2007

S-means: Similarity Driven Clustering and Its application in Gravitational-Wave Astronomy Data Mining

Hansheng Lei
The University of Texas Rio Grande Valley

Lappoon R. Tang
The University of Texas Rio Grande Valley

Juan R. Iglesias
The University of Texas Rio Grande Valley

Soma Mukherjee
The University of Texas Rio Grande Valley, soma.mukherjee@utrgv.edu

Soumya Mohanty
The University of Texas Rio Grande Valley

S-means: Similarity Driven Clustering and Its application in Gravitational-Wave Astronomy Data Mining

Hansheng Lei[1], Lappoon R. Tang[1], Juan R. Iglesias[1]
Soma Mukherjee[2], and Soumya Mohanty[2]

[1] Computer Science Department
[2] The Center for Gravitational Wave Astronomy
The University of Texas at Brownsville
Brownsville TX 78520, USA
hansheng.lei@utb.edu

Abstract. Clustering is to classify unlabeled data into groups. It has been well-researched for decades in many disciplines. Clustering in massive amount of astronomical data generated by multi-sensor networks has become an emerging new challenge; assumptions in many existing clustering algorithms are often violated in these domains. For example, K means implicitly assumes that underlying distribution of data is Gaussian. Such an assumption is not necessarily observed in astronomical data. Another problem is the determination of K, which is hard to decide when prior knowledge is lacking. While there has been work done on discovering the proper value for K given only the data, most existing works, such as X-means, G-means and PG-means, assume that the model is a mixture of Gaussians in one way or another. In this paper, we present a similarity-driven clustering approach for tackling large scale clustering problem. A similarity threshold T is used to constrain the search space of possible clustering models such that only those satisfying the threshold are accepted. This forces the search to: 1) explicitly avoid getting stuck in local minima, and hence the quality of models learned has a meaningful lower bound, and 2) discover a proper value for K as new clusters have to be formed if merging them into existing ones will violate the constraint given by the threshold. Experimental results on the UCI KDD archive and realistic simulated data generated for the Laser Interferometer Gravitational Wave Observatory (LIGO) suggest that such an approach is promising.

1 Introduction

Clustering is unsupervised classification of unlabeled data, which has been a well-researched problem in many disciplines. A large portion of clustering algorithms were developed by computer scientists but much motivation came from applications of an interdisciplinary nature. It is common for modern applications in business data mining, physics, astronomy and environmental sciences to deal with a large amount of data.

While many clustering algorithms are available and work well in small scale data sets [9, 12, 16], comparative study showed that only K-means and its variants are suited for mining very large data sets [15, 22]. The K-means method is more computationally efficient than other commonly used clustering methods such as hierarchical clustering [22, 7] and Kohonen's self organizing map (SOM) [14]. When data set size is large, hierarchical clustering and SOM can be computationally prohibitive.

However, K-means also has its own weaknesses: i) sensitive to initial partition, ii) converge to local minima, iii) the number of cluster, K, must be determined before hand, and iv) outliers from the centroid may pull the centroid away from the real one. Initial partition problem can be alleviated by repeating different initial seed settings. Local convergence is still an open problem. But in most cases, locally optimal solution is satisfactory if global optimization is too costly. Discovering K is a big weakness and several algorithms have been proposed to tackle the problem. The X-means algorithm was presented to learn K [19]. This algorithm tries many values of K and uses Bayesian Information Criterion (BIC) to score each resulting model. The K that produces the highest BIC score is chosen. Besides BIC, other scoring systems, such as Akaike Information Criterion [3] and Minimum Description Length [20] can be applied. X-means is a straightforward extension of regular K-means. The difficulty it faces is: how many K values should be chosen and compared? When the data set is large and data distribution is non-trivial, the range of possible number of clusters can be large.

Addressing problems in X-means where overfitting of data can occur, the G-means algorithm [13] is proposed to grow K from a small number. A statistical normality test is applied to each cluster to see whether it has high confidence of Gaussian distribution. If not, split the current cluster into two clusters and continue with the statistical test for the rest of the clusters. Like X-means, this algorithm is also a wrapper around K-means. It will generate a hierarchical tree of clusters. While the approach is intuitively meaningful, applying normality tests can become difficult when the set of data is extremely large (e.g. on the order of tens of thousands). The one dimensional projection of the data will be very high in dimension and tend to look Gaussian according to the Central Limit Theorem [6] and hence the need of splitting a cluster could not be detected even when it is not Gaussian. Powerful normality test like the Shapiro Wilk test [21] can handle a sample size of at most 5000. Also, the assumption of having Gaussian distribution in clusters is too strong in many real data, such as in Astronomy time series. It has been extensively tested within the LIGO community and it is known that LIGO data is not necessarily Gaussian in nature [2].

