A Prototypes-Embedded Genetic *K*-means Algorithm

Shih-Sian Cheng[1,2], Yi-Hsiang Chao[1,2], Hsin-Min Wang[2], and Hsin-Chia Fu[1]
[1] Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
[2] Institute of Information Science, Academia Sinica, Taipei, Taiwan
E-mail: {sscheng, yschao, whm}@iis.sinica.edu.tw, hcfu@csie.nctu.edu.tw

Abstract

This paper presents a genetic algorithm (GA) for K-means clustering. Instead of the widely applied string-of-group-numbers encoding, we encode the prototypes of the clusters into the chromosomes. The crossover operator is designed to exchange prototypes between two chromosomes. The one-step K-means algorithm is used as the mutation operator. Hence, the proposed GA is called the prototypes-embedded genetic K-means algorithm (PGKA). With the inherent evolution process of evolutionary algorithms, PGKA has superior performance than the classical K-means algorithm, while comparing to other GA-based approaches, PGKA is more efficient and suitable for large scale data sets.

1. Introduction

In many applications, data clustering techniques have been applied to discover and extract the hidden structure in a data set and, thus, the structural relationships between individual data points can be detected. Data clustering is also known as unsupervised learning [1]. A variety of competitive learning (CL) schemes has been developed, distinguishing in their approaches to competition and learning rules. The simplest and most prototypical CL algorithms are mainly based on the winner-take-all (WTA) [2] (or called hard CL) paradigm, where adaptation is restricted to the single prototype that best matches the input patterns. Different algorithms in this paradigm such as LBG (generalized Lloyd) [3] and *K*-means [4] have been well recognized. In the statistical pattern recognition community, *K*-means clustering has been widely applied. For examples, in [5] and [6], the resulting prototypes of the *K*-means algorithm were used as the initial mean vectors for the training of Gaussian mixture models, which were the essential part of their probabilistic decision-based classifiers.

The objective of *K*-means clustering is to partition the input data set into *K* groups/clusters such that the total within cluster variation (TWCV) is minimized. However, the objective function, TWCV, can not be optimized analytically. Both LBG and *K*-mans algorithm only guarantee to converge to a local optimum. To tackle this problem, genetic algorithms (GA's), which have been successfully applied to many function optimization problems, have been proposed to minimize TWCV [7-12]. These GA's can be distinguished by the designs of the GA operators, such as encoding, crossover, and mutation. Two encoding schemes have been proposed. The first, the so-called string-of-group-numbers [7-11], encodes the cluster index of each data sample into a chromosome, while the second encodes the parameters of prototypes of the clusters into a chromosome [12]. For effectively solving a specific optimization problem, it is often desirable to hybridize GA's with problem-specific information/techniques. A good example is the genetic *K*-means algorithm (GKA) [7] and the fast GKA (FGKA) [8] later, which applied the *K*-means operator (KMO), i.e., one step of the *K*-means algorithm, after the mutation operator to speed up the convergence process. Both GKA and FGKA do not contain a crossover operator, and the production of partition is carried out by the selection, mutation, and KMO operators only.

In this paper, unlike GKA and FGKA, the proposed GA implements both crossover and mutation as a classical GA. For encoding, we randomly select data samples from the data set as the parameters of prototypes of the clusters, and then encode them into chromosomes. The crossover operator is to exchange the prototypes within two chromosomes. We adopt KMO as the mutation operator, which plays a very important role in the proposed GA since it guarantees to reduce the TWCV value for each chromosome. The proposed GA is therefore termed as the prototypes-embedded genetic *K*-means algorithm (PGKA). Comparing to GKA and FGKA, PGKA performs equally well in terms of clustering performance, but is more efficient and suitable for large scale data sets.

The rest of this paper is organized as follows. First, the general concept of genetic algorithms is reviewed in Section 2. Then, the proposed genetic algorithm for K-means clustering is introduced in Section 3. Finally, the experimental results conducted on UCI Image Segmentation Database [13] are presented in Section 4, and conclusions are made in Section 5.

2. The genetic algorithms

Genetic algorithms (GA's) belong to a particular class of evolutionary algorithms (EA's) which draws inspiration from the process of natural evolution [14]. The operators involved in the evolution process include encoding, parent selection, recombination, mutation, and survivor selection. Fig. 1 shows a diagram of the general scheme of a GA. GA's maintain a population of candidate solutions and perform parallel search in the search space via the evolution of these candidate solutions.

