Engineering Applications of Artificial Intelligence 68 (2018) 121–130

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Sparse Self-Represented Network Map: A fast representative-based clustering method for large dataset and data stream

Zhen Liu *, Qiuhua Zheng, Zhongping Ji, Weihua Zhao

School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, 310018, China

ARTICLE INFO

Keywords:
Fast clustering
Sparse Self-Represented
Dynamic sparse initialization
Image recognition

ABSTRACT

The demand of fast clustering increases rapidly as we keep collecting tremendously large amount of data in the last decade. In this paper, we propose a nonparametric and representative-based Sparse Self-Represented Network Map for fast clustering on large dataset. Each node in the network generates a heat map for the dataset by receiving stimulations from data within its Accepting Field. We developed a weight adjusting method to learn and summarize the clustering pattern of the data. Such learned map is used for computing clustering results, by breaking weak links and finding connected components Rather than employing an iterative process to find local minima, our network passes the dataset only once and is able to capture the global pattern of the dataset as well as detecting natural number of clusters. As a nonparametric method, we propose Sparse Dynamic Instantiation to avoid the curse of dimensionality, namely a node or a link is instantiated only when stimulated by input data. As a result, the overall complexity is linear to the data dimension. Our algorithm is tested on synthetic and real datasets and compare with popular clustering algorithms (K-means++, Expectation–Maximization, Mean-Shift and StreamKM++) as well as state-of-art clustering algorithm (Affinity Propagation and Density Peak). We also applied our clustering algorithm to mobile location clustering, building a Visual Dictionary for image recognition, and clustering data streams. Our experiments indicate that our algorithm can be a better alternative for all compared popular clustering algorithms especially when efficiency is the primary consideration, namely we drastically improve time and space complexity but retain equal level of accuracy.

1. Introduction

Clustering is a preliminary and fundamental process for many sophisticated systems (Jain, 2010; Kuila and Jana, 2014). Many successful clustering algorithms has been proposed for different domains of application, and people are still searching for more accurate and efficient algorithms for clustering (Silva et al., 2013; Bifet et al., 2010; Yuwono et al., 2016), especially for Big Data Analysis (Manyika et al., 2011) and Data Streams (Silva et al., 2013). In general, the clustering problem is studied in four different ways (Han et al., 2011): representative-based clustering, density-based clustering, hierarchical clustering, and graph-based clustering. Representative-based clustering (Hartigan and Wong, 1979; Arthur and Vassilvitskii, 2007; Bailey and Elkan, 1994; Celebi et al., 2013) seeks representative centers to capture the clusters in the sense that data points in the same cluster are more similar to each other. Density-based clustering (Kriegel et al., 2011; Amini et al., 2014) tries to find dense areas of arbitrary shape and typically takes no consideration on data similarities. In other words, instead of considering similarity of data points, density-based clustering problems focus more on the geometric distribution of data in low dimensionality. Hierarchical clustering algorithms (Murtagh and Legendre, 2014) usually generate a clustering tree representing the parental relationship among micro clusters. Graph-based clustering (Linh and Long, 2014) specialized on graph data and datasets that can be converted into graphs. Among all these four clustering problems, representative-based clustering is the most common one and is our focus in this paper.

We primarily focus on improving trade-off between speed and accuracy for large dataset and conducting applications that can be benefitted from it. Existing state-of-art algorithms achieved satisfactory clustering results, such as Affinity Propagation (AP) (Frey and Dueck, 2007), Density Peak Clustering (DP) (Rodriguez and Laio, 2014) and Dirichlet Process Clustering (DPC) (Blei and Jordan, 2006). However, these methods are too expensive for even moderate sized datasets. Since AP and DP are quadratic with respect to both time and space, DPC is quadratic with respect to dimension, their use is limited to small datasets

* Corresponding author.
E-mail address: liuzhen@hdu.edu.cn (Z. Liu).

