Extended K-means with an Efficient Estimation of the Number of Clusters

Tsunenori ISHIOKA[1]

National Center for University Entrance Examinations,
2-19-23 Komaba, Meguro-ku, Tokyo 153-8501, JAPAN

Abstract. We present a non-hierarchal clustering algorithm that can determine the optimal number of clusters by using iterations of k-means and a stopping rule based on BIC. The procedure requires twice the computation of k-means. However, with no prior information about the number of clusters, our method is able to get the optimal clusters based on information theory instead of on a heuristic method.

1 Introduction

One of the typical methods for non-hierarchal clustering — k-means — is often used for huge data clustering as well as self-organizing map [8, 9], because it requires only $\mathcal{O}(kN)$ computation for a given number of clusters k and sample size N. In the context of recent research in data mining, several high-performance techniques for k-means have been developed [1, 6].

The different methods for k-means calculations vary in several aspects. In all cases, the problem remains that k-means might not converge to a global optimum, depending on the selection of initial seeds. Nevertheless, from data mining and knowledge discoverly perspective, we are convinced that a pre-determinance of the number of clusters is a strict restriction.

Indeed, we can obtain an optimal number of clusters heuristically by performing computations based on different initial settings of cluster numbers. Hardy [2] surveyed seven typical evaluation criteria (two of them can be applied for hierarchal clustering methods) with various datasets. However, varying the number of clusters requires much computation, because we have to use k-means repeatedly.

We propose an algorithm that initially divides data into clusters whose number is sufficiently small, and continues to divide the each cluster into two clusters. We use BIC (Bayesian Information Criterion[7]) as the division criterion. We will show that the division method works well, and present an implementation. The idea was proposed also by [5], but our method differs in the following aspects:

1. Our method can be applied for general or p-dimensional datasets.
2. We consider the magnitude of variance and covariance around the centers of clusters which can be divided progressively.
3. We evaluate the number of clusters by means of computer simulation runs.

Previous research [5] can treat only two-dimensional datasets, and assumes the variance around the cluster centers to be a constant. As a consequence of progressive division, the number of elements which is contained in each cluster becomes fewer, and the variance will become smaller. Therefore, magnitude of variance should be considered.

In section 2, we describe the principle of k-means and show a proposed algorithm in section 3. In section 4, we evaluate the number of final clusters.

2 K-means method

The procedure of k-means proposed by [4] is as follows:

1. Get the initial k-elements in the dataset, and set them as clusters which consist of one element.
2. Allocate the remaining data to the nearest neighborhood cluster centers.
3. Calculate the cluster centers, and regard them as fixed seeds. Repeat once to allocate the all data to the nearest neighbor cluster seeds.

Most k-means procedures, however, require that the data must be allocated repeatedly until the cluster centers will converge.

3 X-means

Pelleg[5] thought of the basic idea for a 2-division procedure and named it x-means, indicating that the number of clusters with k-means is indefinite. The algorithm of x-means is quite simple; we begin to divide data into clusters whose number is sufficiently small, and continue to divide the cluster into two clusters.

The algorithm proposed in this paper is summarized as follows:

step 0: Prepare p-dimensional data whose sample size is n.

step 1: Set an initial number of clusters to be k_0 (the default is 2), which should be sufficiently small.

step 2: Apply k-means to all data with setting $k = k_0$. We name the divided clusters

$$C_1, C_2, \ldots, C_{k_0}.$$

step 3: Repeat the following procedure from step 4 to step 9 by setting $i = 1, 2, \ldots, k_0$.

