Proceedings of the Fourth IASTED International Conference
COMPUTATIONAL INTELLIGENCE
July 4-6, 2005, Calgary, Alberta, Canada

AN EXPANSION OF X-MEANS FOR AUTOMATICALLY DETERMINING THE OPTIMAL NUMBER OF CLUSTERS

— PROGRESSIVE ITERATIONS OF K-MEANS AND MERGING OF THE CLUSTERS —

Tsunenori Ishioka
Research Division
National Center for University Entrance Examinations
2-19-23 Komaba, Meguro-ku,
Tokyo 153-8501, Japan
email: tunenori@rd.dnc.ac.jp

ABSTRACT
We expand a non-hierarchical clustering algorithm that can determine the optimal number of clusters by using iterations of k-means and a stopping rule based on Bayesian Information Criterion (BIC). The procedure requires merging the clusters that a k-means iteration has made to avoid unsuitable division caused by the division order. By using this additional merging operation, the case of adequate clustering was increased for various types of simulation runs. With no prior information about the number of clusters, our method can get the optimal clustering based on information theory instead of on a heuristic method. The computational complexity of our method is $\mathcal{O}(N \log k)$ for the sample size N and the number of final clusters, k.

KEY WORDS
Non-hierarchical clustering, Information criterion, BIC, Feedback operation, Computer simulation

1 Introduction

Clustering is an important technique in the data-mining area [1]. If data mining is considered a knowledge discovery from amount of data, we can obtain useful information from the number of clusters into how many clusters the whole data can be divided, as well as the available maximum sample size and the computational complexity.

Four methods have been developed for finding the number of optimal clusters when no prior information is available about the number of clusters.

1. A method that finds the optimal number of clusters heuristically by using appropriate information criterion based on a different setting of cluster numbers.

2. A method using the minimum volume ellipsoid (MVE) estimator [7].

3. A method that starts with a cluster division of more than the optimal solution. A suitable number of clusters is determined by merging near clusters and/or removing spurious clusters [8].

4. A method that repeats two divisions according to a k-means method until the division is not judged to be appropriate after classifying it into a sufficiently small number of clusters that are made by the first k-means method [11].

The first method is the simplest and most authentic, but the function that evaluates the goodness-of-fit to the model, such as an information criterion, does not necessarily become convex against the number of clusters. In addition, since a lot of clusterings are possible for a huge amount of data, many function values associated with each clustering should be evaluated. Therefore, this is not realistic from the point of view of the computational amount.

Hardy [3] surveyed seven typical evaluation criteria, two of which 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.

The second method [7] determines a single ellipsoid and removes it one by one from the whole, using a Kolmogrov-Smirnov goodness-of-fit test. Each cluster is the basis of assumption of being with an ellipsoid. However, designing validity measures that perform well on a variety of data sets for this method is well known to be difficult, and it is very weak for noise contamination [9].

In the third method [8], not all data is classified exclusively; the data considered as an outlier is eliminated from a cluster. In data mining or data detection, we do not prefer this method because an outlier gives us important information.

The fourth method is called the x-means because the number of final clusters is unfixed. In the context of recent research into data mining, several high-performance techniques for the k-means, the basis of the x-means, have been developed along with self-organizing maps [12, 13]. Pelleg [10] showed a great reduction in effort for updating cluster centers by storing sufficient statistics in kd-trees; Huang [5] presents a clustering technique for large datasets with categorical values; BIRCH, proposed by Zhang [14], can typically find good clusters with a single scan of data and can improve the quality further with a few additional

scans. BIRCH stores summary information on data called the Clustering Feature Tree (CF-Tree), and it limits subsequent operations only to this CF-tree. Since this CF-tree is sufficiently small compared with N, the first scanning time of $\mathcal{O}(N)$ becomes the whole computational amount, and it can be held on main memory for treating even a huge dataset.

Thus, considering that research has been progressing on the k-means, the author previously supported method 4 and improved the following aspects of it [6]:

1. The magnitude of variance and covariance around the centers of clusters that can be divided progressively was considered. This is a vital improvement.
2. This method can be applied for general or p-dimensional datasets.

In our implementation, we do not use a recursive functional call when dividing data into two clusters. Instead, we continue to divide one cluster, and push the other cluster onto the stack. After no further clusters need to be divided, the stacked cluster can be dealt with. Therefore, we can avoid the great overhead time associated with the functional calls if the nesting of the division is deep.

