaly. The complete results of these experiments will be reported in a forthcoming paper, however, Table 1 contains a few representative examples of frequent itemsets, called Rules, and which kind of attacks they allow us to detect. As seen, each Rule consists of selected bytes from a packet, combined with a hexadecimal representation of the corresponding byte value. Thus, considering the first row of the table, network packets of the so-called ps-attack do not support the frequent itemset {ver+ihl:0x45, frag1:0x40, frag2:0x00, proto:0x06, src-port1:0x00, tcphl:0x50, urgptr1:0x00, urgptr2:0x00}, and are therefore reported as anomalies.

Finally, when it comes to computational complexity, note that since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present itemset and the corresponding decisions of the bandit players, the resulting memory footprint is minimal (usually one byte per ISA). Furthermore, computational complexity grows merely linearly with the cardinality of the examplar itemset.

4 Conclusion

In this paper we have addressed frequent itemset mining by the means of a collective of so-called Item Selector Automata (ISA). By on-line interaction with a stream of transactions, the collective of ISA decides which items should be excluded and which should be included in a frequent itemset, with items being chosen from a randomly or manually selected examplar itemset. A novel reinforcement scheme guides the ISA towards finding a candidate itemsets that is supported by transactions with a specified frequency.

Since each bandit player learns simply by updating the state of a finite automaton, and since the reinforcement feedback is calculated purely from the present itemset and the corresponding decisions of the bandit players, the resulting memory footprint is minimal. Furthermore, computational complexity grows merely linearly with the cardinality of the examplar itemset.

In extensive evaluation using both artificial data and data from a real-world network intrusion detection application, we find the results quite conclusive, demonstrating that the ISA collective possesses an excellent ability to find frequent itemsets at various levels of support. Furthermore, the sets of frequent itemsets produced for network intrusion detection are compact, yet accurately describe the different types of network traffic present, allowing us to detect attacks in the form of anomalies.

In our further work, we intend to develop formal convergence proofs for the ISA collective. We are also presently investigating a hierarchical scheme for organizing ISA collectives, with the purpose of increased scalability.

References

1. Han, J., Chen, H., Xin, D., Yan, X.: Frequent pattern mining: current status and future directions. Data Mining and Knowledge Discovery, 55–86 (2007)
2. Lippmann, R., Haines, J., Fried, D., Korba, J., Das, K.: The 1999 DARPA off-line intrusion detection evaluation. Computer Networks 34(4), 579–595 (2000)
3. Mahoney, M.V.: Network traffic anomaly detection based on packet bytes. In: Proceedings of ACM-SAC 2003, pp. 346–350. ACM, New York (2003)
4. Narendra, K.S., Thathachar, M.A.L.: Learning Automata: An Introduction. Prentice Hall, Englewood Cliffs (1989)
5. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (1998)
6. Thathachar, M.A.L., Sastry, P.S.: Networks of Learning Automata: Techniques for Online Stochastic Optimization. Kluwer Academic Publishers, Dordrecht (2004)
7. Tsetlin, M.L.: Automaton Theory and Modeling of Biological Systems. Academic Press, London (1973)
8. Vaarandi, R., Podins, K.: Network ids alert classification with frequent itemset mining and data clustering. In: Proceedings of the 2010 IEEE Conference on Network and Service Management. IEEE, Los Alamitos (2010)
9. Wang, H., Li, Q.H., Xiong, H., Jiang, S.Y.: Mining maximal frequent itemsets for intrusion detection. In: Jin, H., Pan, Y., Xiao, N., Sun, J. (eds.) GCC 2004. LNCS, vol. 3252, pp. 422–429. Springer, Heidelberg (2004)