https://doi.org/10.1016/j.engappai.2017.11.002
Received 27 September 2016; Received in revised form 9 September 2017; Accepted 2 November 2017
0952-1976/

despite accurate. Another kind of clustering algorithms achieves tolerable results and are linear to time, space, and dimension, such as Kmeans (Hartigan and Wong, 1979; Arthur and Vassilvitskii, 2007; Bailey and Elkan, 1994), Expectation–Maximization (EM) (Bailey and Elkan, 1994), Fuzzy C-means (Celebi et al., 2013), Mean Shift (Comaniciu and Meer, 2002), and DP-means (Kulis and Jordan, 2011). For this reason, they are popularly applied to solve real world problems. Moreover, this kind of algorithms are easy to scale out for parallel computing (Bradley et al., 1998; Chitta et al., 2011). However, those algorithms require tens of iterations to converge. This is fine for moderate sized datasets, but their performance is drastically worsened when dealing with datasets which are too large to be fitted into main memory, since they have to load the entire dataset block by block for each iteration with intensive memory swaps.

To solve the I/O problem described above, recent researches are active on Data Streams. In Data Stream Domain, data arrives one by one with fixed order since the dataset is too large which made random access unaffordable (Silva et al., 2013; Bifet et al., 2010). This topic can be trace back to earlier works such as (Zhang et al., 1996; Aggarwal, 2003; O'callaghan et al., 2002) , as well as recent works in density-based clustering (Amini et al., 2014; Cao et al., 2006; Chen and Tu, 2007) trend discovery (Kranen et al., 2011) and scalability (Bradley et al., 1998). Sculley (2010) developed a heuristic data stream method to approximate Kmeans algorithm. Since (Sculley, 2010), more researches addresses approximating popular algorithms in more efficient manners. Ackermann et al. (2012) and Fichtenberger et al. (2013) achieved this by generating a subset of the original data (known as Coreset), and perform Kmeans++ on the reduced "Coreset". They proved that clustering centers from the Coreset can approximate the original method with concrete boundary. Avrithis and Kalantidis (2012) presented a method to boost assignment time for Gaussian Mixture Model and consequently an approximation of EM clustering. Gong et al. (2015), Avrithis et al. (2015) adopted advanced hashing technique for membership assignment and therefore developed an approximation method for Kmeans. Obviously, though outperformed their predecessors, these approximation methods will always be outperformed by their original versions (algorithms that they are trying to approximate).

Among all the representative-based clustering algorithms, K-means and its variances are the most commonly used and approximated. The idea backing these algorithms is known as Expectation–Maximization (EM) (Bailey and Elkan, 1994). The algorithms take K initial centers and perform expectation and maximization iteratively. The EM theory guarantees the lower bond of objective function to be non-decreasing (Bishop, 2006). However, this is far from satisfaction because the goal is to diminish the gap between lower bond and the global optima. In this paper, we developed a single pass fast clustering algorithm to improve the accuracy for large dataset and data streams. We aim at outperforming popular but expensive algorithms (e.g. Kmeans and EM) that have been approximated by data stream algorithms, but still runs as fast as most data stream algorithms.

Our secondary focus is detecting number of cluster automatically since such information is usually unavailable for very large datasets. For the most popular algorithms, e.g. K-means++ (Arthur and Vassilvitskii, 2007), EM (Bailey and Elkan, 1994) and Fuzzy C-means (Havens et al., 2012; Mungle et al., 2013), the number of clusters is pre-defined as a user-specified parameter. However, in most cases the number of clusters is what we seek. Detecting number of clusters accurately becomes more challenging when we expect the algorithm to be efficient. Some sophisticated clustering algorithms, like Affinity Propagation(AP) (Frey and Dueck, 2007) and Density Peak (DP) (Rodriguez and Laio, 2014) and Dirichlet Process Clustering (DPC) (Blei and Jordan, 2006), are able to find accurate number of clusters automatically, but they are computationally infeasible for moderate or large datasets. Few fast clustering algorithms is able to perform clustering without knowing the number of clusters beforehand. BIRCH (Zhang et al., 1996) is a typical one which partitions the data into blocks but no natural clustering pattern can be captured. In this paper, we propose a fast clustering method that is able to detect the accurate number of clusters and runs as fast as SreamKM++ while the clustering accuracy is as good as AP and DP.

