Journal of Computational Science 24 (2018) 379–388

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Learning Automata Clustering

Mohammad Hasanzadeh-Mofrad [a,*], Alireza Rezvanian [b]

[a] School of Computing and Information, Department of Computer Science, University of Pittsburgh, Pittsburgh, US
[b] School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran

ARTICLE INFO

Article history:
Received 28 February 2017
Received in revised form 8 September 2017
Accepted 14 September 2017
Available online 20 September 2017

Keywords:
reinforcement learning
learning automata
machine learning
clustering algorithm

ABSTRACT

Clustering of data points has been a profound research avenue in the history of machine learning algorithms. Using learning automata which are autonomous decision making entities, in this paper, the learning automata clustering algorithm is proposed. In learning automata clustering, each data point is affiliated with a learning automaton where the learning automaton determines the cluster membership of that data point. The cluster rectification is done through a reinforcement signal for each learning automaton which is fabricated from the Euclidean distance of that data point and the mean value of its designated cluster. Finally, the learning automata clustering is compared with four centroid-based clustering algorithms, K-means, K-means++, K-medians, and K-medoids and results demonstrate the high clustering accuracy and comparable Silhouette coefficient of the proposed method.

1. Introduction

Starting from mid 1950s, researchers have been trying to devise new machine learning algorithms [1] to solve conflated problems using loosely defined computational frameworks. From the beginning of the 21st century machine learning has been thriving and dominating in security, healthcare, robotics, vision, finance, transportation, media, and etc. Deep learning [2] appeared as a subset of machine learning algorithms to learn data representation from large-scale unlabeled data. Furthermore, reinforcement learning [3] introduced another branch of machine learning evolving around the idea of rendering autonomous agents adapting to the given context by adjusting their behavior.

In recent years, one of the mainstreams of machine learning is to use deep learning [2] and reinforcement learning [3] to train bot players for Chess, Checker, Othello, Go, and Poker games where the latter is the toughest one because it involves gambling and bluffing. Some conqueror instances of this breed of algorithms are: Deep Blue [4] which is a Chess bot player developed by IBM, Libratus which is a Poker player developed by Carnegie Mellon University, AlphaGo which is a Go board game bot player and Pong from pixel [5] which is a deep reinforcement learning ATARI games player both developed by Google DeepMind.

As an unsupervised learning method, clustering [6,7], is the exercise of putting similar objects in the same bucket. Clustering is computationally hard and minimizing the clustering objective function is an NP-hard problem. A clustering algorithm aims to find the homogenous objects in a collection of unlabeled objects and assigns them to a same (similar) Cluster. Clustering has applications in computational biology [8] where you want to cluster genes based on their expression patterns, social network [9] for identifying people communities, robotics [10,11], for tracking objects using sensor data, image processing for segmenting an image into distinct regions and extracting objects, solving optimization problems [12] and etc.

Learning automata [13] is a class of reinforcement learning algorithms designed for learning the optimal action via a series of action – response steps to an unknown environment. Learning automata root in control theory where centralized or decentralized or even a mixture of these modes is used to study the behavior of a dynamic system. Throughout a sequence of trials on given inputs, the system will receive positive or negative feedback that modifies the system behavior. Learning automata can be used in a multi-agent environment where multiple interacting intelligent agents are collectively trying to solve a problem. Distributed systems [14–16], image processing [17,18], evolutionary algorithms [19–21], complex networks [22,23], social networks [24,25], sensor networks [26], and Petri net [27,28], are among the example use-cases of learning automata.

In this paper Learning Automata Clustering (LAC) is proposed where first, a learning automaton is assigned to each data sample. Next, each of these learning automata determines the cluster mem-

* Corresponding author.
E-mail addresses: mohammad.hmofrad@pitt.edu (M. Hasanzadeh-Mofrad), a.rezvanian@aut.ac.ir (A. Rezvanian).

https://doi.org/10.1016/j.jocs.2017.09.008
1877-7503/

bership of their designated data sample. Then, the new means are computed, and finally, by comparing the new centroids (means) with the last ones, a reinforcement signal is calculated for each learning automaton to serve as a positive or negative feedback for the current cluster assignment. LAC and four other clustering algorithms including K-means [29], K-means ++ [30], K-medians [31], and K-medoids [32] are evaluated using 12 standard University of California Irvine (UCI) clustering benchmarks [33] including 12 small-scale benchmarks and two large-scale benchmarks. Experimental results showed that LAC has a decent performance for clustering of data points and it can easily be extended to any research domain having a clustering task without needing any special change in the input data.

The reminder of this paper is organized as follows. Section 2 presents the related work, different types of clustering algorithms and the basics of learning automata theory. Section 3 shows how to hinge learning automata into a clustering problem and gives detailed description of the key steps of LAC algorithm. Section 4 starts with an analytical evaluation of LAC algorithm for parameter selection and follows a comparison of four clustering algorithms with LAC. Section 5 discusses the results and future work. Section 6 provides a final summary.

2. Related Work

In this section, first a subset of traditional clustering algorithms is introduced, then, the existing learning automata based clustering algorithms are surveyed, and lastly, the preliminary theory of learning automata is introduced

2.1. Types of Clustering Algorithms

Clustering is the process of congregating n data samples into k clusters where sample x_i belongs to the nearest cluster C_j with the lowest distance between itself and j_{th} cluster center c_j (or the most similar cluster C_j with the highest similarity measure). The common measure of choosing the nearest cluster (widely used in K-means clustering algorithm) is the smallest Euclidean distance where data point x_i is compared to the j_{th} cluster representative c_j. Therefore, given k clusters, the cluster with the smallest Euclidean distance is the nearest cluster.

Clustering algorithms have different analytic categories. Examples of these categories include:

- Centroid-based clustering where each cluster is represented using a cluster center which may not belong to the data points. *K-means* [29], *K-means++* [30] and *K-medians* [31] are examples of this kind of clustering where in K-means the mean is calculated for each cluster to determine its centroid and in K-medians the median is calculated instead. Moreover, K-means++ is similar to K-means except for clusters center initialization step which is done by spreading the initial clusters across the entire dataset. Also, *K-mediods* [32] is another example of centroid-based clustering algorithms where unlike K-means and K-medians, K-mediods chooses data points as centers for clusters.
- Hierarchical clustering [34] is designed around the idea of similar objects with closer distance are more related to each other compare to distant objects. It can be divided into two approaches: *Agglomerative clustering* which puts each data point in a cluster and merges the clusters on the way up to the tip of hierarchy, and *divisive clustering* which starts with one cluster for the entire dataset and splits it recursively in the way down the hierarchy. Generally, the result of a hierarchical clustering algorithm is a dendogram showing the configuration of clusters.
- Model-based clustering aims to put data points with the same distribution within a same cluster. While this category of clustering algorithms has a rich statistical foundation, it suffers from overfitting, unless applying specific constraints to the model. To alleviate this issue, the *expectation maximization* [35] can be used to learn the model parameters where the data is modeled with a fixed number of Gaussian distribution that are initialized at random.
- Density-based clustering clusters data points based on identifying dense clusters of points. Thus, data points located in areas with higher density are ended up in the same cluster. *Mean-shift* [36] is one of the well-known density-based clustering approaches. It uses the kernel density estimation to locate the maxima of the density function and assign data points to the densest local maxima in their vicinity. Mean-shift uses kernel width as a parameter to control the tradeoff between the number of clusters and their densities. *Density-based spatial clustering of applications with noise* (*DBSCAN*) [37] is another density-based clustering algorithm which uses ε – neighborhoods to construct clusters of size at least *MinPts* (the minimum number of points required to constitute a cluster).

