International Journal of Computer Applications (0975 – 8887)
Volume 7– No.7, October 2010

Entropy Weighting Genetic k-Means Algorithm for Subspace Clustering

Anil Kumar Tiwari
Disha College of Information Technology
Raipur, CG-INDIA

Lokesh Kumar Sharma
Rungta College of Engineering and Technology
Bhilai, CG - INDIA

G. Rama Krishna
K. L. University
Vijayawada, AP - INDIA

ABSTRACT

This paper presents a genetic k-means algorithm for clustering high dimensional objects in subspaces. High dimensional data faces data sparsity problem. In this algorithm, we present the genetic k-means clustering process to calculate a weight for each dimension in each cluster and use the weight values to identify the subsets of important dimensions that categorize different clusters. This is achieved by including the weight entropy in the objective function that is minimized in the k-means clustering process. Further, the use of genetic algorithm ensure for converge to the global optimum. The experiments on UCI data has reported that this algorithm can generate better clustering results than other subspace clustering algorithms.

General Terms

Data Mining

Keywords

Genetic Algorithm, Clustering, Subspace clustering.

1. INTRODUCTION

The ever-growing repository of data in almost all fields contribute significantly towards future decision making provided appropriate knowledge discovery mechanisms are applied for extracting, hidden but potentially useful information embedded in the data [4][9]. A knowledge-discovery system employs a wide class of machine learning algorithm to explore the relationships among tuples and characterize the nature of relationships that exist between them. Classification and clustering are two most commonly encountered knowledge discovery techniques that are applied to extract knowledge. Classificatory analysis refers to a set of supervised learning algorithms, which study pre -classified data sets in order to extract rules for classification. Clustering on the other hard refers to unsupervised learning algorithms which aim is to partition a given set of data elements into homogeneous groups called clusters. Clustering is one of the principal techniques applied for mining data arising from many fields some of which are banking or medical informatics. Information retrieval in bio-informatics lack of any prior knowledge about the distribution of the data points makes the problem more complex.

One of the initial and most frequently cited k-means algorithm states the process reasonably efficiently in the sense of class variance, corroborated to some extent by Mathematical Analysis and practical experience [4]. Also the k-means procedure is easily programmed and is computationally economical, so that it is feasible to process very large samples on a digital computer.

SYNCLUS algorithm is used for variable weighting in k-means clustering and extension of weighting in k-means clustering is developed to find optimal variable weights for ultra metric and additive tree fitting [9]. However these two algorithms are time consuming [9]. Modha, et al [1] explored the new approach by minimizing the ratio of the average within-cluster distortion over the average between-cluster distortion to achieve minimizing the optimal variables weights. Friedman and Meulman [3] proposed a new method by assigning the weight for each variable in each cluster to obtain the suitable weighting variables. Jing et al [5] recently proposed a novel approach by adding a new step to k-means. The new step iteratively updates variable weights based on the current partition of data by a weighting calculation formula. Unfortunately, all of the above algorithms do not consider the drawbacks of k-means:

(i) k-Means converge to a local optimum. In other words, all the algorithms in the above cannot achieve a global optimum.

(ii) The initial parameters influence the results of k-means.

In this paper, we combine genetic algorithm (GA) with k-means to select the best k value using entropy weighting. Since GA explores the space more thoroughly than k-means, the new algorithm (the genetic k-means algorithm) converges to a global optimum.

In the next section, we present related work on genetic cluster algorithm and entropy weighting cluster. Section 3 presents proposed algorithm. Section 4 contains experiment and result analysis. Finally we conclude our work on section 5.

2. RELATED WORK

Krishna and Murty [4] combined the features of k-means clustering algorithm and genetic algorithm and developed genetic k-means algorithm (GKA) GA finds a globally optimal partition of a given data into a specified number of clusters. GA's used earlier in clustering employ either an expressive cross over operator to generate valid child chromosomes from parent chromosomes or a costly fitness function or both. To circumvent these expressive operations, here k- Means operator is defined and used in GKA as a search operator instead of crossover. Biased mutation operator specific to clustering called distance based mutation is used in GKA. Using finite Markov Chain theory GKA converges to global optimum.

Fast genetic k-means algorithm (FGKA)[7] is inspired by genetic k-means algorithm and developed with some improved feature over GKA. FGKA is faster than GKA [7].

Lu et al[8] proposed Incremental Genetic k-means Algorithm (IGKA). The main idea of IGKA is to calculate objective value total within cluster variation (TWCV) and the cluster centered incrementally whenever the mutation probability is small. IGKA inherits the salient feature of FGKA of always covering to the global optimum.