This method has worked well for various kinds of datasets. However, the k-means assumes that each cluster has a spherical form and that each amount of the data in a cluster is almost equal. Thus, recognizing the structure contrary to this assumption is difficult. Guha [2] illustrated the results of having applied the k-means to the data in which three dense domains exist, as can be seen in Fig. 1. The biggest domain of the amount of data is divided into three, and two comparatively small domains are not divided into two (Fig. 1(a)). This differs from what is understood.

The results of having applied the x-means as initial division $k_0 = 2$ to this same data are shown in Fig. 1(b). The whole is classified into four clusters; the biggest domain is divided into two due to the shortcomings that the k-means method has. And two domains, where the amount of data is small, are divided into two. This is consistent with what is understood. It turns out that the x-means can overcome the k-means' shortcomings to a certain extent, but not completely.

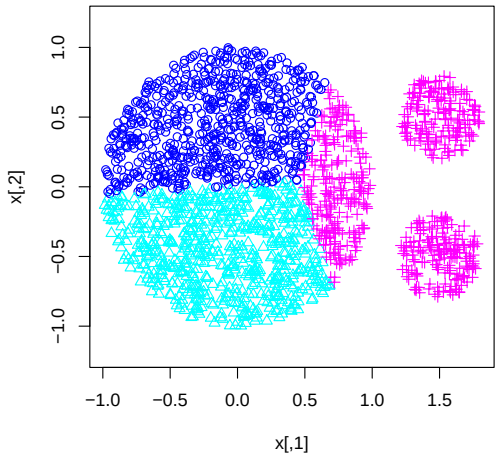
We therefore present on additional merging operation that functions after the x-means divide the clusters. We show that the procedure works well and present the results of the performance of various simulation runs.

In Section 2, the outlines of the x-means are described, and an algorithm is presented. In Section 3, the number of clusters is evaluated. Section 4 concludes the paper.

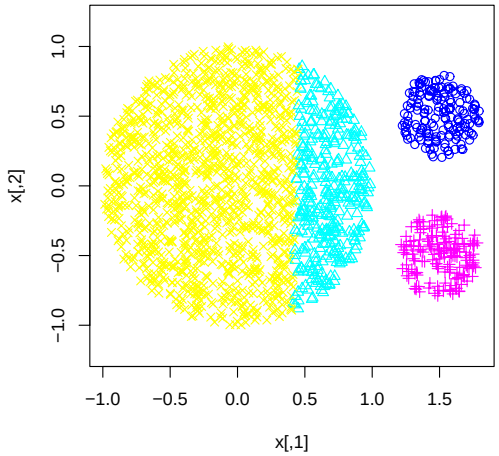
2 x-means

The algorithm proposed in [6] is summarized as follows:

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



(a) k-means $(k = 3)$



(b) x-means $(x = 4)$

Figure 1. Splitting of a large cluster by partional algorithm

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 $\boldsymbol{x}_i$ contained in C_i:

$$f(\boldsymbol{\theta}_i; \boldsymbol{x}) = (2\pi)^{-p/2} |\boldsymbol{V}_i|^{-1/2} \times \exp\left[-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu}_i)^t \boldsymbol{V}_i^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_i)\right],$$

then calculate the BIC as

$$\mathrm{BIC} = -2\log L(\widehat{\boldsymbol{\theta}}_i; \boldsymbol{x}_i \in C_i) + 2p\log n_i,$$

where $\widehat{\boldsymbol{\theta}}_i = [\widehat{\boldsymbol{\mu}_i}, \widehat{\boldsymbol{V}_i}]$ is the maximum likelihood estimate of the p-dimensional normal distribution; $\boldsymbol{\mu}_i$ is the p-dimensional means vector, and $\boldsymbol{V}_i$ is the $p \times p$ dimensional variance-covariance matrix; the total number of the parameters is $2p$. $\boldsymbol{x}_i$ 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 $L(\cdot) = \prod f(\cdot)$.

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

$$g(\boldsymbol{\theta}_i^{(1)}, \boldsymbol{\theta}_i^{(2)}; \boldsymbol{x}) = \alpha_i [f(\boldsymbol{\theta}_i^{(1)}; \boldsymbol{x})]^{\delta_i} [f(\boldsymbol{\theta}_i^{(2)}; \boldsymbol{x})]^{1-\delta_i}, \quad (1)$$

where

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

$\boldsymbol{x}_i$ will be included in either $C_i^{(1)}$ or $C_i^{(2)}$; α_i is a constant which lets equation (1) be a probability density function; that is

$$\alpha_i = 1 / \int [f(\boldsymbol{\theta}_i^{(1)}; \boldsymbol{x}_i)]^{\delta_i} [f(\boldsymbol{\theta}_i^{(2)}; \boldsymbol{x}_i)]^{1-\delta_i} \mathrm{d}\boldsymbol{x},$$

$(1/2 \leq \alpha_i \leq 1)$. If obtaining an exact value is wanted, we can use p-dimensional numerical integration. But this requires significant computation.

