The Minimum Code Length for Clustering Using the Gray Code

Mahito Sugiyama[1,2] and Akihiro Yamamoto[1]

[1] Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, 606-8501, Japan
mahito@iip.ist.i.kyoto-u.ac.jp,
akihiro@i.kyoto-u.ac.jp
[2] Research Fellow of the Japan Society for the Promotion of Science

Abstract. We propose new approaches to exploit compression algorithms for clustering numerical data. Our first contribution is to design a measure that can score the quality of a given clustering result under the light of a *fixed* encoding scheme. We call this measure the *Minimum Code Length* (MCL). Our second contribution is to propose a general strategy to translate any encoding method into a cluster algorithm, which we call COOL (COding-Oriented cLustering). COOL has a low computational cost since it scales linearly with the data set size. The clustering results of COOL is also shown to minimize MCL. To illustrate further this approach, we consider the *Gray Code* as the encoding scheme to present G-COOL. G-COOL can find clusters of arbitrary shapes and remove noise. Moreover, it is robust to change in the input parameters; it requires only two lower bounds for the number of clusters and the size of each cluster, whereas most algorithms for finding arbitrarily shaped clusters work well only if all parameters are tuned appropriately. G-COOL is theoretically shown to achieve internal cohesion and external isolation and is experimentally shown to work well for both synthetic and real data sets.

Keywords: Clustering, Compression, Discretization, Gray code.

1 Introduction

Clustering is a fundamental task in data analysis, and many clustering algorithms have been developed in the fields of machine learning and knowledge discovery [1,9,14]. Several clustering algorithms have been recently proposed that focus on the *compression* of data points.

Kontkanen *et al.* [19] proposed the *minimum description length* (MDL) approach to clustering taking advantage of an information theoretic framework. However, data encoding has to be optimized to find the best clusters; that is, all encoding schemes are considered within the clustering process under the MDL criterion. As a result, their approach takes quadratic time with respect to the data set size and can only be handled in practice using a stochastic algorithm [18]. Cilibrasi and Vitányi [6] proposed a clustering algorithm based on the *Kolmogorov complexity*. Since their method measures the distance between two data points on the basis of compression of finite sequences (*i.e.*,

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 365–380, 2011.

discrete variables), it is difficult to apply it to multivariate data of continuous variables. Moreover, although there are other approaches [16,20] that focus on compression of data, they perform simple agglomerative hierarchical clustering, so it takes quadratic time with respect to the data set size. These approaches are therefore not suitable for clustering massive data sets.

Here we propose a new measure, called the *minimum code length* (MCL), to score the quality of a given clustering result under a *fixed* encoding scheme. This use of fixed encoding enables the performance of fast (*i.e.*, linear complexity with respect to the data set size) and exact clustering since we do not need to optimize data encoding. We present a clustering algorithm, called COOL (COding-Oriented cLustering), that always finds the best clusters; *i.e.*, the globally optimal clusters which minimizes MCL, and requires only the lower bounds for the number and size of the clusters. The discretization of continuous variables with the fixed encoding scheme coincides with the clustering process itself — a hierarchy of clusters is introduced automatically by increasing the accuracy of discretization.

Mathematically, an encoding, or *embedding*, is a mapping from real numbers to infinite sequences over some alphabet [29], and discretization is realized by truncation of infinite sequences. For example, in the binary embedding γ_{B}, every real number in $[0, 1]$ is translated into an infinite sequence composed of `0` and `1`; *e.g.*, $\gamma_{\mathrm{B}}(0) = \mathtt{000}\ldots$, $\gamma_{\mathrm{B}}(0.2) = \mathtt{001}\ldots$, and $\gamma_{\mathrm{B}}(0.4) = \mathtt{011}\ldots$, where the first bit is `0` if the value is in the interval $[0, 0.5]$ and `1` if in $(0.5, 1]$. If these sequences are truncated at the first bit, all of them become `0`, and hence they are considered as in the same cluster since they cannot be distinguished. If they are then truncated at the second bit, both 0 and 0.2 become `00`, and 0.4 becomes `01`. Thus, two clusters are generated: $C_1 = \{0, 0.2\}$ and $C_2 = \{0.4\}$. This means that representatives of C_1 and C_2 are `00` and `01`, respectively. Finally, if they are truncated at the third bit, 0 and 0.2 are separated. The hierarchy is therefore constructed as $\{\{0, 0.2, 0.4\}\}$, $\{\{0, 0.2\}, \{0.4\}\}$, and $\{\{0\}, \{0.2\}, \{0.4\}\}$.