Entropy weighting is used in k-means algorithm by or subspace clustering of high dimension sparse data by Jing et al in 2007[5]. Data sparsity problems are faced in clustering high dimensional data [5]. In this algorithm, k-Means clustering process is extended to calculate a weight for each dimension in each cluster and use the weight values to identify the function that is minimized in k-Means clustering process. An additional step is added to the k-means clustering process to automatically compute the weights of all dimensions in each cluster.

High dimensional data [5] is a phenomenon in real world data mining applications. Text data is a typical example. In text mining, a text document is viewed as a set of pairs $<t_i, f_i>$, where t_i is a term or word, and f_i is a measure of t_i, for example the frequency of ti in the document. The total number of unique terms in a text data set represents the number of dimensions, which is usually in thousands. High- dimensional data occurs in business as well. In retail companies, for effective supplier relationship management (SRM), suppliers are often categorized in groups according to their business behavior data is high dimensional because thousands of attributes are used to describe the supplier's behavior, including product items, ordered amounts, order frequencies, product quality, and so forth.

Sparsity is an accompanying phenomenon of high dimensional data. In text data, documents related to a particular topic, for instance sport, are categorized by one subset of terms. The term describing sport may not occur in the documents describing music. This implies that f_i is zero for the subset of terms in the documents describing music and vice versa. Such situation also occurs in supplier categorization. A group of suppliers are categorized by the subset of product items supplied by the suppliers. Other suppliers who did not supply product items have zero order amounts for them in the behavior data.

Clearly, clustering of high dimensional sparse data requires special treatment. This type of clustering methods is referred to as subspace clustering, aiming at finding clusters from subspace of data instead of the entire data space. In a subspace clustering, each cluster is a set of objects identified by a subset of dimensions and different subsets of dimensions and different clusters are represented in different subsets of dimensions.

3. ENTROPY WEIGHTING GENETIC k-MEANS ALGORITHM (EWGKM)

The main objective of the clustering algorithm under consideration is to partition a collection of n given patterns; each pattern is a vector of dimension d, into K groups such that this partition minimizes the TWCV, which is defined as follows.

$w_{ik} = 1$, if ith pattern belongs to kth cluster,

$w_{ik} = 0$, otherwise

Then, the matrix $w = [w_{ij}]$ has the properties that

$$w_{ij} \in \{0,1\} \text{ and } \sum_{j=1}^{K} \omega_{ij} = 1 \qquad (1)$$

In this section we use entropy weighting concept adopted from EWKM algorithms given by Jing et al 2007 [5] in Genetic k-means algorithm. Here we consider that the weight of a dimension in a cluster represents the probability of contribution of that dimension in forming the cluster. The entropy of the dimension weights represents the certainty of dimensions in the identification of a cluster. Therefore, we modify the objective function by adding the weight entropy term to it so that we can simultaneously minimize the within cluster dispersion and maximize the negative weight entropy to stimulate more dimensions to contribute to the identification of clusters. In this way we can avoid the problem of identifying cluster by few dimensions in spare data.

$$S(\omega) = \sum_{k=1}^{K}\left[\sum_{i=1}^{n}\sum_{j=1}^{d}\omega_{ik}\lambda_{ik}\left(x_{ij}-c_{kj}\right)^2 + \gamma\sum_{j=1}^{d}\lambda_{jk}\log l_{jk}\right] \quad (2)$$

Subject to

$$\sum_{k=1}^{K}\omega_{ik} = 1\; i \le i \le n,\; i \le k \le k, \qquad \omega_{ik}\,\epsilon\{0,1\}$$

$$\sum_{j=1}^{d}\lambda_{ik} = 1\; i \le k \le k, \qquad i \le j \le d, 0 \le \lambda_{jk} \le 1$$

Minimization of S in above objective function we use.

$$\lambda_{ik} = \frac{\exp\left(\frac{-D_{ik}}{\gamma}\right)}{\sum_{j=1}^{d}\exp\left(\frac{-D_{ik}}{\gamma}\right)}$$

Where $D_{ik} = \sum_{i=1}^{n}\omega_{ik}\left(x_{ij}-c_{kj}\right)^2$.

The positive parameter γ controls the strength of the incentive for clustering a more dimension for minimizing S we take ω and Λ fixed and C_{ij} is updated using.

$$C_{kj}\frac{\sum_{i=1}^{n}\omega_{ik}x_{ij}}{\sum_{i=1}^{n}\omega_{ik}}\;\; i \le j \le d\;\; i \le k \in K \qquad (3)$$

Above is independent of the parameter γ and the dimension weights λ_{ik}.