Thus, we approximate α_i as follows:

$$\alpha_i = 0.5 / K(\beta_i),$$

where

$$\beta_i = \sqrt{\frac{\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\|^2}{|\boldsymbol{V}_1| + |\boldsymbol{V}_2|}}.$$

$K(\cdot)$ stands for a lower probability of normal distribution.

When we set $\beta_i = 0, 1, 2, 3$, α_i becomes $0.5/0.500 = 1$, $0.5/0.841 = 0.59$, $0.5/0.977 = 0.51$, and $0.5/0.998 = 0.50$, respectively.

The BIC for this model is

$$\mathrm{BIC}' = -2\log L'(\widehat{\boldsymbol{\theta}}_i'; \boldsymbol{x}_i \in C_i) + 4p\log n_i,$$

where $\widehat{\boldsymbol{\theta}}_i' = [\widehat{\boldsymbol{\theta}_i^{(1)}}, \widehat{\boldsymbol{\theta}_i^{(2)}}]$ 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 total number of parameters becomes $2 \times 2p = 4p$. L' is the likelihood function which indicates $L'(\cdot) = \prod g(\cdot)$.

step 7: If $\mathrm{BIC} > \mathrm{BIC}'$, the two-divided model is preferred, and the division is continued; 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 $\mathrm{BIC} \leq \mathrm{BIC}'$, clusters are no longer divided.

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 BIC was chosen over other common information criteria for model selection are follows:

- BIC considers the selection among from exponential family of distributions, to which normal distribution belongs.
- BIC is based on prior probability rather than the distance between two distributions.

BIC is clearly the best stopping criterion for our 2-division procedure.

In our implementation, we do not use a recursive functional call when dividing into two clusters. Instead, we continue to divide one cluster, and push the other cluster onto the stack; after there are no further clusters to be divided, the stacked cluster can be dealt with. We can avoid the great overhead time associated with the functional calls if the nesting of the division is deep.

Now, the biggest domain is divided into two clusters in the example of Fig. 1(b), because it was divided so that the amount of data might become equal by the first division. If we have to choose the biggest domain is better based on whether it is divided into two or is collected into one, we will choose the latter. Therefore, additional procedures between **steps 10** and **11** are presented here.

step 10-2: We denote n_i as the number of data items contained in C_i and sort them in a descending order. The row of subscript i is set to $\boldsymbol{I} = (I_1, I_2, \ldots, I_k)$. We set the subscript of I to i and j anew $(i, j = 1, \ldots, k)$.

Perform the following operations as a round robin for i and j, where we chose $i < j$ so that the number of C_i is less than that of C_j.

Compare BIC applied for independent C_i to BIC$'$ for merged C_i and C_j. If BIC $>$ BIC$'$, then C_i should be merged into C_j.

However, this merging operation is restricted to one time at most for any combination of i and j.

The procedure when considering $n_1 = 15,\ n_2 = 50,\ \text{and}\ n_3 = 10$ as an example is as follows. When n_i is sorted in descending order, because it becomes $n_3 < n_1 < n_2$, we obtain $\boldsymbol{I} = (3, 1, 2)$. Therefore, we should evaluate it in order of

$$(i, j) = (3, 1),\ (3, 2),\ (1, 2).$$

We first compare two BICs for $(i, j) = (3, 1)$. If the relationship of

$$(\text{BIC to } C_3) > (\text{BIC to } C_3 \text{ and } C_1) \tag{2}$$

is found, C_3 will be merged to C_1.

Next, we compare two BICs for (i, j) = (3, 2). If the relationship of formula (2) is found, because C_3 is already merged into C_1, this operation is skipped; we do not evaluate this. Otherwise, we evaluate the relationship of

$$(\text{BIC to } C_3) > (\text{BIC to } C_3 \text{ and } C_2). \tag{3}$$

If it is found, C_3 will be merged to C_2.

Finally we evaluate (i, j) = (1, 2). If the relationship of (2) or (3) is found, because C_1 or C_2 already has been merged, this operation is skipped. That is, further annexation is not performed on the merged cluster.