To wrap up this section, there are also other clustering algorithms which are worth mentioning:

- Unlike K-means, K-medians and K-medoids clustering algorithms, *affinity propagation* [38] is a clustering algorithm based on message passing which does not require that the number of clusters be defined prior to running the algorithm. Similar to K-medoids, affinity propagation models the dataset using a small set of *exemplar* which are the most promising points to represent the cluster centers. It works well for a small number of clusters, however, it is not advised to be used when having large number of clusters.
- Spectral clustering[39] is a simple yet efficient clustering algorithm which has application in image segmentation. In spectral clustering a similarity matrix is used which shows the similarity of each pair of data points. Using the eigenvalues of this similarity matrix, the spectral clustering, first reduces dimensionality of the problem space and then runs K-means algorithm in lower dimensions. It is worth noting that, prior running of spectral clustering, the number of clusters should be specified.
- Balanced iterative reducing and clustering using hierarchies (BIRCH) [40] is a hierarchical clustering suitable for clustering of large datasets. Given a dataset, first, it builds a characteristic feature tree from the dataset. Next, the algorithm scans all the leaf entries rebuilding a smaller tree. Then, a clustering algorithm, e.g. agglomerative clustering is used to cluster leafs. Last, the new cluster centroids are used as seeds to redistribute data points among clusters. BIRCH is an incremental clustering that does not need to scan the entire dataset in each iteration. Thus, minimizing the I/O cost.

2.2. Clustering Algorithms and Applications

Narrowing down our search for clustering algorithms, we found the following clustering algorithms which use learning automata or some other similar frameworks in order to perform clustering.

Neural networks have been widely adapted for data clustering. As a supervised learning approach, neural networks cannot directly be used to solve a clustering problem, yet researchers use them to produce an expressive representation of the input data that can be used for data clustering. Besides using the standard Hebb rule for radial basis function neural network to determine the weight changes, in [41] learning automata are used to learn the weights (synaptic delays for firing time) of spiking neurons. The neural

network delays are modeled using probability vectors of learning automata and the evaluations showed that the new neural network achieves higher clustering precision compare to K-means.

Cellular learning automata [42] is a combination of learning automata and cellular automata [43]. It is an interconnected grid of learning automata where each learning automaton is evaluated individually and collectively based on its local and neighbors' actions. In [44], a clustering algorithm based on cellular learning automata and an evolutionary algorithm is proposed. Learning automata and their taking actions are used as the model genome and string genome of an evolutionary computation algorithm, respectively. The algorithm tries to minimize the squared error measure where in each step, the cellular learning automata generate new genomes by taking new actions until a termination condition is met.

Firefly algorithm is an evolutionary algorithm inspired by flashing behavior of fireflies. In [45], The firefly algorithm is combined with K-means algorithm for data clustering. In each iteration of this algorithm, fireflies move and attract other members of the group toward the global optimum which is determined using K-means clustering. In the evaluation, it is reported that the proposed algorithm outperforms its counterparts with low intra-cluster distance.

Artificial fish swarm algorithm is another evolutionary algorithm derived from the movement of herd of fish. In [46], the initial solutions for the artificial fish swarm algorithm are initialized using K-means algorithm and then the algorithm carries out the clustering task. In order to achieve the clustering goal, a new fitness function is used which minimizes the sum of intra-cluster distance and lets artificial fish swarm algorithm to determine the cluster centers. Based on the results, the proposed method has small error rate and sum of intra-cluster distance while taking comparable number of steps to its rival algorithms for finding the clusters.

Irregular cellular learning automata are cellular learning automata with loosely couple structures that can be non-rectangular. In [26], a wireless sensor network clustering algorithm is proposed to cluster network nodes. This algorithm has two phases: initial clustering and local reclustering of network nodes. Based on their evaluation, the proposed algorithm saves more energy and has high clustering quality in terms of number of clusters and percentage of sparse clusters.

A Peer-to-Peer network is an unconstructed overlay network that distributes the workload between equally privileged network nodes. In landmark clustering, the landmarks are a subset of data points where a linear combination of them will represent the original data. In [47], learning automata are used to select landmark peers for each cluster while minimizing the total communication cost. From the results, the algorithm decreases the communication delay and average round trip time between clusters.

2.3. Learning Automata Theory

Learning automata [48] are probabilistic decision making elements. Having a series of interactions with the environment, they iteratively adopt to the environment and learn the optimal action. A widespread type of learning automata is variable structure learning automata which is defined by a quadruple $[\alpha, \beta, P, T]$ where

- $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ is the set of actions where r is the number of actions.
- β is the reinforcement signals where in a P-model environment $\beta \in \{0, 1\}$.
- $P = \{p_1, p_2, \ldots, p_r\}$ is the set of actions probability.
- T is the learning algorithm where in the n_{th} step, $p(n+1) = T[p(n), \alpha(n), \beta(n)]$ is linear, if $p(n+1)$ is a linear function of $p(n)$, or nonlinear if $p(n+1)$ is a nonlinear function of $p(n)$.

In n_{th} step of a linear learning algorithm, if the i_{th} selected action $\alpha_i(n)$ receives the reward reinforcement signal $\beta(n) = 0$, the corresponding probability vector of learning automaton $p(n+1)$ is updated using (1). If it receives the penalty reinforcement signal $\beta(n) = 1$, the corresponding probability vector of learning automaton $p(n+1)$ is updated using (2) [49]:

$$p_j(n+1) = \begin{cases} p_j(n) + \mathrm{a}\left(1 - p_j(n)\right) & \mathrm{j} = \mathrm{i} \\ p_j(n)(1-a) & \forall \mathrm{j} | \mathrm{j} \neq \mathrm{i} \end{cases} \quad (1)$$

$$p_j(n+1) = \begin{cases} p_j(n)(1-b) \; j = i \\ \dfrac{b}{(r-1)} + (1-b)p_j(n) \quad \forall j | j \neq i \end{cases} \quad (2)$$

where in (1) and (2), a and b are learning parameters (reward and penalty parameters). Different values for a and b creates different learning algorithms:

- If $a = b$, the learning algorithm will be of type Linear Reward-Penalty (L_{RP})
- If $b = 0$, the learning algorithm will be of type Linear Reward-Inaction (L_{RI})
- If $a \gg b$, the learning algorithm will be of type Reward-epsilon-Penalty ($L_{R\varepsilon P}$)

3. Learning Automata Clustering

Mimicking the act of clustering using reinforcement learning requires answering questions about where to put learning automata and how to provide a meaningful feedback from the environment for them. Having a close look at the legacy of an example clustering algorithm like K-means, we will answer these questions in the following.

3.1. Learning Automata Placement

Similar to a typical clustering algorithm, LAC aims to group the similar data points into a same cluster. Here, we choose to assign a learning automaton to each data instance for two reasons:

1. Input size: The input size of a clustering algorithm is a function of the number of input samples and the number of feature dimensions. Considering the fact that the number of feature dimensions contribute to the complexity of the input data and does not generally bias the clustering task itself, we assign a learning automaton per instance to cover the entire clustering data space.
2. Number of clusters: Mounting a learning automaton atop of each data sample and trying to put it inside the most coherent cluster, the task of cluster assignment can be intuitively mapped into the action set of a learning automaton where each action represents picking a specific cluster for the corresponding data sample.

The schematic view of the mapping from the problem space to the clustering space utilizing learning automata is illustrated in Fig. 1. In this figure, the two above mentioned constraints for placing learning automata are followed.