3.1 Coding

Here the search space of all W matrices that satisfy (1) A natural way of coding such W into a string sw, is to consider a chromosome of length n and allow each allele in the chromosome to take values from {1,2, …. K}. In this case, each allele corresponds to a pattern and its value represents the cluster number to which the corresponding pattern belongs. This is possible because (refer to eq. (1)) for all i, w_{ik} =1 for only one k. GKA maintains a population of such strings.

3.2 Initialization

The initial population P(0) is selected randomly. Each allele in the population can be initialized to a cluster number randomly selected from the uniform distribution over the set {1,…, K}. In this case, we may end up with illegal strings, strings representing a partition in which some clusters are empty, with some nonzero probability. This is avoided by assigning p, the greatest integer which is less than n/K, randomly chosen data points to each cluster and the rest of the points to randomly chosen clusters.

3.3 Selection

The selection operator randomly selects a chromosome from the previous population according to the distribution given by

$$P(s_i) = \frac{F(S_i)}{\sum_{j=1}^{N}F(S_j)}$$

Where $F(s_i)$ represents fitness value of the string s_i in the population and is defined in the next paragraph. We use the roulette wheel strategy for this random selection.

Solutions in the current population are evaluated based on their merit to survive in the next population. This requires that each solution in a population be associated be associated with a figure of merit or a fitness value. In the present context, the fitness value of a solution string s_w depends on the total within-cluster variation S (W). Since the objective is to minimize S(W), a solution string with relatively small square error must have relatively high fitness value. There are many ways to defining such a fitness function. We use the α -truncation mechanism for this purpose. Let

$f(sw) = -S(W), g(sw) = f(sw) - (\bar{f} - c.\sigma)$ where

$\bar{f}$ and σ denote the average value and standard deviation of f(*sw*) in the current population, respectively c is constant between 1 and 3. Then, the fitness value of *sw*, F(*sw*), is given by.

$$F(s_w) = \begin{cases} g(sw), & if g(sw) \geq 0 \\ 0, & otherwise \end{cases}$$

3.4 Mutation

Mutation changes an allele value depending on the distances of the cluster centroids from corresponding data point. It may be recalled that each allele corresponds to a point and its value represents the cluster to which the data point belongs. An operator is defined such that the probability of changing an allele value to a cluster number is more if the corresponding cluster center is closer to the data point. To apply the mutation operator to the allele s_w (i) corresponding to pattern x_i, let dj = d(x_i, c_j) be the Euclidean distance between x_i and c_j. Then, the allele is replaced with a value chosen randomly from the following distribution:

$$p_j = \Pr(sw(i) = j) = \frac{c_m d_{\max} - d_j}{\sum_{i=1}^{K} c_m d_{\max} - d_j}$$

where c_m is a constant usually ≥ 1 and $d_{\max} = \max_j d_j$. In case of a partition with one or more than one singleton clusters, the c_m is introduced because; we need p_j to be nonzero for all j to prove the convergence of GKA. It forces c_m to be strictly greater than 1.

Above mutation may result in the formation of empty clusters with a nonzero probability. It may be noted that smaller the number of clusters, larger the SE measure: so empty clusters must be avoided. A quick way of detecting the possibility of empty cluster formation is check whether the distance of the data x_i from its cluster center $C_{sw(i)}$ is greater than zero. It be noted that $d_{sw(i)}$ =0 even in the case of non-singleton clusters were in the data point and the center of the cluster are the same. Thus, an allele is mutated only when $d_{sw(i)} > 0$. The strings that represent K nonempty clusters are called legal strings; otherwise, they are called illegal strings. Each allele in a chromosome is mutated as described above with a probability p_m, called mutation probability. A pseudo-code of the operator is given below.

```
Mutation (sw)
{for i = 1 to n
{if (drand( ) < $P_m$)
{Calculate cluster centers, $c_j$'s,
 corresponding to $s_w$;
 for j = 1 to K, $d_j$ = d($x_i$,$c_j$);
 if ($d_{sw}$(i)> 0)
 {$d_{max}$ = max {$d_1$, $d_2$, ... $d_k$}
 for j = 1 to K,
   $p_j = (c_m d_{\max} - d_j) / \sum_{k=1}^{K} (c_m d_{\max} - d_k)$

 sw(i)= a number, randomly selected from
 {1, 2, ..., K} according to the
 distribution {$p_1$, $p_2$, ..., $p_k$};
            }  } } }
```

(drand( ) returns a uniformly distributed random number in the range [0, 1]).