Similar to G-means, there are a number of algorithms that rely on statistical tests to check the "goodness of fit" of data according to some distribution. For example, PG-means projects both the data set and learned clusters to one dimension and then applies the Kolmogorov-Smirnov test (KS) to check the goodness of fit of the data to distribution implied by the clusters where model parameters are learned by Expectation Maximization (EM) [8]. Combining normality test and splitting for discovering K can be problematic due to application of possibly costly statistical tests and difficulty in applying the distribution test itself when data dimension is high.

Due to advances in multi-sensor networks, large amounts of astronomical data have been gathered in the form of sensor information. Discovering interesting patterns in these astronomical data has profound implication for making new discoveries in Astro-Physics, for instance, in opening up new understanding of the nature of the universe. Since sensor data can be gathered on a rate of terabytes per week, processing such a gigantic amount of data requires at least semi-automated data mining mechanisms. Hence, clustering large amounts of astronomical data has recently become an interesting problem in the time series data community [10, 18]. Astronomical data are usually plagued with noise, very high in dimension, and not necessarily Gaussian in distribution. Limitations in current approaches motivated us to present a similarity driven clustering algorithm that we call S-means. Instead of specifying the number of clusters K, a similarity threshold T is used as a quality constraint in the search of optimal solutions to the clustering problem.

The rest of the paper is organized as follows. In Section 2, we provide a background on K-means, then the S-means algorithm is described afterward. Its time complexity and convergence are also discussed. In Section 3, we describe the experimental domains and experimental evaluation are demonstrated in which S-means is compared to existing approaches like K-means and G-means. Finally, conclusion and future work are presented in Section 4.

2 From K-means to Similarity driven clustering

Before we describe our similarity driven approach to clustering, we need to first revisit the classic K-means algorithm.

2.1 K-means

Two clustering algorithms are most popularly used: hierarchical clustering and K-means. Hierarchical clustering produces a nested hierarchy of clusters according to a pairwise distance matrix of all the given points. The hierarchy gives intuitive visualization. A user does not need to have expertise in Computer Science since no parameter excepts distance measure is needed in hierarchical clustering. However, the distance matrix limits its application to small data sets (both time complexity and space complexity are $O(n^2)$ or higher).

K-means basically divides a given data set into K clusters via an iterative refining procedure. The procedure simply consists of three steps:

1. initialize K centroids (c_i, $1 \leq i \leq K$) in the vector space.
2. Calculate the distances from every point to every centroid. Assign each point to group i, if c_i is its closest centroid.
3. Update centroids. Each centroid is updated as the mean of all the points in its group.
4. If no point changed its membership or no centroid moved, exit, otherwise, go to step 2.

The iterative procedure uses hill climbing to minimize the objective function:

$$J = \sum_{i}^{K} \sum_{j}^{N} \|x_j^{(i)} - c_i\|^2 \tag{1}$$

where $\|x_j^{(i)} - c_i\|^2$ denotes Euclidean distance between point x_j to corresponding centroid c_i. The Euclidean distance can be substituted by any distance measure.

Although the procedure will always terminate, K-means might converge to a local minima. K-means is a simple algorithm that has been employed in many problem domains. However, one of the major problems of K-means is that we do not know the right number of clusters in advance. There is no existing theoretical solution to find the optimal number of clusters for any given data set. A common approach is to score the results of multiple runs with different K values according to a given criterion. The criterion might incur new risk and parameter setting problems. We propose to use a similarity driven approach to clustering that does not require specification of K.