step 4: For a cluster of C_i, apply k-means by setting $k = 2$. We name the divided clusters

$$C_i^{(1)}, C_i^{(2)}.$$

step 5: We assume the following p-dimensional normal distribution for the data $\mathbf{x}_i$ contained in C_i:

$$f(\theta_i; \mathbf{x}) = (2\pi)^{-p/2} |\mathbf{V}_i|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x} - \mu_i)^t \mathbf{V}_i^{-1} (\mathbf{x} - \mu_i)\right], \tag{1}$$

then calculate the BIC as

$$\text{BIC} = -2 \log L(\widehat{\theta}_i; \mathbf{x}_i \in C_i) + q \log n_i, \tag{2}$$

where $\widehat{\theta}_i = [\widehat{\mu}_i, \widehat{\mathbf{V}}_i]$ is the maximum likelihood estimate of the p-dimensional normal distribution; μ_i is p-dimensional means vector, and $\mathbf{V}_i$ is $p \times p$ dimensional variance-covariance matrix; q is the number of the parameters dimension, and it becomes $2p$ if we ignore the covariance of $\mathbf{V}_i$. $\mathbf{x}_i$ is the p-dimensional data contained in C_i; n_i is the number of elements contained in C_i. L is the likelihood function which indicates $L(\cdot) = \prod f(\cdot)$.
We choose to ignore the covariance of $\mathbf{V}_i$.

step 6: We assume the p-dimensional normal distributions with their parameters $\theta_i^{(1)}, \theta_i^{(2)}$ for $C_i^{(1)}, C_i^{(2)}$ respectively; the probability density function of this 2-division model becomes

$$g(\theta_i^{(1)}, \theta_i^{(2)}; \mathbf{x}) = \alpha_i [f(\theta_i^{(1)}; \mathbf{x})]^{\delta_i} [f(\theta_i^{(2)}; \mathbf{x})]^{1-\delta_i}, \tag{3}$$

where

$$\delta_i = \begin{cases} 1, \text{ if } \mathbf{x} \text{ is included in } C_i^{(1)}, \\ 0, \text{ if } \mathbf{x} \text{ is included in } C_i^{(2)}; \end{cases} \tag{4}$$

$\mathbf{x}_i$ will be included in either $C_i^{(1)}$ or $C_i^{(2)}$; α_i is a constant which lets equation (3) be a probability density function $(1/2 \leq \alpha_i \leq 1)$. If obtaining a exact value is wanted, we can use p-dimensional numerical integration. But this requires much computation. Thus, we approximate α_i as follows:

$$\alpha_i = 0.5/K(\beta_i), \tag{5}$$

where β_i is a normalized distance between the two clusters, shown by

$$\beta_i = \sqrt{\frac{\|\mu_1 - \mu_2\|^2}{|\mathbf{V}_1| + |\mathbf{V}_2|}}, \tag{6}$$

$K(\cdot)$ stands for an lower probability of normal distribution.
When we set $\beta_i = 0, 1, 2, 3$, α_i becomes $0.5/0.500 = 1$, $0.5/0.841 = 0.59$, $0.5/0.977 = 0.51$, $0.5/0.998 = 0.50$ respectively.
The BIC for this model is

$$\mathrm{BIC}' = -2 \log L'(\widehat{\theta}_i'; \mathbf{x}_i \in C_i) + q' \log n_i, \tag{7}$$

where $\widehat{\theta}_i' = [\widehat{\theta_i^{(1)}}, \widehat{\theta_i^{(2)}}]$ is a maximum likelihood estimate of two p-dimensional normal distributions; since there are two parameters of mean and variance for each p variable, the number of parameters dimension becomes $q' = 2 \times 2p = 4p$. L' is the likelihood function which indicates $L'(\cdot) = \prod g(\cdot)$.

step 7: If BIC $>$ BIC$'$, we prefer the two-divided model, and decide to continue the division; we set

$$C_i \leftarrow C_i^{(1)}.$$

As for $C_i^{(2)}$, we push the p-dimensional data, the cluster centers, the log likelihood and the BIC onto the stack. Return to step 4.

step 8: If BIC $\leq$ BIC$'$, we prefer not to divide clusters any more, and decide to stop.
Extract the stacked data which is stored in step 7, and set

$$C_i \leftarrow C_i^{(2)}.$$

Return to step 4. If the stack is empty, go to step 9.
step 9: The 2-division procedure for C_i is completed. We renumber the cluster identification such that it becomes unique in C_i.
step 10: The 2-division procedure for initial k_0 divided clusters is completed. We renumber all clusters identifications such that they become unique.
step 11: Output the cluster identification number to which each element is allocated, the center of each cluster, the log likelihood of each cluster, and the number of elements in each cluster. [stop]

The reasons why we choose BIC over other common information criteria for model selection are follows:

- BIC considers the selection among from exponential family of distributions.
- BIC is based on prior probability rather than the distance between two distributions.