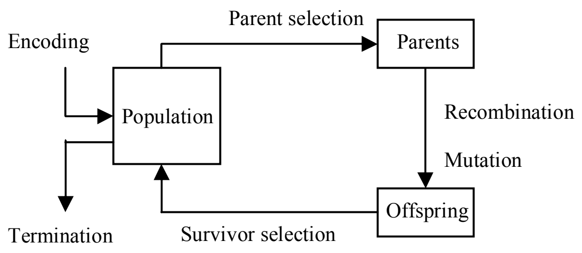


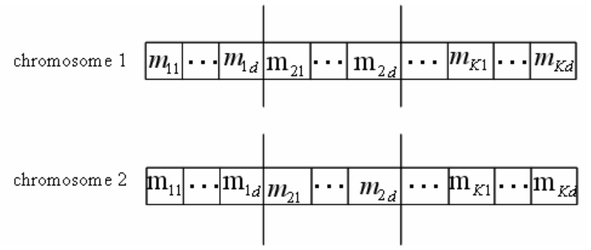
Figure 1. The general scheme of a genetic algorithm as a flow chart.

3. The proposed prototypes-embedded genetic K-means algorithm

The objective of K-means clustering is to partition a data set $X=\{\mathbf{x}_i \mid \mathbf{x}_i \in R^d, i=1,2,...,n\}$ into K clusters so as to minimize the so-called total within-cluster variation (TWCV),

$$f(M)=\sum_{j=1}^{K}\sum_{\mathbf{x}\in C_j}\|\mathbf{x}-\mathbf{m}_j\|^2,$$

where $M=\{\mathbf{m}_1,\mathbf{m}_2,...,\mathbf{m}_K\}\subset R^d$ is the parameter set of $f(M)$ and $\mathbf{m}_j=[m_{j1},m_{j2},...,m_{jd}]$ is the prototype of cluster C_j. In the following we describe the design of the proposed PGKA for K-means clustering thoroughly.



$$\begin{array}{l|c|c|c|c|c|c|c|c|c|c|c|}
\text{chromosome 1} & m_{11} & \cdots & m_{1d} & m_{21} & \cdots & m_{2d} & \cdots & m_{K1} & \cdots & m_{Kd} \\
\text{chromosome 2} & m_{11} & \cdots & m_{1d} & m_{21} & \cdots & m_{2d} & \cdots & m_{K1} & \cdots & m_{Kd}
\end{array}$$

Figure 2. *N*-point crossover of two chromosomes. Each crossover point is set on the boundary between two prototypes. In this example, the value of *N* is 2.

1) Encoding: In PGKA, each chromosome is a string of length $K\times d$ which is the concatenation of the prototypes, i.e., $\{\mathbf{m}_1,\mathbf{m}_2,...,\mathbf{m}_K\}$. For each chromosome, we randomly select K samples from X as the initial prototypes.

2) Parent selection: Five chromosomes are randomly selected from the population, and then the one with the smallest TWCV value is selected as a parent. The above procedure is repeated iteratively until the predefined number (the same as the population size in this study) of parents is reached. This is known as the deterministic tournament selection [14].

3) Recombination: The N-point crossover is used here. Each crossover point is set on the boundary between two prototypes in the chromosome. Fig. 2 shows an example of two-point crossover. In other words, the crossover here means that two chromosomes exchange their prototypes with each other with a predefined crossover probability. When $N \geqq 2$, the crossover probability must be less than 1 to avoid the situation that the offspring are just duplicates of their parents.

4) Mutation: In most cases, the mutation operator is to make a random change to the alleles of the chromosomes. For example, while mutating a gene of a chromosome, we can just randomly draw a number from a uniform distribution, and then add this number to the allele of this gene. Here, we use the one-step K-means operator (KMO) [7, 8] for mutation since it guarantees to improve the clustering. KMO performs one step of the K-means algorithm using the prototypes in the chromosome as the initial prototypes, and then the resulting prototypes are concatenated to produce the new chromosome. In GKA, both mutation and KMO operators might yield chromosomes with some empty clusters containing no data samples. In [7], these chromosomes were termed as illegal strings and were eliminated by several time-consuming schemes. FGKA [8] improved GKA by avoiding this overhead. In the proposed PGKA, the illegal prototypes corresponding to empty clusters will not be updated by the one-step K-means clustering in the KMO mutation. Since the chromosomes with illegal prototypes usually have larger TWCV values, they will be knocked out during the evolution process of the GA.

5) Survivor selection: We adopt the generational model, in which the whole population is replaced with its offspring [14].