2.2 S-means: Similarity Driven Clustering

The clustering problem we need to solve is: *given N data points, group them into clusters such that within each cluster, all members have similarity $\geq T$ with the centroid where T is a user-defined threshold*. Similarity is a central notion in classification problem. The definition of cluster also implies that the cluster members should have high similarity with each other. The most popular Euclidean distance is a dissimilarity measure, which can be converted to a similarity measure in Gaussian form: $k(x_i, y_j) = exp(-\gamma \|x_i - y_j\|^2)$. This is also called the Radial Basis Function (RBF kernel) in kernel machines. Kernel methods all use similarity measures instead of dissimilarity. Similarity value is usually normalized to between 0 and 1; a confidence threshold in [0, 1] also makes intuitive sense to users. There are a large number of similarity measures available beside the RBF, such as correlation r, R-squared (the square of r). Indeed, any kernel function can be considered a similarity measure. Therefore, the clustering problem, if defined in terms of similarity, is more user-friendly and will likely gain more popularity due to the increasing amount of interests in kernel methods.

S-means starts from $K = 1$ by default and a user can specify any starting K. Note that the starting K is only an optional parameter in S-means. First, same as in K-means, we initialize K centroids. Second, calculate the similarities from every point to every centroid. Then, for any point, if the highest similarity to centroid c_i is $\geq T$, group it to cluster i, otherwise, add it to a new cluster (the $K + 1^{\text{th}}$ cluster). Third, update each centroid, using the mean of all member points by default. If one group becomes empty, remove its centroid and reduce K by 1. Repeat the second and third step until no new cluster is formed and none of the centroids moves.

Note that S-means is similar to K-means but with some differences. The major difference lies in the second step, which basically groups all the points to a *new* cluster whose highest similarity to *existing* centroids is below the given threshold . In K-means, all points must go to one of the existing K groups, which is unfair for some points when their similarities to corresponding closest centroid are very low. This simple difference makes big impact on the output of clusters. Also, we can let K starts from 1 and it will converge to a value, which eliminates the need of specifying a fixed K value. Also, there is a minor difference in the third step. While K is incremented by 1 if a new cluster is formed, it is decremented when some groups become empty. It is not unusual that as K keeps increasing, some old groups would disappear (as points in existing clusters could change membership as new clusters are formed). This way, K will not go beyond control.

The following is the pseudo code of S-means in Matlab style. The running code in Matlab is downloadable from our website[1].

S-means

Inputs: N data points $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N]$ and similarity threshold T.
Outputs: number of clusters K , centroids $\mathbf{C} = [\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_K]$ and labels Y.

```
 1: $K = 1$; /* starting number can be specified*/
 2: Randomly choose $K$ centroids $C$;
 3: change=1;
 4: while change==1 do
 5:   $NewCluster = []$; /*initialize to empty */
 6:   for $i = 1$ to $N$ do
 7:     for $j = 1$ to $K$ do
 8:       $SimToClusters(i, j) = Similarity(x_i, c_j)$;
 9:     end for
10:   end for/*End of similarity calculation */
11:   for $i = 1$ to $N$ do
12:     $maxSim = max(SimToClusters(i, :))$;
13:     if $maxSim \geq T$ then
14:       $Y(i) = find(SimToClusters(i, :) == maxSim)$; /*Assign label */
15:     else
16:       $NewCluster = [NewCluster, x_i]$; /* Add to a new cluster */
17:       $Y(i) = K + 1$; /*Assign label $(K + 1)$*/
18:     end if
19:   end for/*End of label assignment*/
20:   for $i = 1$ to $K$ do
21:     $c_i = mean(find(X(:, find(Y == i))))$;
22:     if $c_i$ is empty then
23:       remove $c_i$; /* $K$ will also reduce by 1 */
24:     end if
25:   end for/*End of centroid update*/
26:   if NewCluster is empty and no centroid changed then
27:     change==0;
28:   end if;
29: end while. /*end of algorithm*/
```

For the sake of simplifying description, two for loops are used to calculate the similarities in line 6-10. The loops are usually slow in Matlab. Using matrix dot product will be very efficient. Since many similarity measures can be implemented in dot product, the matrix product in Matlab can be utilized. The loop in line 11-19 varies from 1 to N. Matlab can also provide an efficient way to implement it using built-in functions like `find` and `max`. Interested readers should reference our Matlab real code.

The convergence of the S-means is guaranteed, because in the extreme case when K equals N every point has 100% similarity to itself. Of course, the extreme case is not desired. The result of K depends on threshold T. Intuitively, a high T produces more

[1] http://blue.utb.edu/hlei/Smeans/Smeans_V01.zip

clusters. When $T = 0$, S-means is reduced back to K-means. In this sense, S-means is a flexible generalization of K-means.