3.2. Reinforcement Signal Representation

No reinforcement learning algorithm can achieve a decent performance without receiving proper feedback from the environment it is working on. After placing learning automata in the clustering problem space, now it is time to design a meaningful reinforcement signal to show learning automata how good was their latest actions. In this way, we devise three different reinforcement signals, but

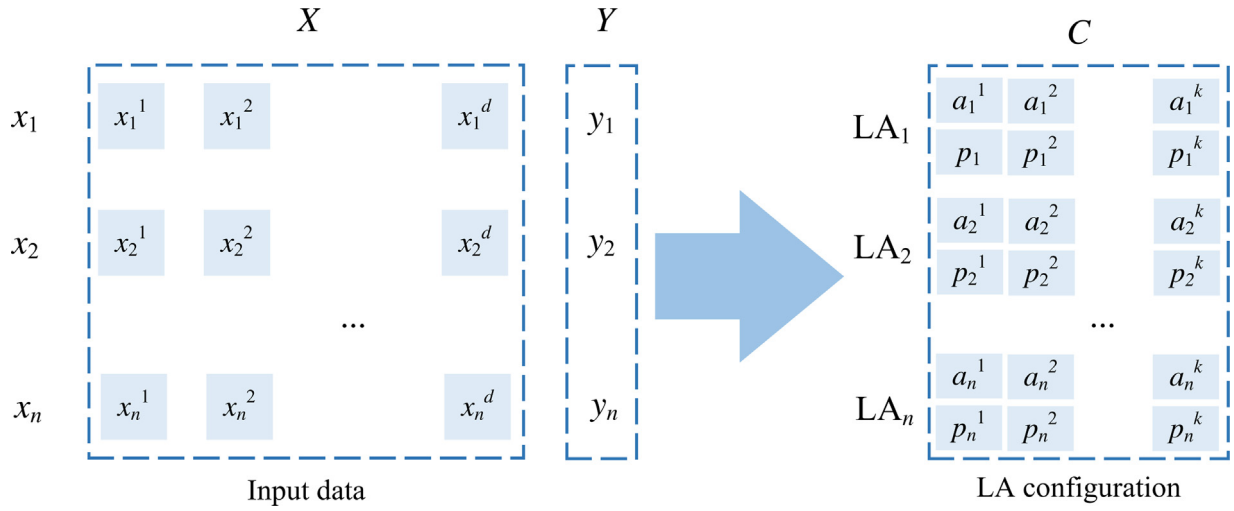

Fig. 1. How to place Learning Automata (LA) atop of an input dataset where a learning automaton is associated with a data sample. $X_{n\ x\ d} = [x_1, x_2, \ldots, x_n]$ and $Y_{n\ x1} = [y_1, y_2, \ldots, y_n]$ are the input data and labels, where n is the number of input samples and d is the number of input features. C is the clustering space with k available clusters. $A_{1x\ k} = [a_1, a_2, \ldots a_k]$ and $P_{1x\ k} = [p_1, p_2, \ldots p_k]$ are the action set and probability vector of an example automaton where k is the number of actions which is the same as the number of clusters.

finally in LAC implementation, we used the last one because it led to the highest accuracy among these signals. Assuming that a set of learning automata determine an example cluster configuration, then a sample reinforcement signal can be calculated as follows:

1. Calculating the new cluster mean and compare it to the last one; if it was less than the previous cluster's mean, it means that the cluster configuration is improved, and all of the learning automata contributed on creating this cluster receives the reward signal. This reinforcement signal is not a good choice and it is too crude to be used because it is focusing on decreasing the cluster mean without paying attention to individuals forming the cluster.
2. Computing the Euclidean distance between each data sample and cluster mean; if this distance was less than the previous one, it is assumed that the cluster assignment is improved, and all learning automata constitute forming this cluster receives the reward signal. Even though this signal has a limited knowledge about individual learning automaton progress, it cannot quantify the goodness of clustering because it does not measure the quality of the entire group of learning automata which are creating the cluster.
3. Calculating the Euclidean distance between each data point and the previous cluster mean and save it as an *exemplar cluster configuration* followed by calculating the new cluster mean based on the current cluster configuration which is opted out by learning automata in the current iteration. Then, for each data sample, we compare its exemplar and learning automaton cluster assignment, if they were the same, the corresponding learning automaton receives the reward signal, otherwise it receives penalty signal. Here, the exemplar clusters act as the beacon clusters configuration while constructing the reinforcement signal.

Among the above reinforcement signals, although the first two signals and especially the second one seems reasonable, the backlash effect of not properly transmitting the merit of action is dominant, that only the third one achieves permissible results. Furthermore, the third reinforcement signal not only measures the virtue of the current actions of learning automata constituting a cluster, but it also extrapolates the previous actions.

3.3. Pseudo-code of Learning Automata Clustering

LAC consists of two primary parts:

1. Initialization step (Algorithm 1): (a) Setting the number of clusters, reading the input dataset and determining the number of instances, and dimensions, (b) Initializing the clusters mean and exemplar clusters assignment and the maximum number of iterations, (c) Initializing learning parameters a and b and the number of actions r, and (d) Initializing learning automata probability vector, action set, and reinforcement signal vector.
2. Main loop (algorithm 2): (a) For each data sample the SELECT function of the corresponding learning automaton is called. The SELECT function has a roulette wheel implementation which uses the probability vector of the learning automaton. Thus, the odds of picking a cluster for the current data sample is determined by spinning this roulette wheel, (b) For each data point, compute the exemplar cluster membership based on its previous cluster mean using MINI function. The MINI function returns the cluster index using the minimum Euclidean distance between the current data point and clusters mean, (c) For each cluster, update cluster mean using the MEAN function while taking account of the current cluster assignment which is selected by learning automata in the current iteration, (d) For each learning automaton, compare the current selected action with the exemplar cluster assignment and calculate the reinforcement signal. If these two values were the same, a learning automaton receives reward signal, otherwise, it receives penalty signal, (e) After calculating the reinforcement signal, the UPDATE function will be called. Using the current learning automata configuration and the calculated reinforcement signal, the UPDATE function will update the learning automata probability vector, and (f) Finally, if the sum of squared error of the current clusters mean and the previous ones is less than or equal zero, we assume that learning automata converge to a stable clusters configuration. The alternative stopping criterion is to reach the maximum allowed number of iterations.

Algorithm 1 and 2 are the pseudocode of initialization step and main loop of LAC.

Algorithm 1: Learning Automata Clustering (LAC) – Initialization step

```
Input: READ(data)
BEGIN INIT
  SET X and Y    // (a) Data samples and labels
  SET n, d, and k // (a) #data samples, #dimensions, and #clusters
  SET M          // (b) M_{k x d} Clusters mean
  SET C          // (b) C_{1 x n} Exemplar clusters assignment
  SET t_max = 100 // (b) #iterations
  SET a = 0.45   // (c) Reward signal
  SET b = 0.09   // (c) Penalty signal
  SET r = k      // (c) #actions
  INIT LA action set              // (d) A_{n x k} = 0
  INIT LA Probability vector      // (d) P_{n x k} = 1/a
  INIT LA reinforcement signal    // (d) S_{n x 1} = 0
END INIT
```

Algorithm 2: Learning Automata Clustering (LAC) – Main loop

```
BEGIN MAIN
  WHILE TRUE
    // (a) Select an action based on current Probability distribution (P)
    FOR each sample i
      A_i = SELECT(LA_i)
    END FOR
    // (b) Reassign exemplar clusters
    FOR each sample i
      C_i = MINI(X_i, M)
    END FOR
    // (c) Update clusters mean
    FOR each cluster j
      M_j = MEAN(X_{A=j})
    END FOR
    // (d) Calculate reinforcement signal S for LA
    FOR each sample i
      IF A_i == C_i THEN
        S_i = 0
      ELSE
        S_i = 1
      END IF
    END FOR
    // (e) Update probability reinforcement signal (S)
    // using (1) and (2)
    FOR each sample i
      P_i = UPDATE(LA_i)
    END FOR
    // (f) M_t and M_{t-1} are current and previous clusters means
    t = t + 1
    IF M_{t-1} == M_t OR t > t_max THEN
      BREAK
    END IF
  END FOR
END MAIN
```

3.4. LAC Analysis

Originally inspired from K-means, learning automata clustering maps an unsupervised learning family of algorithms into a reinforcement learning type of algorithms and this mapping comes with a cost. The cost of having a reinforcement signal which indirectly measures the merit of clusters membership. Given the Monte Carlo simulation is used to implement K-means; and K-means does not always produce the optimal results, LAC also presents the same Monte Carlo characteristics where it gives a good approximate of the optimal cluster configuration, but the results are not always optimal. So, one may need to run LAC more than one time in order to see the optimal result.