4 Evaluation of the performance

4.1 An investigation of the number of generated clusters

A simulation procedure is adopted. It generates 250 two-dimensional normal variables; these random variables should be clustered into 5 groups. Each group consists of 50 elements:

$$\begin{aligned}
x_j &\sim N(\mu = [0, 0],\ \sigma = [0.2, 0.2]),\ (j = 1, \ldots, 50)\\
x_j &\sim N(\mu = [-2, 0],\ \sigma = [0.3, 0.3]),\ (j = 51, \ldots, 100)\\
x_j &\sim N(\mu = [2, 0],\ \sigma = [0.3, 0.3]),\ (j = 101, \ldots, 150)\\
x_j &\sim N(\mu = [0, 2],\ \sigma = [0.4, 0.4]),\ (j = 151, \ldots, 200)\\
x_j &\sim N(\mu = [0, -2],\ \sigma = [0.4, 0.4]),\ (j = 201, \ldots, 250)
\end{aligned}$$

where μ is a mean, and σ^2 is a variance. We set $k_0 = 2$ as an initial division, and performed 1,000 simulation runs of x-means. Two-dimensional normal variables are generated for each simulation run. X-means will call k-means repeatedly; the algorithm of k-means is based on [3], which is provided in R.

Table 1 summarizes the number of clusters generated by x-means (upper row). For 1,000 simulation runs, the most frequent case is when 5 clusters are generated, this occurs 533 times. The second most frequent case is 6 clusters, which occurs 317 times. The middle row shows the results applying AIC (Akeike's Information Criterion) instead of BIC to x-means. We found that x-means by AIC tends to overgenerate clusters. The bottom row in Table 1 shows the number of optimal clusters when the goodness of model for give data is maximum (i.e.,

Table 1. The number of clusters by using 250 random variables of two-dimensional normal distribution

number of clusters	2	3	4	5	6	7	8	9	10	11	12	13	14	total
x-means (BIC)	2	6	9	469	383	99	27	5	0	0	0	0	0	1,000
x-means (AIC)	2	1	1	322	295	162	93	54	36	17	11	2	4	1,000
heuristic method	0	2	37	559	265	90	35	8	4	0	0	0	0	1,000

the AIC for given data is minimum) by varying k applied to k-means. This distribution is very similar to the distribution in upper row.

The cluster centers found by k-means are not always located where the elements cohere; thus x-means often divides a cluster into two clusters until new clusters centers will converge where the elements cohere. Consequently, x-means produces rather more clusters than adequate. Actually in our simulation, when x-means divides all 250 ($= 50 \times 50$)data into two clusters equally (i.e, 125 elements each), both subclusters are often divided into three clusters ($50+50+25$), resulting in 6 clusters.

4.2 An investigation of the number of cluster elements

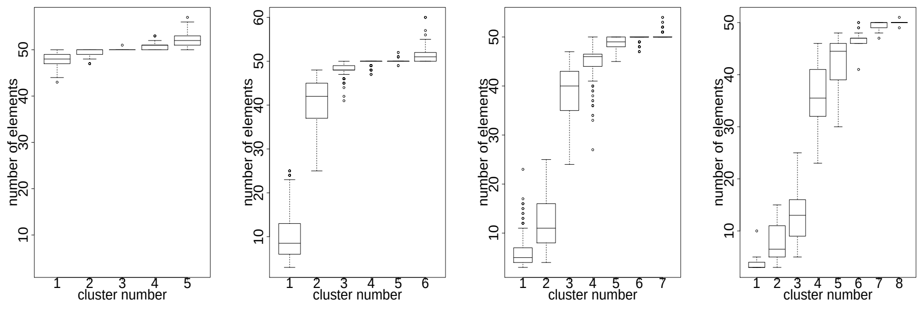
After applying x-means to the simulation runs, we can obtain the distributions of the number of cluster elements, as shown in Fig.1. The horizontal axis gives the cluster identification number, which is sorted in increasing order by the number of cluster elements; the vertical axis gives the distribution of number of the cluster elements; box-and-whisker charts are used.