When this operation was applied to the data given by Fig. 1, being divided into three clusters is adequate, and it occurred 532 times across 1,000 simulations where the initial point of the k-means was changed. Without a merging operation, three adequate clusters can never be obtained.

3 Evaluation of the performance

3.1 An investigation of the number of generated clusters

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

$$\begin{aligned}
&x_j \sim N(\boldsymbol{\mu} = [0, 0],\ \boldsymbol{\sigma} = [0.2, 0.2]),\ (j = 1, \ldots, 50)\\
&x_j \sim N(\boldsymbol{\mu} = [-1, -1],\ \boldsymbol{\sigma} = [0.2, 0.2]),\ (j = 51, \ldots, 100)\\
&x_j \sim N(\boldsymbol{\mu} = [1, 1],\ \boldsymbol{\sigma} = [0.2, 0.2]),\ (j = 101, \ldots, 150)\\
&x_j \sim N(\boldsymbol{\mu} = [2, 2],\ \boldsymbol{\sigma} = [0.2, 0.2]),\ (j = 151, \ldots, 200)\\
&x_j \sim N(\boldsymbol{\mu} = [3, 3],\ \boldsymbol{\sigma} = [0.2, 0.2]),\ (j = 201, \ldots, 250),
\end{aligned}$$

where $\boldsymbol{\mu}$ is a mean, and $\boldsymbol{\sigma}^2$ is a variance. We set $k_0 = 2$ as an initial division, and performed 1,000 simulation runs of the x-means. Two-dimensional normal variables were generated for each simulation run. X-means calls k-means repeatedly; the algorithm of the k-means is based on Hartigan [4]. The program is provided in R. Figure 2 is an example of the random variables.

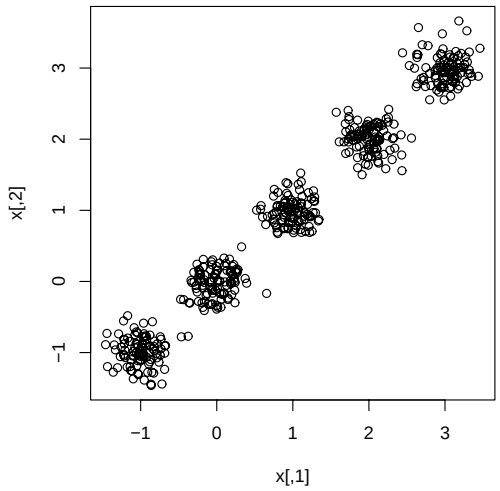


Figure 2. Example of 2-dimensional normal variables whose cluster centers form a line

Table 1 summarizes the number of clusters generated by the x-means. The upper row shows authentic x-means without merging operations. For 1,000 simulation runs, the most frequent case is when 5 clusters were generated; this occurred 533 times. The second most frequent case is 6 clusters, which occurred 317 times. The lower row shows the results applying the described x-means with merging operations. The most frequent case is when 5 clusters were generated; this occurred 909 times. We found that an authentic x-means tends to overgenerate clusters, while the new x-means can overcome this.

Table 1. Number of clusters by using 250 random variables of two-dimensional normal distribution, where cluster centers form a straight line

x-means method	4	5	6	7	8	9	total
authentic [6]	0	533	317	108	34	8	1,000
new	0	909	86	5	0	0	1,000

The cluster centers found by k-means are not always located where the elements cohere; thus, the authentic x-means often divides a cluster into two until new clusters centers converge where the elements cohere. Consequently, an authentic x-means produces more clusters than adequate. In our actual simulation, when the x-means divided all 250 $(= 50 \times 5)$ items of data into two clusters equally (i.e, 125 elements each), both subclusters were often divided into three clusters $(50 + 50 + 25)$, resulting in 6 clusters. The new procedure can merge two split clusters of

$(25 + 25)$.