If S-means converges to K clusters, then time complexity is $O(N * (1 + 2 + \cdots + K)) \approx O(N * K^2/2)$. Recall that the time complexity of K-means is $O(NKL)$, where L is the number of iterations, strongly related to K and the distribution of data points. If using model selection based method to try different K and choose the best one, then the time complexity is approximately $O(N * K^2/2 * L)$, assuming K value varies from 1 to desired number of clusters. Besides avoiding the use of statistical tests (since both the number of data points and the data dimensionality could be high), S-means has advantages in low time complexity. In the following section, extensive experiments were performed to evaluate S-means from different perspectives.

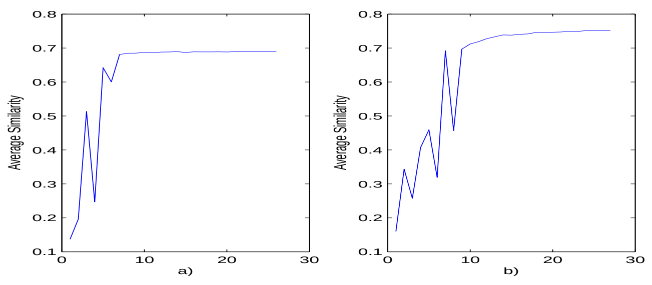


Fig. 1: S-means converges on dataset SCT in less than 30 iterations. a) the maximum number of clusters is restricted (up-bound is set 6). b) without restriction (up-bound is set 600).

3 Experiments

First, a small dataset was used to evaluate the convergence and execution speed of S-means. Second, a medium size dataset with ground-truth class labels was used to evaluate the accuracy of S-means in comparison with K-means and G-means. Third, we applied S-means to mine compact clusters in a simulated large dataset of Gravitational-wave time series. All the following experiments were performed in Matlab on a SUN Ultra 40 microsystem that has 2.8Ghz CPU and 3.0G RAM memory.

3.1 Convergence and Execution speed

We adopted the popular toy dataset Synthetic-Control time series dataset (SCT) to test the convergence and robustness of S-means [4]. SCT contains 600 samples of synthetically generated control charts. The length of each sample is 60. The similarity measure used is R-squared (squared Pearson's r) which is essentially equivalent to Euclidean distance after mean-variance normalization [17].

Fig. 1 show the convergence of S-means with $T = 0.7$ with/withoud restriction on the maximum number of clusters. Without restriction, S-means need more iterations to finish and more clusters are generated. The average similarity of all points to their corresponding centroids is an equivalent measure of the objective function described in equation (1). Like K-means, S-means is also a hill-climbing algorithm. Although global optimum might not be reached, convergence is guaranteed.

We varied T from 0.05 to 0.75 by step 0.05 to watch the changing of the number of clusters on the SCT dataset. The result is illustrated in fig. 2, from which we can see that the number of returned clusters are sensitive to the threshold setting. Increasing similarity threshold T significantly increases the number of clusters, because it imposes the requirement that all cluster are compact (minimum similarity to the centroid is no less than T). Depending on the similarity adopted and the mining target, intuitive T should be properly set. Big T tends to lead to overfitting, which can be considered as one weakness of S-means. But from perspective of outlier detection, it is a good phenomenon that some outliers are grouped as single-item stand-alone clusters. That is, the clusters with only one elements are outliers which might interest user.

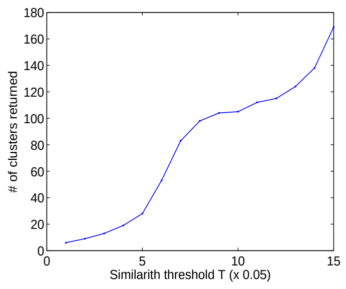


Fig. 2: The number of returned clusters with respect to the similarity threshold T setting.

S-means was compared in execution time with standard K-means and fast K-means with triangle inequality acceleration [11]. The source codes for standard K-means and fast K-means in Matlab are provided by the authors' of fast K-means. For K-means, we let K vary from 1 to 20 on the SCT dataset. For S-means, we let the up-bound number of clusters varied from 1 to 20. In this way, the comparison in execution time is fair. The results are plotted in Fig. 3. S-means has the same level of speed with fast the K-means when K beyond 5. S-means has a peak execution time when K=4. The cause is that S-means has to force itself to converge when the clusters reaches up-bound. S-means is fast because: i) in every iteration only one new cluster is added and ii) as the total number of clusters increase, a large portion of old cluster centroids do not move (already converged), thus, there is no necessity to recalculate distances from all the points to those converged centroids.