3.5 k-Means Operator

The algorithm with the above selection and mutation operators may take more time to converge, since the initial assignments are arbitrary and the subsequent changes of the assignments are probabilistic. Moreover, the mutation probability is forced to assume a low value because high values of Pm lead to oscillating behavior of the algorithm. To improve this situation, a one-step k-means algorithm, named k-Means operator (KMO), is introduced. Let sw be a string. The following two steps constitute KMO on sw which yields

1. Calculate cluster centers using (3) for the given matrix W;
2. Reassign each data point to the cluster with the nearest cluster center.

There is a penalty to be paid for the simplicity of this operator. The resulting string may represent a partition with empty clusters, i.e. KMO may result in illegal strings. We convert illegal strings to legal strings by creating desired number of new singleton clusters. This is done by placing in each empty cluster a pattern x from the cluster C with the maximum within-cluster variation (refer to eq. (2)) x is the farthest from the cluster center of the cluster C. We chose to do as above because this technique is found to be effective and computationally less expensive.

3.6 EWGKA

To start with, the initial population is generated as mentioned above and the subsequent populations are obtained by the application of selection, KMO over the previous population. The algorithm is terminated when the limit on the number of generations is exceeded.

4. EXPERIMENT AND RESULT ANALYSIS

We implemented our proposed clustering technique Java language. Our experiments were conducted on a P-IV system with 2.2 Hz CPU and 1 GB RAM. We experiment our clustering algorithm on some standard data sets. These data were taken from the UCI repository [10]. In this paper result analysis on data set Text Data is reported. For each set of the parameters (N, Pm, Gmax), It is ran 100 times. In these tests, we choose a wide range of the mutation probability, and we observed that the average clustering accuracy of algorithm is above 88% and the number of correct clustering is at least 49 out of 100. Because of the limit of the number of generations, the algorithm stops before achieving the global optimum in some cases.

We experimented on text data applying algorithms EWGKA, EWKM, PROCLUS [5] and HARP. We observed that EWKM outperformed PROCLUS and HARP. EWGKA shows good accuracy as EWKM with global optimum convergence. The reason is that other clustering algorithm with sparse problem for high-dimensional data adopted an approximation process to minimize their objective functions so that some raw information may be missed. The clustering accuracy of the two hard subspace clustering algorithm PROCLUS dropped quickly as the sparsity increased. Our observation shows that GEWKM is better in clustering complex data, such as sparse data.

5. CONCLUSTION

The algorithm EWGKM benefits from the advantages of both genetic algorithm (GA) and k-means. Since GA searches the space more thoroughly than k-means, the genetic k-means algorithm will not be trapped in a local optimum. Here we used weight entropy to minimize the within cluster dispersion and maximize the negative weight entropy in the clustering process this way we get more dimensions to make a contribution to identification of each cluster. The problem of identifying cluster by few sparse dimensions can be avoided.

6. REFERENCES

[1] D.S. Modha and W.S. Spangler, Feature weighting in k-Means Clustering,Machine learning, vol. 52, pp.217-237, 2003.

[2] Huan, J.Z., Ng, M.K. Hongqiang Rong, Zichen Li, Automated variable weighting in k-Means type clustering, IEEE Transactions on pattern Analysis and Machine Intelligence, vol. 27 Issue 5, May 2005 pages : 657-668.

[3] J.H. Friedman and J. J. Meulman, Clustering Objects on Subsets on subsets of Attributes, J. Royal Statstical Soc. B., 2002.

[4] K. Krishna and M. N. Murty, Genetic K-Means Algorithm, IEEE Transactions on Systems, Man, and Cybernetics vol. 29, NO. 3, (1999), 433-439.

[5] L. Jing, M. K. Ng and J. Z. Huang , ‘An Entropy weighting k-Means Algorithm for subspace clustering of high dimensional sparse data’, IEEE Transaction on knowledge and Data Engineering Vol 19, No 8, August 2007.

[6] W. Frawley, G.Piatetsky-Shapiro, C. Matheus, Knowledge discovery in databases: an overview, AI Magazine (1992) pp. 213-228.

[7] Y. Lu, S. Lu , F. Fotouhi ,Y. Deng , and S.J. Brown, FGKA: A Fast Genetic K-means Clustering Algorithm, ACM Symposium on Applied Computing ISBN:1-58113-812-1 (2004), 622-623

[8] Y. Lu, S. Lu, F. Fotouhi,Y. Deng and S. J. Brown S. J, Incremental genetic K-means algorithm and its application in gene expression data analysis, BMC Bioinformatics (2004), 5(172).

[9] Z. Yu and H. S. Wong, Genetic based k-means algorithm for selection of feature variables, IEEE ICPR’06, 2006.

[10] S. Hettich and S. D. Bay, The UCI KDD Archive [http://kdd.ics.uci.edu] Invine, CA: University of California, Department of Information and Computer Science.