In this paper, we propose a representative-based clustering algorithm to tackle with above problems. The algorithm employs the idea of Artificial Neural Network such as computing units and weighted connections between these units. However, our computing unit model (known as a node or a neuron) is quite different from traditional Artificial Neural Network. It is a single pass clustering algorithm and the speed can be fully boosted by parallelization. The general model is non-parametric, but the proposed Sparse Dynamical Instantiation solved the curse of dimensionality and makes it linear to data dimension in computation complexity. Following are the highlights of our algorithm:

(1) Efficiency: we provide better trade-off between speed and accuracy than all compared algorithms. Our algorithm achieves state-of-art clustering accuracy with hundreds of times of speedup.
(2) Ability: our algorithm finds the number of clusters automatically and provide accurate clustering centers by a single pass of the dataset.
(3) Novelty: we propose a new Artificial Neural Network model with entirely new unsupervised training method which makes Efficiency and Ability possible.

In the rest of this paper, Section 2 discusses the technical detail about Sparse Self-Represented Network Map and its unsupervised learning method. Section 3 shows how to use the network to cluster data points. Section 4 presents the simulations on synthetic data and we apply our method to 3 real world engineering problems in Section 5. Finally, we present our findings and conclusions in Section 6.

2. Sparse Self-Represented Network Map for fast clustering

2.1. Network map model and unsupervised training method

In this paper, we designed a grid-structured network for fast clustering. The network is constructed by connected neurons, where each neuron in the network receives stimulus from data that fall into its Accepting Field (AF). AF is a continuous subspace of real value which can be presented as a square in two dimensions or a cubic in three dimensions. AF can be defined as soft or hard. For hard Accepting Field, the neuron takes binary stimulation which is 1 if a data point falls into its accepting field and 0 otherwise. In this paper, we use soft AF. Let $S(\boldsymbol{x}|\boldsymbol{\mu}_i, \sigma)$ be the stimulation from data point $\boldsymbol{x}$ to neuron i, we define:

$$S\left(\boldsymbol{x}|\boldsymbol{\mu}_i, \sigma\right) = \begin{cases} \exp \dfrac{-\left(\boldsymbol{x}-\boldsymbol{\mu}_i\right)^2}{\sigma^2} & if \ \ |\boldsymbol{x}-\boldsymbol{\mu}_i| \le \sigma \\ 0 & otherwise \end{cases} \tag{1}$$

where $\boldsymbol{\mu}_i$ is the center of the accepting field of neuron i and σ is a parameter controls the width of the accepting field. Each neuron has its own center while they all sharing a common σ. In other words, the weight between input data and the network depends on the value of the data points.

The network is stimulated by data points sequentially where we denote the tth data point as $\boldsymbol{x}_t$. The procedure of processing tth data point is known as step t or time t. The activation value $a_i(t)$ is cut down by 1 in each step no matter it is stimulated or not. Let the output of neuron i at time t be $y_i(t)$, then

$$a_i(t) = y_i(t-1) - 1 + \alpha \cdot S\left(\boldsymbol{x}_t|\boldsymbol{\mu}_i, \sigma\right) \tag{2}$$

$$y_i(t) = \begin{cases} 0 & if \ a_i(t) < 0 \\ a_i(t) & if \ 0 \le a_i(t) \le \theta \\ \theta & if \ a_i(t) > \theta \end{cases} \tag{3}$$

where α is a user-specified parameter controls the learning rate and θ is the upper threshold of the output of a neuron. The active value of