The loose way to calculate the running time of K-means is to apply the Lloyd's algorithm which is a Monte Carlo method if some randomization added to the algorithm. Using Lloyd's algorithm, the worst-case time complexity of K-means is O($ndki$) where n is the number of instances, d is the number of dimensions, k is the number of clusters and i is the number of iterations. Following this loose linear upper bound, we can calculate the worst-case time complexity for LAC algorithm. First, let us disassemble it into the following parts:

1. Selecting an action for all learning automata: O(nk)
2. Computing distance between two data points: O(d)
3. Reassigning exemplar clusters: O(nkd)
4. Updating clusters means: O(nd)
5. Calculating reinforcement signal: O(n)
6. Updating probability vector of all learning automata: O(nk)
7. Running the entire algorithm: O(i) where i is not a very large number

Last, Putting the above information in (3), we can calculate the worst-case time complexity of LAC. Since O(nkd) dominates the rest of the terms in (3) and LAC is executed i times, the worst-case time complexity of the algorithm is O($ndki$) which is linear.

$$i(\mathrm{O}(nk) + \mathrm{O}(d) + \mathrm{O}(nkd) + \mathrm{O}(nd) + \mathrm{O}(n) + \mathrm{O}(nk)) = \mathrm{O}(ndki) \quad (3)$$

4. Experiments

In this section, first, a heatmap is used to select appropriate values for learning parameters of learning automata a and b. Then, detailed graphs on how the clusters configuration is converged to a steady state are depicted. These graphs show how learning automata actions converge to an optimal action (corresponding to the optimal cluster configuration). Finally, the evaluation results of LAC, K-means, K-means++, K-medians, and K-medoids on 14 UCI benchmarks [33] are reported. For all experiments the maximum number of allowed iterations is 100. Furthermore, alongside its counterpart algorithms, LAC is implemented in Python3.4 with extensive use of NumPy package and the implementation is open sourced (https://goo.gl/fPEzM5).

4.1. UCI Datasets

14 real world datasets are picked from UCI machine learning repository [33]. Since some of the datasets have missing values or characters as class labels, they are preprocessed before they can be used. For example, for datasets with missing values, the incomplete instances are removed, and for datasets with characters as class labels, the labels are changed to integers. Table 1 shows the characteristics of these datasets after preprocessing.

4.2. Learning Automata Parameter Selection

A heatmap is a graphical representation of data where each value in a matrix is shown using a unique color. It is a powerful 2-dimensional data analysis tool that presents rows of data using a series of cells each containing a value and a corresponding color.

The performance of learning automata largely depends on a and b learning parameters (a and b parameters used in (1) and (2)). Since

Table 1
Selected datasets properties and corresponding learning automata configurations.

Type	Dataset	#instances	#dimensions	#clusters	LA #actions	LA initial prob.
Small-scale datasets	Balance Scale	625	4	3	3	1/3
	BCW	683	10	2	2	1/2
	CMC	1473	9	3	3	1/3
	Glass	214	10	6	6	1/6
	Hayes-Roth	132	5	3	3	1/3
	Ionosphere	351	34	2	2	1/2
	Iris	150	4	3	3	1/3
	Pima	768	8	2	2	1/2
	Segmentation	2310	19	7	7	1/7
	Sonar	208	60	2	2	1/2
	Soybean	47	35	4	4	1/4
	Wine	178	13	3	3	1/3
Large-scale datasets	Drift	13910	129	6	6	1/6
	HAR	10299	561	6	6	1/6

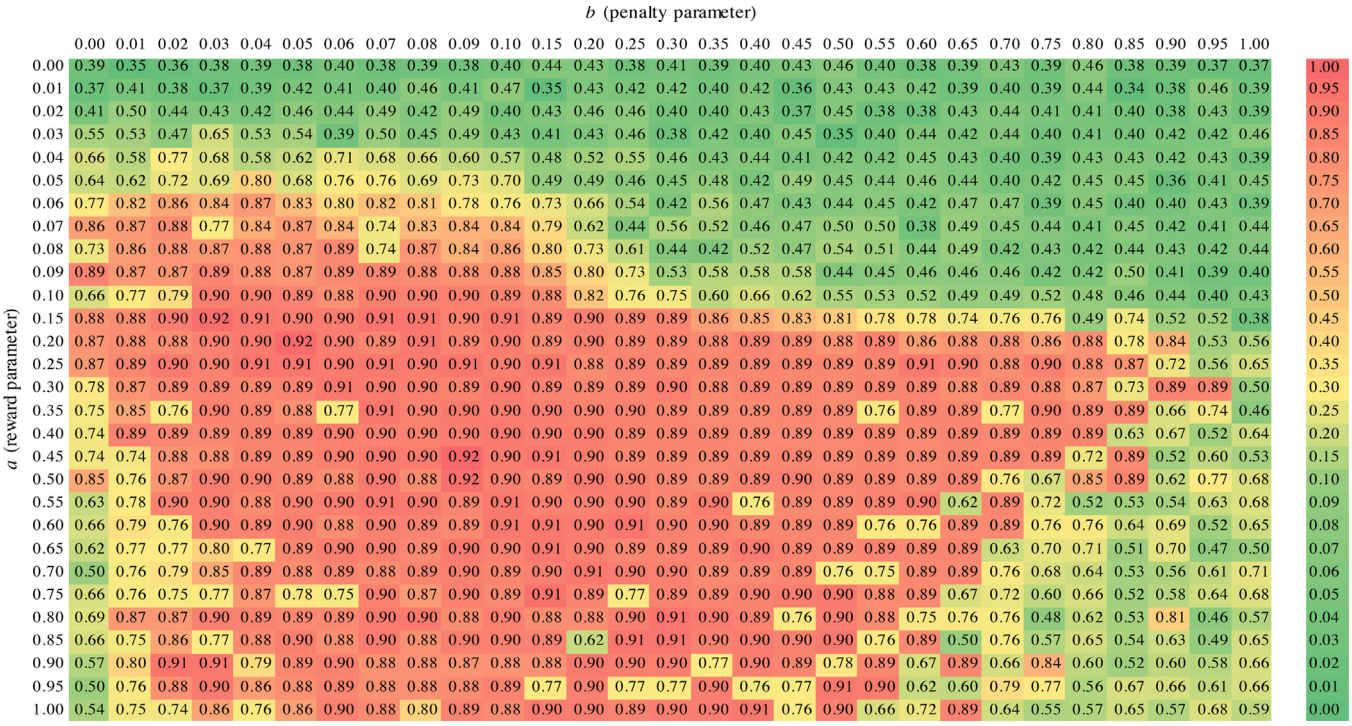

Fig. 2. The heatmap of average accuracy of LAC with different a and b learning parameters. The rightmost bar is the colormap which indicates the range of accuracies from 0 to 1.

these parameters can vary in range of [0,1], first this range is discretized using 0.01 steps for values between 0 and 0.1 (a and $b \in [0,0.1]$) and 0.05 steps for values between 0.1 and 1 (a and $b \in (0.1, 1]$). Next, for each of these combinations, the average LAC accuracy is calculated. Finally, a heatmap is drawn from the resultant accuracies which is shown in Fig. 2.