In GKA and FGKA, the major time-consuming operations (operators) for a given solution string are fitness (TWCV) evaluation, mutation, and KMO. The time complexities are $O(nd)$, $O(n^2d)$ and $O(nKd)$, respectively [7, 8], where K is the number of clusters. n and d are the number and dimension of data samples, respectively. PGKA requires the same computational load in fitness evaluation, and use KMO as the mutation operator, and the time cost of the applied crossover operator is $O(Kd)$. Cleary, PGKA has much less time cost at each evolution generation, especially when the number of data samples is large. In our experiments, the number of generations needed for convergence in PGKA is very close to that in FGKA (refer to Sec. 4). As a whole, PGKA is more efficient and suitable for large scale data sets than GKA and FGKA.

4. Experimental results

The effectiveness of the proposed PGKA is evaluated on the SKY testing data in UCI Image Segmentation Database [13]. This data set contains three hundred 19-dimensional real-valued vectors. The number of prototypes/clusters for K-means clustering is set as 20 in all the following experiments. Because GA's are randomly initialized in this study, the performance reported in this paper is the average TWCV value of ten independent runs.

4.1. One-point versus (*K*-1)-point crossover

First of all, we would like to evaluate the N-point crossover operator in PGKA. Two extreme cases are compared, namely, N=1 and N=K-1. Fig. 3 shows the performance of PGKA with and without KMO (mutation). The population size was set as 40. It is obvious that the (K-1)-point crossover outperforms the one-point crossover, in particular when KMO is not applied. In the case with one-point crossover and pc=1,

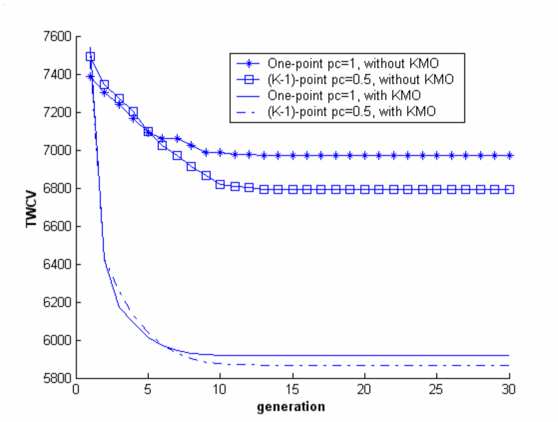


Figure 3. The performance (evolution) of PGKA with and without KMO using *N*-point crossover. "pc" denotes the crossover probability.

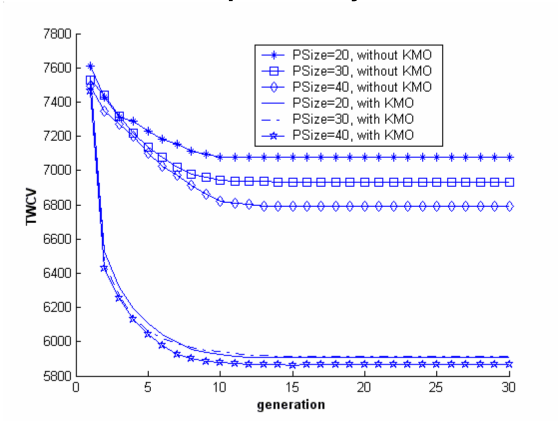


Figure 4. The performance (evolution) of PGKA with and without KMO in different population sizes. "PSize" denotes the population size.

on average, the two parents exchange K/2 prototypes (the prototypes at the right side of the crossover point) with each other. In the case with (K-1)-point crossover and pc=0.5, the two parents also exchange K/2 prototypes on average. But, different from the one-point crossover, the exchange for each prototype is considered independently. This may be the reason why the (K-1)-point crossover has better performance than the one-point crossover, especially when KMO is not involved. When KMO is used as the mutation operator in PGKA, the merit of the flexibility in prototype exchange seems to become minor.