3.2 Clustering Accuracy

One of the major concerns for new algorithm is whether it is accurate. Classification accuracy usually depends on both (dis)similarity measure and classification strategy. To

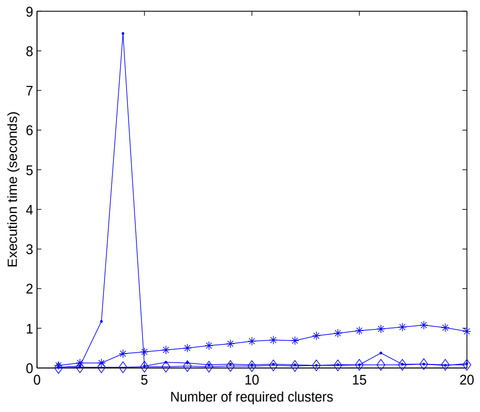


Fig. 3: Execution time comparison between S-means (dot point), standard K-means (star point and fast K-means (diamond point).

compare S-means and K-means fairly in accuracy, we still used the R-squared measure in S-means and equivalent Euclidean distance in K-means. The benchmark dataset used was Pendigit [5], which has been widely used in evaluating classification algorithms. It has 10 classes (digit 0 to 9). Total training samples is 7494 and test samples 3498. Each sample has 16 attributes (x-, y- coordinates). We concatenated x-, y- coordinates and made each sample a 32-length vector. All the vectors were normalized by mean-variance before input for clustering.

First, accuracy was compared in clustering the training dataset by K-means, G-means and S-means respectively. Suppose we don't know how many classes in Pendigit. For K-means, the necessary step is to guess K=1 up through to some up-bound P (P was set 20 in our experiments). In G-means, a confidence threshold is needed in place of K. The default confidence is set 0.001 in the original G-means package [13]. We let confidence vary from 0.0001 to 0.002 by step 0.0005. We recorded the accuracy of clustering results against the ground-truth labels in each step. For S-means, the necessary step is to vary T if no prior knowledge about the number of clusters. T was varied from 0 to 0.95 by step 0.05 and the up-bound number of clusters P was set 20, same as K-means.

Calculating clustering accuracy is a tricky task. Suppose K cluster are returned and the cluster assignment is $L = [l_1, l_2, \cdots, l_N]$, $1 \le l_i \le K$. And suppose the ground-truth number of clusters is K_t and the true labels are $Lt = [lt_1, lt_2, \cdots, lt_N]$, $1 \le lt_i \le K_t$. Note that K does not necessarily equal K_t. We used the following pseudo code to calculate accuracy:

```
1: **for** $i = 1$ to $K$ **do**
2:     **for** $j = 1$ to $K_t$ **do**
3:         $Common(i, j) =$
           $NumberOfCommonMemebers(find(L == i), find(Lt == j));$
```

```
4:     end for
5: end for
6: count=0;
7: for $i = 1$ to $K_t$ do
8:     $count = count + max(Common(:, i));$
9: end for
10: $Accuracy = count/N;$
```

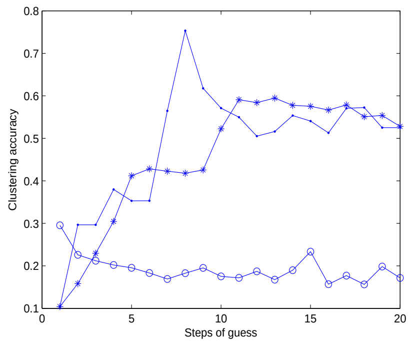


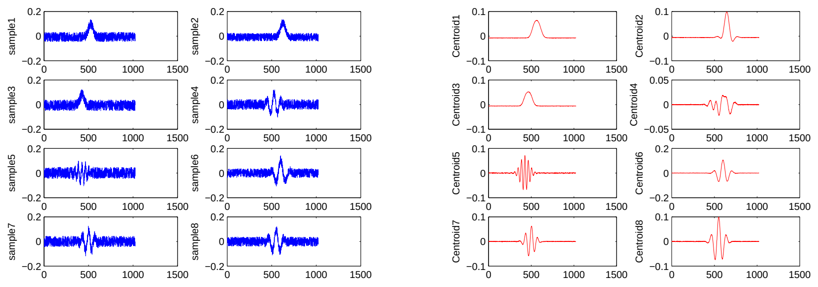
Fig. 4: Accuracy comparison between S-means (dot point), fast K-means (star point) and G-means (circle point). K varies from 1 to 20 in K-means. Confidence varies from 0.0001 to 0.002 in G-means. T varies from 0 to 0.95 with step 0.05 and up-bound of clusters is set 20 in S-means.