A box-and-whisker chart contains a box surrounding two hinges, two whiskers, and outlier(s) if any; a lower or upper hinge shows 25 or 75 percentile of the distribution; the median(50 percentile) is in between two hinges. The two whiskers stands for tails of the distribution; the whiskers extend to the most extreme data point which is no more than 1.5 times interquartile range from the box; the outlier(s) may be shown if any.

In case (a), i.e, when obtaining 5 clusters, we found that each cluster consists of about 50 elements. In case (b) obtaining 6 clusters, 4 clusters consist of about 50 elements and the remainder is divided into 2 clusters. Case (c), obtaining 7 clusters, is similar to (b); 3 clusters consist of about 50 elements and the remainder is divided into 4 clusters. For cases (b), (c), and (d), the proper division in clusters of 50 was performed, although the generated cluster may be rather small.

4.3 Consideration of the computational amount

X-means requires to find k final clusters, even if it repeats to divide into two clusters. In addition, we need to judge if these k final clusters should not be divided any more. Thus, remembering that k-means requires $\mathcal{O}(kN)$ computation, x-means will take twice as much computation compared to k-means.



(a) 5 clusters (b) 6 clusters (c) 7 clusters (d) 8 clusters

Fig. 1. Distribution of the number of cluster elements contained in final clusters

Indeed the computation of BIC is needed, but we can ignore this because we calculate it only once after fixing the cluster elements. The BIC can be easily obtained from the mean and variance-covariance of a p-dimensional normal distribution. We are convinced that x-means gives us a quite good solution which meets with its computational expense, although the solution may not be an optimum. This program can be obtained via http://www.rd.dnc.ac.jp/˜tunenori/src/xmeans.prog.

References

1. Huang, Zhexue: Extension to the k-Means Algorithm for Clustering Large Data Sets with Categorical Values, *Data Mining and Knowledge Discovery*, **2**[3] (1998) 283–304.
2. Hardy, Andre: On the Number of Clusters, *Computational Statistics & Data Analysis*, **1**[23] (1996) 83–96.
3. Hartigan, J.A. and Wong, M.A. : A K-means clustering algorithm. *Applied Statistics* 28 (1979) 100–108.
4. MacQueen, J.B.: Some methods for Classification and Analysis of Multivariate Observations," *Proc. Symp. Math. Statist. and Probability, 5th Berkeley*, **1** (1967) 281–297.
5. Pelleg, Dan and Andrew Moore: X-means: Extending K-means with Efficient Estimation of the Number of Clusters, *ICML-2000* (2000).
6. Pelleg, Dan and Andrew Moore: Accelerating Exact k-means Algorithms with Geometric Reasoning, *KDD-99* (1999).
7. Schwarz, G.: Estimating the dimension of a model, *Ann. Statist.*, 6–2: (1978) 461-464.
8. Vesanto, Juha and Johan Himberg and Esa Alhoniemi and Juha Parhankangas: Self-Organizing Map in Matlab: the SOM Toolbox, *Proceedings of the Matlab DSP Conference 1999*, Espoo, Finland, November (1999) 35–40.
9. Yang, Ming-Hsuan and Narenda Ahuja: A Data Partition Method for Parallel Self-Organizing Map, *Proceeding of the 1999 IEEE International Joint Conference on Neural Networks (IJCNN 99)*, Washington DC, July (1999).