(2) It generates 300 two-dimensional normal variables; these random variables should be clustered into 5 groups:

$$
\begin{aligned}
x_j &\sim N(\boldsymbol{\mu} = [0, 0],\ \boldsymbol{\sigma} = [0.2, 0.2]),\ (j = 1, \ldots, 100) \\
x_j &\sim N(\boldsymbol{\mu} = [-2, 0],\ \boldsymbol{\sigma} = [0.3, 0.3]),\ (j = 101, \ldots, 150) \\
x_j &\sim N(\boldsymbol{\mu} = [2, 0],\ \boldsymbol{\sigma} = [0.3, 0.3]),\ (j = 151, \ldots, 200) \\
x_j &\sim N(\boldsymbol{\mu} = [0, 2],\ \boldsymbol{\sigma} = [0.4, 0.4]),\ (j = 201, \ldots, 250) \\
x_j &\sim N(\boldsymbol{\mu} = [0, -2],\ \boldsymbol{\sigma} = [0.4, 0.4]),\ (j = 251, \ldots, 300).
\end{aligned}
$$

One of the groups $(j = 1, \ldots, 100)$ consists of 100 elements, and the others consist of 50 elements each. The cluster centers of two-dimensional normal variables in (1) form a straight line. On the other hand, the cluster centers in (2) are in the shape of a cross. Moreover, the variance of two-dimensional normal variables in each group is not constant. An example of the random variables is Fig. 3. Table 2 shows the results.

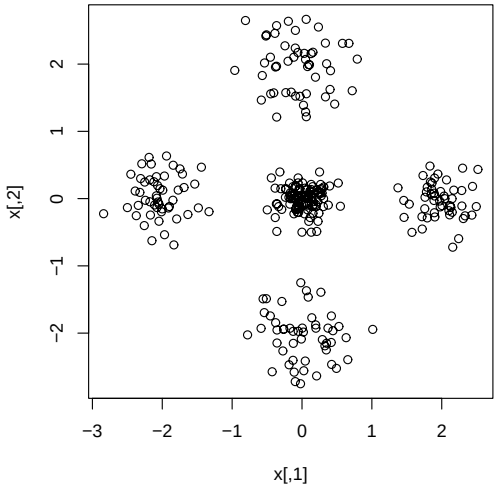


Figure 3. Example of 2-dimensional normal variables whose cluster centers are cross shaped

Table 2. Number of clusters by using 300 random variables of two-dimensional normal distribution, where cluster centers form a cross

x-means method	2	3	4	5	6	7	8	9	total
authentic [6]	2	6	9	469	383	99	27	5	1,000
new	22	5	0	890	74	9	0	0	1,000

In the new method, the most frequent case is when 5 clusters were generated; this occurred 890 times. Many cases where 6 or more clusters were generated in the authentic method have been translated to the category of 5 clusters.

(3) A total of 300 two-dimensional normal random variables with a correlation of 0.5 were examined. An example of which has the covariance not equal to zero. The mean and the variance of each group are the same as the example of (2). Figure 4 shows an example of random variables used in this simulation. Table 3 shows the results.

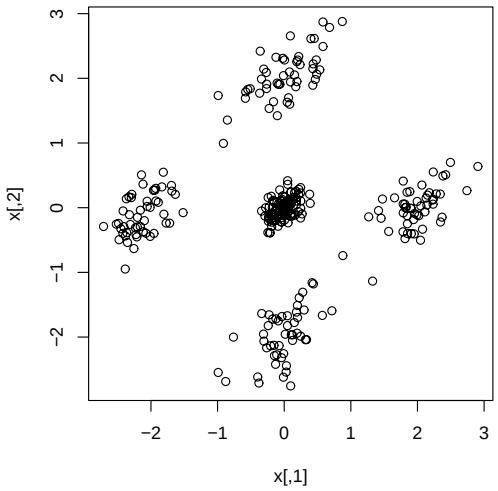


Figure 4. Example of 2-dimensional normal variables with correlation of 0.5

Table 3. Number of clusters by using 300 random variables of two-dimensional normal distribution with correlation of 0.5, where cluster centers form a cross

x-means method	2	3	4	5	6	7	8	9	11-12	total
authentic [6]	13	9	1	345	388	166	61	12	5	1,000
new	17	4	0	638	273	59	8	1	0	1,000

In the new method, the most frequent case is when 5 clusters were generated; this occurred 638 times. The second most frequent case is when 6 clusters, which occurred 273 times. The number of adequate clusterings is less than that obtained by (2), but it has the same tendency as (2); that is, there are many cases where 6 or more clusters generated in the authentic method are improved.

3.2 Consideration of the computational amount

We should know that the computational complexity of the k-means is $\mathcal{O}(kN)$, for a given number of clusters k and sample size N. The x-means is a repetition of the k-means, and the clusters of k are finally discovered. Thus, if a 2-division tree is assumed to be balanced, the k-means ($k = 2$) will be performed once for N of the sample size, and twice for $N/2$ and 4 times for $N/4$. After all, the total of the k-means computational complexity in each level does not depend on k, but becomes $\mathcal{O}(N)$.