The accuracies which are reported in Fig. 2 belongs to Iris dataset [33]. For each cell, LAC is ran three times and its average accuracy is calculated. The best results belong to $(a, b) = \{(0.15, 0.03), (0.2, 0.05), (0.45, 0.09), (0.5, 0.09)\}$ with accuracy of 0.92. Therefore, for the rest of experiments, $a = 0.45$ and $b = 0.09$ is picked because they belong to the middle of the reddish area which has a high concentration of promising accuracies. Following heatmap results as a tool for parameter selection, we make sure that a useful set of parameters for the rest of experiments is used.

4.3. Underlying Profiling of Learning Automata Clustering

In this experiment, we aim to show the underlying behavior of learning automata. In the first part of this experiment, a learning automaton will be picked at random and it will be shown that how its probability vector converges to the optimal action which is the act of assigning the dominant cluster number with the highest probability to its associated data sample (Fig. 3(a)). It will next be shown that how the average probability vector of learning automata constituting clusters assignment evolves throughout a sequence of iterations (Fig. 3(b)). For both of these experiments, the Iris dataset is used which has 3 clusters. Also, using the findings from the previous sub-section, the a and b learning parameters of learning automata are set to 0.45 and 0.09, respectively.

Fig. 3(a) shows how a randomly picked learning automaton gradually learns to pick the cluster number for the data sample it is mounted on. The learning automaton has 3 actions corresponding to the number of clusters in the Iris dataset. Thus, the probability vector is initialized with 0.33 (=1/3) for all 3 actions. From Fig. 3(a), after 37 out of 100 iterations, the learning automaton action set converges to the action which is represented the cluster number 1. Starting from iteration 1 to 12, the probability of choosing cluster 1 is monotonically increasing, and from iteration 12 to 37 the probability of choosing this cluster number is almost frozen and about

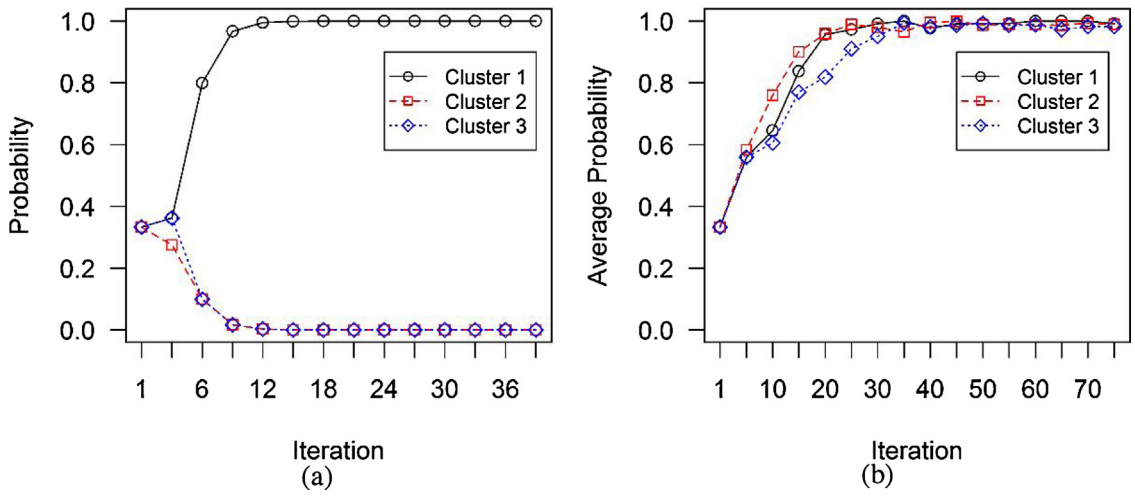

Fig. 3. (a) The process of converging a learning automaton for assigning the cluster number to its mounted data sample. (b) The process of converging learning automata for assigning clusters to the entire dataset.

Table 2
Average clustering accuracy. The best results are boldfaced.

Type	Dataset	K-means	K-means++	K-medians	K-medoids	LAC
Small-scale datasets	Balance Scale	0.43 ± 0.07	0.42 ± 0.06	0.35 ± 0.06	0.42 ± 0.07	**0.53 ± 0.09**
	BCW	0.50 ± 0.00	**0.54 ± 0.00**	0.50 ± 0.00	0.45 ± 0.00	0.45 ± 0.00
	Sonar	0.49 ± 0.06	0.49 ± 0.06	0.48 ± 0.04	0.49 ± 0.05	**0.51 ± 0.04**
	CMC	0.33 ± 0.04	0.32 ± 0.00	0.30 ± 0.04	**0.35 ± 0.03**	0.34 ± 0.02
	Glass	0.37 ± 0.16	0.35 ± 0.20	0.39 ± 0.14	0.41 ± 0.12	**0.44 ± 0.03**
	Hayes-Roth	0.36 ± 0.03	0.41 ± 0.02	0.35 ± 0.03	0.37 ± 0.03	**0.42 ± 0.02**
	Ionosphere	0.46 ± 0.03	0.48 ± 0.00	0.46 ± 0.03	0.48 ± 0.01	**0.48 ± 0.00**
	Iris	0.58 ± 0.21	0.78 ± 0.16	0.55 ± 0.16	0.71 ± 0.17	**0.90 ± 0.01**
	Pima	0.49 ± 0.00	**0.51 ± 0.00**	0.49 ± 0.00	0.51 ± 0.01	0.50 ± 0.00
	Wine	0.52 ± 0.04	0.60 ± 0.10	0.49 ± 0.16	0.62 ± 0.09	**0.68 ± 0.01**
Large-scale datasets	Drift	0.15 ± 0.01	0.23 ± 0.03	**0.24 ± 0.01**	0.22 ± 0.03	0.19 ± 0.01
	HAR	0.16 ± 0.00	0.17 ± 0.00	0.17 0.01	0.17 ± 0.01	**0.18 ± 0.00**

1. On the other hand, the probabilities of choosing clusters 2 and 3 are monotonically decreasing with slight variations in the last 20 last iterations.

The pattern which is depicted by Fig. 3(a) shows that the devised reinforcement signal is working properly and after taking 12 iterations the learning automaton decides on assigning the cluster 3 to the associated instance with a high confidence and this decision is not changed for the rest of iterations (from 12 to 37). Thus, this assignment also complies with the configuration of cluster 1. Since the utilized reinforcement signal is implemented in a way to keep the clusters mean as small as possible, the merit of a cluster configuration is conveyed to each learning automaton constituting it by sending the reward or penalty signal. Therefore, the probability of retaking the former action will be prompted by receiving these reward signals.

Fig. 3(b) shows how the pack of learning automata converges and creates the final cluster configuration on a single sweep for Iris dataset. Since we have 3 clusters in Iris dataset, each learning automaton has 3 actions with initial probabilities of 0.33 (= 1/3). In each iteration of Fig. 3(b), we take the average probability of all learning automata constituting these 3 clusters and plot them. The average probabilities start with 0.33 for all 3 clusters and evolve during the first 30 iterations and converge to 1 for the rest of iterations (from 30 to 75). This shows that the learning automata composing clusters come to an agreement for clusters assignment and learn the optimal cluster configuration. There are no subtle changes in average probabilities from iteration 30 to iteration 75 because the learning automata almost converge to the optimal clusters configuration. It is worth note that compare to Fig. 3(a) where a learning automaton took 37 iterations to converge, for the group of learning automata in Fig. 3(b), it takes 80 iterations to converge. Since except for stopping criterion, there is no intervention in the process of converging the learning automata, this is the rationale behind the difference between the number of iterations in these two figures belonged to two separate runs.