Line 3 computes the maximum common number of members between two clusters. The nested *for* loops (line 1-5) find the best matching between two sets of clusters. Line 7-9 sums up all the number of common members (which are correctly assigned). Fig. 4 shows the clustering accuracy with S-means and K-means on each step of guess. In step 9, S-means reaches its peak accuracy (when T=0.40). K-means reaches its peak accuracy when K=10. S-means's peak accuracy is significantly high than K-means's peak accuracy. After step 10, both algorithms decreases in accuracy, which is because the number of returned clusters run farther away from the true number of classes. G-means shows poor performance due to the weak assumption of Gaussian distribution. According to our experience, the real data's statistical distribution is usually not as simple as Gaussian.

3.3 Mining Gravitational-wave Astronomy time series

The detection of Gravitational waves is the next frontier in astronomy. Several large scale detectors have been constructed around the world, such as the Laser Interferometric Gravitaitonal wave Observatory (LIGO) in the U.S. [1]. These detectors are part of a world wide network that is collecting data at the rate of several Tb per week. Mining gravitational wave data for useful information is a daunting task and one of the major challenges in the area of astronomical data analysis. The following simulation is an example of a typical clustering task that arises in such an analysis and also demonstrates the application of the S-means.

A dataset consisting of 20020 time series with length 1024 were artificially generated. Each sequence is first generated by a single Gaussian modulated sinusoid signal. The amplitude is scaled such that the matched filtering signal to noise ratio (SNR) is 1 in white Gaussian noise with zero mean and unit variance. Then, a single Gaussian pulse is added to the signal in random position. The pulse amplitude is also scaled with SNR=1 in white Gaussian noise. Fig. 5a shows the typical shape of the each cluster. The simulated time series is a close representation of the actual triggers in Gravitational-wave Astronomy time series.



(a) (b)

Fig. 5: (a) Samples of simulated Gravitational-wave time series. (b)Centroids mined by S-means when $T = 0.1$.

Although Gaussian process is used here, the possible clusters inside the dataset does not follow Gaussian distribution. As discussed in the introduction, GW events dot not follow any known statistical distribution. As a part of GW data analysis, clustering time series based on shapes (which can be matched by similarity measures) is a reliable method.

Same as experiments above, the simple similarity measure R-squared was used in S-means. Since the number of clusters in the dataset is unknown, our goal is to mine how many compact clusters exist. With high similarity threshold T, it is expectable that many time series will be stand-alone clusters. Therefore, we started with low T. Initially, we set T=0.1. We found that S-means converges to 8 clusters in less than 50 iterations. The centroids are as shown in Fig. 5b. The convergence was completed in 34.1 seconds. The average similarity of each item to its corresponding centroid was also converged to about 0.55, as shown in Fig. 6. Note that the average similarity times the number of items is equivalent to the objective function in equation (1). This means, S-means converges via a hill-climbing approach to minimize the object function and at the same time discover the number of clusters according to the similarity requirement.

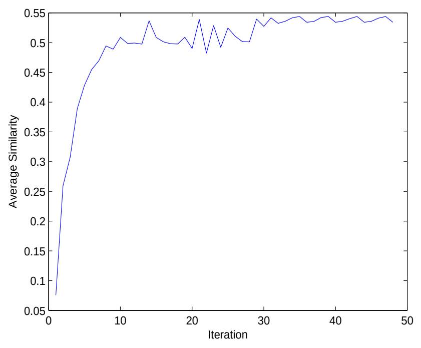


Fig. 6: S-means converges to 8 clusters in 50 interations when $T = 0.1$.

Then, we varied T from 0.2 to 0.6 by step 0.1 and recorded the number of clusters, iteration, execution time and average similarity on each step. The results are shown in Table 1 The number of clusters increases dramatically when $T = 0.4$ to unbound 1000. Correspondingly, and execution time and number of iterations changes sharply at step 4, while the average similarity increases steadily. So, we can see it is not worth to set $T \geq 0.4$ in mining this dataset. We can conclude that the simulated dataset has 8 to 16 compact clusters, depending on parameter T.