On the other hand, the depth of the levels in a 2-division tree is $(\log k)/(\log 2)$ on the average when assuming the depth of the root is zero, because the number of the node (cluster) at the lowest layer is 2^k. It turns out that the computational complexity of the x-means is $\mathcal{O}(N \log k)$.

In usual cases, however, the size of k is much smaller than N. Therefore, the size of $\log k$ is not considered advantageous. The computational amount of the x-means becomes much larger than the k-means does.

The following are the reasons:

1. After optimal clusters division was performed, each cluster needs to be divided further, and the division needs to be evaluated as being unsuitable. Namely, when the optimal number of the cluster is k, it needs to be divided into $(2k)$.
2. BIC needs to be evaluated as an information criterion whether the division is adequate.

As for the computational amount in **step 10-2** that was appended, we need the $k(k-1)/2$ times computation of BIC. However, we can ignore this as a whole because the BIC had been already calculated and the k is sufficient smaller than N.

4 Conclusion and remarks

The performance of x-means is markedly improved by evaluating created clusters with a 2-division procedure, and after that, adding an operation wherein the cluster is annexed if required. That is, we have solved the problem of the k-means, whose division is based on a tacit assumption where the amount of data in each cluster is almost equal.

Our method uses the k-means repeatedly. Since less computationally-intensive k-means are being developed rapidly, we may be able to utilize these results. New k-means programs written in languages such as C or Fortran can be easily embedded into our system. Our program coded in R and Fortran (g77) can be obtained via http://www.rd.dnc.ac.jp/~tunenori/xmeans_e.html. The fortran source program is compressed by tar+gzip on linux. You can easily unpack this using the standard GNU procedure; that is, ./configure; make; make install.

Acknowledgements

This work was supported by a Grant-in-Aid for Scientific Research No. 16500628 to Tsunenori Ishioka.

References

[1] U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, From data mining to knowledge discovery: An overview, in U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy (Ed.) *Advances in Knowledge Discovery and Data Mining,* (AAAI Press/The MIT Press, 1996) 1–34.

[2] S. Guha, R. Rastogi, and K. Shim, CURE: An efficient clustering algorithm for large databases, *Proc. the ACM SIGMOD International Conference on Management of Data*, 1998, 73–84.

[3] A. Hardy, On the number of clusters, *Computational Statistics & Data Analysis*, l(23), 1996, 83–96.

[4] J.A. Hartigan and M.A. Wong, A k-means clustering algorithm. *Applied Statistics* 28, 1979, 100–108.

[5] Z. Huang, Extension to the k-means algorithm for clustering large data sets with categorical values, *Data Mining and Knowledge Discovery*, 2(3), 1998, 283–304.

[6] T. Ishioka, Extended k-means with an efficient estimation of the number of clusters, Intelligent Data Engineering and Automated Learning — IDEAL 2000, Second International-Conference, Hong Kong, China, 2000, 17–22.
(*Lecture Notes in Computer Science* 1983, Kwong Sak Leung, Lai-Wan Chan, and Helen Meng (Ed.), Springer, 2000)

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

[8] R. Krishnapuram, and C.P. Freg, Fitting an unknown number of lines and planes to image data through-compatible cluster merging, *Pattern recognition*, 25, 1992, 385–400.

[9] O. Nasraoui, E. Leon and R. Krishnapuram, Niche clustering: Discovering an unknown number of clusters in noisy data sets, in A. Ghosh and L.C. Jain (Ed.) *Evolutionary Computing in Data Mining* (Springer Verlag, 2004) 157–188.

[10] D. Pelleg, and A. Moore: Accelerating exact k-means algorithms with geometric reasoning, *KDD-99*, 1999, 277-281.

[11] D. Pelleg, and A. Moore, X-means: Extending k-means with efficient estimation of the number of clusters, *17th International Conf. on Machine Learning*, 2000, 727–734,

[12] J. Vesanto, J. Himberg, E. Alhoniemi and J. Parhankangas, Self-organizing map in matlab: the SOM toolbox, *Proc. the Matlab DSP Conference 1999*, Espoo, Finland, 1999, 35–40.

[13] M.-H. Yang, and N. Ahuja: A data partition method for parallel self-organizing map, *Proc. the 1999 IEEE International Joint Conference on Neural Networks (IJCNN 99)*, Washington DC, USA, 1999.

[14] T. Zhang, R. Ramakrishnan and M. Livny: BIRCH: An efficient data clustering method for very large databases, *SIGMOD Conf*. 1996, 103–114.