4.2. The population size

Secondly, we investigate the effect of population size. The (K-1)-point crossover with pc=0.5 is applied in PGKA. Fig. 4 shows the performance of PGKA with and without KMO in different population sizes. From Fig. 4, we can observe that, when KMO is not

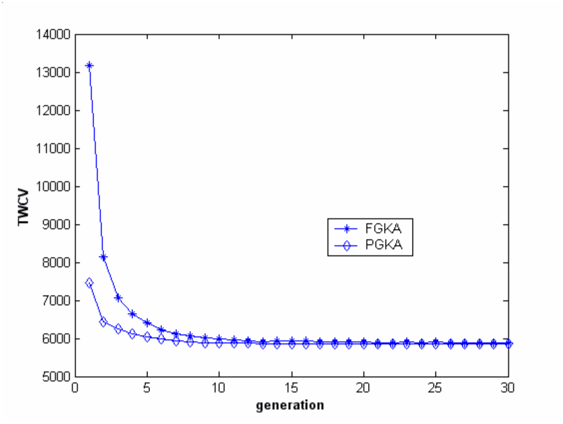


Figure 5. The performance (evolution) of FGKA and PGKA. The population size is 40.

applied in PGKA, the performance is very sensitive to the population size. It is because, in this case, PGKA just searches solutions from the data samples. By contrast, with KMO, the performance is not sensitive to the population size. The experimental results again demonstrates that KMO dominants the performance.

4.3. Comparing with FGKA and *K*-means algorithm

Thirdly, we compare PGKA with FGKA. Table 1 summarizes the performance of FGKA and PGKA in different population sizes. It is found that there is no significant difference between these two GA-based approaches in the clustering performance. Fig. 5 shows the evolution over generations of FGKA and PGKA. The population size is 40. From Fig. 5, we can observe that PGKA converges slightly faster than FGKA. In addition, as discussed in Sec. 3, PGKA requires a lower computational load in each generation than FGKA. Thus, we can conclude that PGKA is more efficient than FGKA. Finally, we compare PGKA with the classical *K*-means algorithm. We run the *K*-means algorithm forty times with random initialization, and the minimum, average, and maximum TWCV values are 5985.41, 6190.23 and 6473.86, respectively. Comparing to the results in Table 1, it is obvious that both FGKA and PGKA outperform the classical *K*-means algorithm.

5. Conclusion

In this paper, a genetic algorithm for *K*-means clustering is proposed. The proposed PGKA algorithm can be characterized by the design of its operators, including encoding, crossover, and mutation. Comparing to other GA-based approaches, PGKA is more efficient and suitable for large data sets. The experimental results evaluated on an open data set indicate that PGKA performs equally well as the existing GA-based approach, FGKA, and outperforms the classical *K*-means algorithm.

Table 1. Performance, in TWCV values, of the FGKA and PGKA in different population sizes.

Approach	Population size		
	20	30	40
FGKA	5923.52	5908.25	5873.10
PGKA	5905.43	5914.66	5864.77

6. References

[1] J.-M. Jolion, P. Meer, and S. Bataouche, "Robust clustering with applications in computer vision," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 13(8), pp. 791–802, 1991.

[2] J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, New York, USA, 1991.

[3] Y. Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design," *IEEE Trans. Communications*, 28(1), pp. 84–95, 1980.

[4] J. Mac Queen, "Some methods for classification and analysis of multivariate observations," *Proc. 5th Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297, 1967.

[5] S.-H. Lin, S. Y. Kung, and L. J. Lin, "Face recognition/detection by probabilistic decision-based neural networks," *IEEE Trans. Neural Networks*, 8(1), pp. 114-132, 1997.

[6] H. C. Fu, H. Y. Chang, Y. Y. Xu, and H. T. Pao, "User adaptive handwriting recognition by self-growing probabilistic decision-based neural networks," *IEEE Trans. Neural Networks*, 11(6), pp. 1373-1384, 2000.

[7] K. Krishna and M. Narasimha Murty, "Genetic *K*-means algorithm," *IEEE Trans. Systems, Man, and Cybernetics,* 29(3), June 1999.

[8] Y. Lu, S. Lu, F. Fotouhi, Y. Deng, and S. J. Brown, "FGKA: A fast genetic *K*-means clustering algorithm," *Proc. ACM Symposium on Applied Computing*, 2004.

[9] R. Krovi, "Genetic algorithms for clustering: a preliminary investigation," *Proc. of the 25th Hawaii International Conference on System Sciences*, 1992.

[10] J. C. Bezdek, S. Boggavarapu, L.O. Hall, and A. Bensaid, "Genetic algorithm guided clustering," *Proc IEEE Conference on Evolutionary Computation*, 1994.

[11] M. Painho, "Using genetic algorithms in clustering problems," *Proc. International Conference on GeoComputation*, 2000.

[12] J. C. Bezdek and R. J. Hathaway, "Optimization of fuzzy clustering criteria using genetic algorithms," *Proc. IEEE Conference on Computational Intelligence,* 1994.

[13] UCI Machine Learning Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html.

[14] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing*, Springer, Berlin, 2003.