Table 1: Number of clusters, iterations, execution time and average similarity change as similarity threshold T increases.

Threshold T	0.1	0.2	0.3	0.4	0.5	0.6
Number of clusters	8	11	15	1000+	1000+	1000+
Number of iterations	48	29	20	1008	1003	1001
Execution time (secs)	34.1	23.1	23.3	1673.8	1221.8	1064.5
Average similarity	0.563	0.575	0.612	0.643	0.663	0.675

4 Conclusions and future work

S-means eliminates the necessity of specifying K (the number of clusters) in K-means clustering. An intuitive argument, similarity threshold T is used instead of K in S-means. Experiments demonstrates the efficiency and effectiveness of S-means in comparison with standard K-means, fast K-means and G-means. S-means mines the number of compact clusters in a given dataset without prior knowledge in its statistical distribution. We applied S-means to simulated Gravitations-wave time series analysis and discovered the existence of compact clusters.

While we believe S-means is promising in its simplicity, efficiency and effectiveness, we are aware that more extensive comparative experiments are needed to further validate the algorithm with other clustering algorithms. For instance, the clustering result is very sensitive to threshold T and the number of returned clusters can be unexpectedly large when T is high (e.g, $T > 0.4$). Also, it is necessary to evaluate S-means with different similarity measures such as Dynamic Time Warping and kernel functions in our future work.

References

1. B. Abbott and et al. (LIGO Scientific Collaboration). Search for gravitational waves from binary black hole inspirals in ligo data. *Physics Review*, 73, 062001, 2006.
2. A. Abramovici and et al. LIGO: The laser interferometer gravitational wave observatory. *Science*, 256:325–333, 1992.
3. H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.
4. R. J. Alcock and Y. Manolopoulos. Time-series similarity queries employing a feature-based approach. In *Proceedings of the 7th Hellenic Conference on Informatics*, 1999.
5. A. Asuncion and D. Newman. UCI machine learning repository, http://www.ics.uci.edu/∼mlearn/mlrepository.html, 2007.
6. G. Casella and R. Berger. *Statistical Inference*. Duxbury Press, 2001.
7. R. D'andrade. U-Statistic hierarchical clustering, 1978.
8. A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(1, pages =).
9. C. Ding, X. He, H. Zha, and H. Simon. Adaptive dimension reduction for clustering high dimensional data. In *Proceedings of the 2nd IEEE International Conference on Data Mining*, pages 107–114, 2002.
10. S. G. Djorgovski, C. Donalek, A. Mahabal, R. Williams, A. Drake, M. Graham, and E. Glikman. Some pattern recognition challenges in data-intensive astronomy. In *the 18th International Conference on Pattern Recognition (ICPR 2006)*, page 856, 2006.
11. C. Elkan. Using the triangle inequality to accelerate kmeans. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 147–153, 2003.
12. U. M. Fayyad, C. Reina, and P. S. Bradley. Initialization of iterative refinement clustering algorithms. In *Proceedings of the 4th International Conference on Knowledge Discovery and Data Mining*, pages 194–198, 1998.
13. G. Hamerly and C. Elkan. Learning the k in k-means. In *Advances in Neural Information Processing Systems*, volume 17, 2003.
14. S. Haykin. *Neural Networks: A Comprehensive Foundation (2nd Edition)*. Prentice Hall, 1998.
15. A. Jain, M. Murty, and P. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3), September 1999.
16. K. Kalpakis, D. Gada, and V. Puttagunta. Distance measures for effective clustering of arima time-series. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 273–280, 2001.
17. H. Lei, S. Palla, and V. Govindaraju. ER2: An intuitive similarity measure for on-line signature verification. In *the 9th International Workshop on Frontiers in Handwriting Recognition*, pages 191–195, 2004.
18. S. Mukherjee. Multidimensional classification from kleine welle triggers from ligo science run. *Classical Quantum Gravity*, 23(S661-71), 2006.
19. D. Pelleg and A. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, pages 727–734, 2000.
20. J. Rissanen. Modeling by the shortest data description. *Automatica*, 14:465–471, 1978.
21. S. Shapiro and M. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3 and 4):591–611, 1965.
22. W. Sheng and X. Liu. A hybrid algorithm for k-medoid clustering of large data sets. In *IEEE Congress On Evolutionary Computation*, volume 1, pages 77–82, 2004.