4.4. Performance Analysis of Clustering Algorithms

In this experiment, 14 UCI machine learning datasets [33] are used to compare LAC with four state of the art clustering algorithms including K-means, K-means++, K-medians, and K-medoids. The details of these datasets are reported in Table 1. Also, the mean and standard deviation of average clustering accuracy and clustering Silhouette coefficient of 10 runs of these algorithms are reported in Tables 2 and 3, respectively. The mapping between real clusters and final cluster assignment is shown in Table 4. Moreover, based on our observation from Section 4.2, a and b learning parameters are set to 0.45 and 0.09, respectively. Finally, the initial probability distribution of learning automata action set follows the same distribution of Table 1.

For computing clustering accuracy or formally *cluster purity*, each cluster is assigned to the class with the highest number of points, and then the clustering accuracy is calculated by comparing the ground truth and predicted labels using (4) [50]:

$$\text{Accuracy} = \frac{1}{n}\sum_{j=1}^{k}\max_{i}|t_i \cap c_j| \ \in [0, 1] \tag{4}$$

Table 3
Average clustering Silhouette coefficient.

Type	Dataset	K-means	K-means++	K-medians	K-medoids	LAC
Small-scale datasets	Balance Scale	-0.10 ± 0.05	-0.16 ± 0.01	-0.10 ± 0.03	-0.11 ± 0.04	-0.13 ± 0.01
	BCW	-0.18 ± 0.02	-0.24 ± 0.39	0.88 ± 0.00	-0.46 ± 0.00	-0.18 ± 0.02
	Sonar	0.10 ± 0.23	0.03 ± 0.13	0.12 ± 0.25	0.03 ± 0.19	-0.15 ± 0.06
	CMC	-0.02 ± 0.33	0.20 ± 0.00	0.04 ± 0.43	-0.12 ± 0.09	-0.15 ± 0.07
	Glass	-0.14 ± 0.22	-0.02 ± 0.25	-0.06 ± 0.36	-0.34 ± 0.21	-0.24 ± 0.02
	Hayes-Roth	-0.07 ± 0.23	-0.36 ± 0.08	-0.01 ± 0.27	-0.24 ± 0.10	-0.30 ± 0.01
	Ionosphere	-0.04 ± 0.20	-0.31 ± 0.00	-0.01 ± 0.23	-0.03 ± 0.44	-0.31 ± 0.01
	Iris	0.02 ± 0.43	-0.21 ± 0.25	0.18 ± 0.42	-0.04 ± 0.32	-0.06 ± 0.11
	Pima	0.81 ± 0.11	-0.04 ± 0.00	0.75 ± 0.17	-0.28 ± 0.20	-0.12 ± 0.07
	Wine	0.09 ± 0.23	0.21 ± 0.13	0.21 ± 0.23	-0.17 ± 0.16	-0.30 ± 0.03
Large-scale datasets	Drift	0.54 ± 0.35	-0.20 ± 0.13	-0.20 ± 0.14	-0.22 ± 0.11	-0.06 ± 0.13
	HAR	0.53 ± 0.32	-0.07 ± 0.01	0.11 ± 0.19	0.13 ± 0.14	0.13 ± 0.11

Table 4
Comparison between SOM, associative CLA and LAC on cluster assignment. $k \rightarrow c_i$ means cluster k has data of cluster c_i. CRA and PRA stand for Completely Randomized Assignment and Partial Randomized Assignment, respectively.

Dataset (#clusters)	SOM (assigned #clusters) [52]	Associative CLA (assigned #clusters) [52]	LAC (assigned #clusters)
Ionosphere (2)	(6) CRA	(3) CRA	(2) $1 \rightarrow c_1, c_2$ $2 \rightarrow c_1, c_2$
Iris (3)	(6) CRA	(2) $1 \rightarrow c_1$ $2 \rightarrow c_2, c_3$	(3) $1 \rightarrow c_1$ $2 \rightarrow c_1, c_2$ $3 \rightarrow c_1, c_2$
Segment (7)	(7) $1 \rightarrow c_2$ $2 \rightarrow c_4$ $3 \rightarrow c_7$ $4 \rightarrow c_6$ $5 \rightarrow c_3, c_5$ $6 \rightarrow c_2, c_4, c_6$ $7 \rightarrow$ PRA	(6) $1 \rightarrow c_2$ $2 \rightarrow c_7$ $3 \rightarrow c_7$ $4 \rightarrow c_4, c_6$ $5 \rightarrow c_1, c_3, c_5, c_7$ $6 \rightarrow c_1, c_3, c_4, c_5$	(7) $1 \rightarrow$ PRA $2 \rightarrow$ PRA $3 \rightarrow$ PRA $4 \rightarrow c_1, c_3, c_4, c_5$ $5 \rightarrow$ PRA $6 \rightarrow c_6$ $7 \rightarrow c_1, c_3, c_4, c_7$
Soybean (4)	(5) $1 \rightarrow c_1$ $2 \rightarrow c_1$ $3 \rightarrow c_1$ $4 \rightarrow c_2$ $5 \rightarrow c_3, c_4$	(2) $1 \rightarrow c_1, c_2$ $2 \rightarrow c_3, c_4$	(4) $1 \rightarrow c_3$ $2 \rightarrow c_3$ $3 \rightarrow c_1, c_2, c_4$ $4 \rightarrow c_1, c_2, c_4$
Wine (3)	(9) CRA	(2) $1 \rightarrow c_1$ $2 \rightarrow c_2, c_3$	(3) $1 \rightarrow c_2, c_3$ $2 \rightarrow c_1, c_2, c_3$ $3 \rightarrow c_1, c_3$

where, n is the number of samples, k in the number of clusters, j is the cluster index where c_j is the j_{th} cluster center, and i is the class index where t_i is the class with maximum number of samples within j_{th} cluster.

Table 2 presents the average clustering accuracy of running clustering algorithms on 12 UCI datasets. For 7 out of 10 small-scale datasets, LAC has the highest accuracy with comparably low standard deviation among clustering algorithms showing that the proposed algorithm can effectively and robustly cluster datasets with different characteristics. The small-scale datasets have different number of samples, dimensions and clusters. Thus, LAC outperformance exhibits that dataset diversity does not deteriorate its performance and it can efficiently carry out the clustering task almost independent of the nature of dataset. Although this is not a general observation for small-scale datasets, but one can see that even though LAC is not the best algorithm while clustering CMC and Pima datasets, it yields comparable results. The only case that LAC could not compete with its counterparts is CMC datasets where LAC is lagging behind K-means++ with 10% difference.

In the second half of Table 2, for large-scale datasets where the number of instances and dimensions are drastically larger, LAC gets the best results for clustering HAR dataset with 561 dimensions, However, LAC could not achieve good results for drift dataset with 129 dimensions. Since the aim of having these large-scale datasets is to show that how LAC will perform on complex datasets, having the best results for HAR which has the highest number of dimension shows that LAC can effectively cluster exponentially large datasets as well.

The first observation of Table 2 is that except for Wine and Iris datasets that LAC has an exceedingly high clustering accuracy, the clustering accuracy is not high enough among clustering algorithms. This is due to the sparse nature of the rest of datasets where the centroid based clustering algorithm cannot perform well and especially for LAC the cluster mean is not a proper way to enforce the reinforcement signal.

There is another observation in Table 2 that demonstrates the utility of learning automata. As an illustration, a byproduct of the difference between performance of the LAC on Iris and Glass Identification dataset is that learning automata are suitable for problems with less number of required decisions (clusters). Problems like binary yes/no or trinary yes/maybe/no questions would leverage the small number of learning automaton actions which can converge way faster compare to problems which require large number of actions. Having a look at the last column of Table 1, large number of actions means that the initial probabilities have smaller values which makes the process of finding the optimal action slower and harder.

In Table 3, we report the average Silhouette coefficient over all clusters. The Silhouette ranges from −1 to +1 and measures how well each instance lies within its cluster. Silhouette coefficient is usually used for determining the number of clusters. The average Silhouette coefficient over a cluster shows how tightly grouped all data samples within a cluster while the average Silhouette coefficient over all clusters measures how densely configured the overall clusters assignment. The high value of Silhouette near 1 shows that a data sample is perfectly matched to its assigned cluster and poorly matched to the other ones. On the other hand, the Silhouette near −1 shows that it was more appropriate to cluster a data sample in its neighboring cluster. Finally, Silhouette near 0 means that the datum is around the border of two similar clusters showing a low inter-cluster distance. The Silhouette coefficient is calculate using (5) [51]:

$$s_i = \frac{(b_i - a_i)}{\max(b_i,\ a_i)},\ -1 \le s_i \le +1 \tag{5}$$

where a_i is the average dissimilarity distance between the i_{th} data point and its assigned cluster and b_i is the lowest dissimilarity distance between the i_{th} data point and any other cluster except for its assigned cluster.

Table 3 shows the average Silhouette coefficient over all clusters for all datasets. Silhouette coefficient on itself is not a proper to metric to evaluate the cluster distribution. Thus, we need to have a cross analysis between clustering accuracies (Table 2) and Silhouette coefficients (Table 3).

From Table 3, the LAC Silhouette coefficient is negative for all datasets except for HAR dataset and the lowest one belongs to Iris dataset. For Iris dataset, combining the 90% accuracy of LAC from Table 2 with negative Silhouette coefficient values of Table 2, we can conclude that the clustering accuracy is directly contributed to having a comparable Silhouette coefficient. Due to the different logic backed different clustering algorithms, this observation is not a universal one. For example, although K-means and K-medians achieve the same result for clustering BCW dataset, their Silhouette coefficient is −0.18 and 0.88, respectively which is far different. This observation shows that even while having the same clustering accuracies, depending on the clustering strategy different algorithms may have positive or negative Silhouette coefficients.

Associative Cellular Learning Automata (CLA) [52] is a new CLA based algorithm which in addition to the conventional reinforcement signal it receives extra information from the environment. This new founded algorithm has applications in clustering. An interesting aspect of clustering is to figure out the mapping between real clusters and resultant clusters from the algorithm. Thus, extending one of the experiments of [52], Table 4 shows an application of associative CLA, Self Organizing Map (SOM) which is neural network, and LAC in data clustering where $k \rightarrow c_i$ demonstrates the mapping between real cluster number k and cluster detected by the algorithm c_i. There is no quantifiable metric to exactly evaluate the performance of algorithms in this experiment. Though from the results of Table 4, LAC can find a decent mapping between real clusters and final clusters assignment having comparable results to associative CLA.

5. Discussion and Future Work

This paper presents an existence proof of the capability of learning automata to integrate with a clustering problem. The proposed algorithm has the highest numerical clustering accuracy with a comparable standard deviation among the clustering methods for 8 out of 12 datasets (10 small-scale and 2 large-scale). While the proposed algorithm shows a relatively negative Silhouette coefficient for utilized datasets and decent clustering mapping, it is important to note that we are not concentrating on tuning the number of clusters to achieve better performance, yet we are trying to enable a wide range of possible applications for this algorithm in machine learning research. It is worth mentioning that compare to other clustering algorithms presented in this paper, LAC has comparable overhead in terms of runtime.

Unlike most of learning automata based clustering algorithms where learning automata is used for an application of clustering [26], [44], in this work, learning automata is exclusively used to solve the opaque problem of data clustering. While developing this algorithm, we address two folds of challenges including the intuitive placement of learning automata for a clustering problem and the design of an effective reinforcement signal.

LAC can be easily improved by introducing new reinforcement signals. The current reinforcement signal is based on clusters mean which may not be a good choice for datasets with relatively sparse instances. Also, given LAC is implemented using Euclidean distance, plugging new distance functions like Mahalanobis, Manhattan, or Minkowski distance to LAC may improve the performance. One of the future avenues of research for LAC would be to investigate the impact of new distribution-based or density-based reinforcement signals with new distance functions. The datasets which are used in this work are acting as a proof of work and without any required changes, LAC can also be applied to other datasets.

LAC is a centroid-based clustering where the number of clusters should be known prior running the algorithm and fixed while processing the data. Future work will likely focus on using learning automata with variable number of actions to support density-based and distribution-based clustering as well as supporting problems with unknown or changing number of clusters.

6. Conclusion

In this paper, the implementation and evaluation of the Learning Automata Clustering (LAC) is presented. The algorithm is designed to adaptively assign clusters to data samples while utilizing the mean distance of clusters members as an improvement signal. The proposed algorithm consists of adding a learning automaton atop of each data sample where the learning automaton has an action set corresponding to the number of clusters. An evaluation of parameter setting is provided for the proposed algorithm and it is shown that how the underlying cluster configuration is constructed by learning automata interactions during the runtime. Finally, it is shown that LAC can provide a better numerical accuracy on 8 out of 12 well-known UCI datasets compare to other existing centroid-based clustering algorithms.

Acknowledgments

Authors would like to thank anonymous reviewers for their insightful comments.

References

[1] R.S. Michalski, J.G. Carbonell, T.M. Mitchell, Machine learning: An artificial intelligence approach, Springer Science & Business Media, 2013.
[2] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436–444.
[3] R.S. Sutton, A.G. Barto, Reinforcement learning: An introduction, vol. 1, Cambridge Univ Press, 1998.
[4] M. Campbell, A.J. Hoane, F. Hsu, Deep blue, Artificial intelligence 2 (2002) 57–83.
[5] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, D. Hassabis, Human-level control through deep reinforcement learning, Nature 518 (no. 7540) (2015) 529–533.
[6] F. Gullo, G. Ponti, A. Tagarelli, G. Tradigo, P. Veltri, 'A time series approach for clustering mass spectrometry data', Journal of Computational Science 3 (no. 5) (2012) 344–355.

[7] T. Hillerman, J.C.F. Souza, A.C.B. Reis, R.N. Carvalho, Applying clustering and AHP methods for evaluating suspect healthcare claims, Journal of Computational Science 19 (2017).
[8] E. Elsebakhi, F. Lee, E. Schendel, A. Haque, N. Kathireason, T. Pathare, N. Syed, R. Al-Ali, Large-scale machine learning based on functional networks for biomedical big data with high performance computing platforms, Journal of Computational Science 11 (2015) 69–81.
[9] C.A. Bliss, M.R. Frank, C.M. Danforth, P.S. Dodds, 'An evolutionary algorithm approach to link prediction in dynamic social networks', Journal of Computational Science 5 (no. 5) (2014) 750–764.
[10] N. Kherici, Y. Mohamed Ben Ali, Using PSO for a walk of a biped robot, Journal of Computational Science 5 (5) (2014) 743–749.
[11] M. Komosinski, Applications of a similarity measure in the analysis of populations of 3D agents, Journal of Computational Science 21 (2017).
[12] T.T. Ngo, A. Sadollah, J.H. Kim, A cooperative particle swarm optimizer with stochastic movements for computationally expensive numerical optimization problems, Journal of Computational Science 13 (March) (2016) 68–82.
[13] K.S. Narendra, M. Thathachar, Learning Automata: A Survey, Systems, Man and Cybernetics, IEEE Transactions on (4) (1974) 323–334.
[14] M. Hasanzadeh, M.R. Meybodi, Grid resource discovery based on distributed learning automata, Computing 96 (September (9)) (2014) 909–922.
[15] M. Hasanzadeh, M.R. Meybodi, Distributed optimization Grid resource discovery, The Journal of Supercomputing 71 (January (1)) (2015) 87–120.
[16] M. Hasanzadeh Mofrad, O. Jalilian, A. Rezvanian, M.R. Meybodi, Service level agreement based adaptive Grid superscheduling, Future Generation Computer Systems 55 (February) (2016) 62–73.
[17] M. Hasanzadeh, S. Sadeghi, A. Rezvanian, M.R. Meybodi, Cellular edge detection: Combining cellular automata and cellular learning automata, AEU - International Journal of Electronics and Communications 69 (2015).
[18] D. Ganguly, M.H. Mofrad, A. Kovashka, Detecting Sexually Provocative Images, 2017 IEEE Winter Conference on Applications of Computer Vision (WACV) (2017) pp. 660–668.
[19] M. Hasanzadeh, M.R. Meybodi, S.S. Ghidary, Improving Learning Automata based Particle Swarm: An optimization algorithm, 2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI) (2011) pp. 291–296.
[20] M. Hasanzadeh, M.R. Meybodi, M.M. Ebadzadeh, Adaptive cooperative particle swarm optimizer, Applied Intelligence 39 (September (2)) (2013) 397–420.
[21] M. Hasanzadeh, S. Sadeghi, A. Rezvanian, M.R. Meybodi, Success rate group search optimiser, Journal of Experimental & Theoretical Artificial Intelligence 0 (November (0)) (2014) 1–17.
[22] A. Rezvanin, M. Rahmati, M.R. Meybodi, Sampling from complex networks using distributed learning automata, Physica A: Statistical Mechanics and its Applications, Physica A: Statistical Mechanics and its Applications (2013).
[23] A. Rezvanian, M.R. Meybodi, Sampling social networks using shortest paths, Physica A: Statistical Mechanics and its Applications 424 (2015) 254–268.
[24] A. Rezvanian, M.R. Meybodi, Stochastic graph as a model for social networks, Computers in Human Behavior 64 (November) (2016) 621–640.
[25] A. Rezvanian, M.R. Meybodi, A new learning automata-based sampling algorithm for social networks, Int. J. Commun. Syst (January) (2015), p. n/a-n/a.
[26] M. Esnaashari, M. Meybodi, A cellular learning automata based clustering algorithm for wireless sensor networks, Sensor Letters 6 (5) (2008) 723–735.
[27] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Learning Automata-Based Adaptive Petri Net and Its Application to Priority Assignment in Queuing Systems With Unknown Parameters, IEEE Transactions on Systems, Man, and Cybernetics: Systems 45 (no. 10) (2015) 1373–1384.
[28] S.M. Vahidipour, M.R. Meybodi, M. Esnaashari, Adaptive Petri net based on irregular cellular learning automata with an application to vertex coloring problem, Appl Intell 46 (March (2)) (2017) 272–284.
[29] J. MacQueen, et al., Some methods for classification and analysis of multivariate observations, Proceedings of the fifth Berkeley symposium on mathematical statistics and probability 1 (1967) 281–297.
[30] D. Arthur, S. Vassilvitskii, k-means ++: The advantages of careful seeding, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (2007) pp. 1027–1035.
[31] A.K. Jain, R.C. Dubes, Algorithms for clustering data, Prentice-Hall, Inc., 1988.
[32] L. Kaufman, Clustering by means of medoids, in: *Statistical data analysis based on the L1-norm and related methods*, 2017.
[33] M. Lichman, UCI machine learning repository (2013).
[34] S.C. Johnson, Hierarchical clustering schemes, Psychometrika 32 (3) (1967) 241–254.
[35] A.P. Dempster, N.M. Laird, D.B. Rubin, Maximum likelihood from incomplete data via the EM algorithm, Journal of the Royal Statistical Society. Series B (Methodological) (1977) 1–38.
[36] K. Fukunaga, L. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Transactions on Information Theory 21 (1) (1975) 32–40.
[37] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, et al., A density-based algorithm for discovering clusters in large spatial databases with noise, Kdd 96 (1996) 226–231.
[38] B.J. Frey, D. Dueck, Clustering by passing messages between data points, Science 315 (5814) (2007) 972–976.
[39] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (8) (2000) 888–905.
[40] T. Zhang, R. Ramakrishnan, M. Livny, BIRCH: an efficient data clustering method for very large databases, ACM Sigmod Record 25 (1996) 103–114.
[41] P. Adibi, M.R. Meybodi, R. Safabakhsh, Unsupervised learning of synaptic delays based on learning automata in an RBF-like network of spiking neurons for data clustering, Neurocomputing 64 (2005) 335–357.
[42] M. Mozafari, M.E. Shiri, H. Beigy, A cooperative learning method based on cellular learning automata and its application in optimization problems, Journal of Computational Science 11 (November) (2015) 279–288.
[43] J. Wąs, G.C. Sirakoulis, Cellular Automata Applications for Research and Industry, Journal of Computational Science 11 (November) (2015) 223–225.
[44] R. Rastegar, M. Rahmati, M.R. Meybodi, A Clustering Algorithm Using Cellular Learning Automata Based Evolutionary Algorithm, in: Adaptive and Natural Computing Algorithms, Springer, 2005, pp. 144–150.
[45] T. Hassanzadeh, M.R. Meybodi, A new hybrid approach for data clustering using firefly algorithm and K-means, Artificial Intelligence and Signal Processing (AISP), 2012 16th CSI International Symposium on (2012) pp. 007–011.
[46] D. Yazdani, B. Saman, A. Sepas-Moghaddam, F. Mohammad-Kazemi, M.R. Meybodi, A new algorithm based on improved artificial fish swarm algorithm for data clustering, International Journal of Artificial Intelligence™ 11 (A13) (2013) 193–221.
[47] A.M. Saghiri, M.R. Meybodi, A distributed adaptive landmark clustering algorithm based on mOverlay and learning automata for topology mismatch problem in unstructured peer-to-peer networks, International Journal of Communication Systems (2015).
[48] M. Thathachar, P.S. Sastry, Varieties of learning automata: an overview, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 32 (no. 6) (2002) 711–722.
[49] K.S. Narendra, M.A. Thathachar, Learning automata: an introduction, Courier Corporation, 2012.
[50] C.D. Manning, P. Raghavan, H. Schütze, et al., Introduction to information retrieval, vol. 1, Cambridge University Press, Cambridge, 2008.
[51] P.J. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, Journal of Computational and Applied Mathematics 20 (1987) 53–65.
[52] M. Ahangaran, N. Taghizadeh, H. Beigy, Associative cellular learning automata and its applications, Applied Soft Computing 53 (April) (2017) 1–18.

Mohammad Hasanzadeh-Mofrad received the B.E. degree in Software Engineering from Payame Noor University (PNU), Birjand, Iran, in 2009 and the M.E. degree in Artificial Intelligence from Amirkabir University of Technology, Tehran, Iran in 2013. He is currently pursuing the Ph.D. degree in Computer Science at the Computer Science Department of University of Pittsburgh, Pittsburgh, USA. His research interests include high performance computing, operating system, machine learning, and computational intelligence.

.**Alireza Rezvanian** was born in Hamedan, Iran, in 1984. He received the B.Sc. degree from Bu-Ali Sina University of Hamedan, Iran, in 2007, the M.Sc. degree in Computer Engineering with honors from Islamic Azad University of Qazvin, Iran, in 2010, and the Ph.D. degree in Computer Engineering at the Computer Engineering & Information Technology Department from Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, in 2016. Currently, he works as a researcher in School of Computer Science, Institute for Research in Fundamental Sciences (IPM), Tehran, Iran. Prior to the current position, he joined the Department of Computer Engineering and Information Technology at Hamedan University of Technology, Hamedan, Iran as a lecturer. He has authored or co-authored more than 60 research publications in reputable peer-reviewed journals and conferences including IEEE, Elsevier, Springer, Wiley and Taylor & Francis. He has organized various special sessions in international conferences in his area of expertise in Iran and abroad. He is a member of Technical Program Committees (TPCs) of various IEEE sponsored conferences in Iran and abroad. He has been a reviewer of many reputable conferences and journals. His research activities include soft computing, evolutionary algorithms, complex social networks, and learning automata.