The complexity of making clusters can be measured by the length of the cluster representatives. In the above example, $2 + 2 = 4$ for clusters $\{0, 0.2\}$ and $\{0.4\}$ (`00` and `01`), and $3 + 3 + 2 = 8$ for $\{0\}$, $\{0.2\}$, and $\{0.4\}$ (`000`, `001`, and `01`). We call these values the MCL since we cannot distinguish a pair of data points from different clusters if their truncated codes are shorter than the MCL.

Since COOL does not optimize an embedding scheme within the clustering process, the clustering result strongly depends on the embedding used. This means that we have to carefully choose an appropriate embedding for effective clustering. In this paper, we consider the *Gray code* as an embedding scheme for COOL — resulting in an algorithm we call G-COOL. Gray code was originally developed for binary encoding of natural numbers and has become especially important in applications requiring conversion between analog and digital information [17]. From the geometrical point of view, Gray code embedding is the partitioning of each interval into *overlapping* smaller intervals. This enables clusters with arbitrary shapes to be found, which cannot be done with binary embedding. There is theoretical support for clustering by G-COOL as shown in Lemma 2 and Theorem 1. Figure 1 illustrates examples of computing the MCL with binary and Gray code embedding.

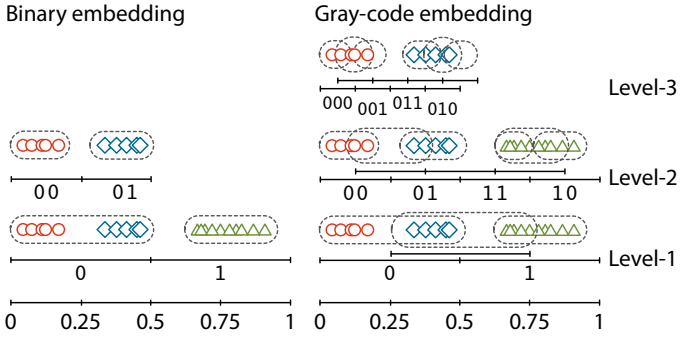

Fig. 1. Examples of computing MCL with binary (left) and Gray code (right) embedding. These one-dimensional data sets are in $[0, 1]$ and partitioned into three clusters, ◯, ♢, and △. Level means the length of each prefix. With binary embedding, cluster △ is separated at level 1, and ◯ and ♢ are separated at level 2. They are encoded by `1`, `00`, and `01`, respectively, so the MCL is $1 + 2 + 2 = 5$. With Gray code embedding, the intervals overlap, and adjacent clusters are merged at each level. As a result, △ is separated at level 2, and ◯ and ♢ are separated at level 3. Their representatives are $\{\texttt{11}, \texttt{10}\}$, $\{\texttt{000}, \texttt{001}\}$, and $\{\texttt{011}, \texttt{010}\}$, respectively, so the MCL is $4 + 6 + 6 = 16$.

The motivation for using Gray code comes from Computable Analysis [31], a well-established mathematical framework for addressing analytical and computational aspects of real numbers in a fully computational manner through representation of real numbers as infinite sequences. Computability for real numbers depends on the embedding method used, and computation makes sense only if the method meets a key mathematical property: "admissibility" (see [31] for its mathematical definition and properties). It is thus natural that the clustering results depends on the embedding method. Gray code has been shown to be admissible [29] whereas binary embedding is not, and this property is a key for embedding that can detect arbitrarily shaped clusters.

This paper is organized as follows: Section 2 gives notation and Section 3 introduces the MCL. Section 4 gives a formal definition of clustering based on the MCL and explains the integration of COOL with the computation of the MCL. In Section 5, we introduce Gray code embedding and analyze G-COOL theoretically. Section 6 describes the experiments, presents the results, and discusses them. Section 7 summarized the key points with reviewing related work.

2 Notation

In the following, $\mathbb{R}^d$ denotes the d-dimensional Euclidean space. A *data point* x is a vector in $\mathbb{R}^d$, and a *dataset* X is a finite set of data points. For a pair of sets X and Y, $X \setminus Y$ means the relative complement of Y in X.

Clustering is the partition of a dataset X into K subsets $C_1, \ldots, C_K$, called *clusters*, where $C_i \neq \emptyset$, $C_i \cap C_j = \emptyset$ with $i \neq j$, and $\bigcup_{i \in \{1,\ldots,K\}} C_i = X$. Here we say that a set $\mathcal{C} = \{C_1, \ldots, C_K\}$ holding the above properties is a *partition* of X and denote the

set of all possible partitions by $\mathcal{C}(X)$; *i.e.*, $\mathcal{C}(X) = \{\mathcal{C} \mid \mathcal{C} \text{ is a partition of } X\}$. For a cluster C, $\#C$ denotes the number of data points in C.

The set of finite and infinite sequences over an alphabet Σ is denoted by Σ^* and Σ^ω, respectively. The length $|w|$ of a finite or an infinite sequence w is the number of positions for symbols other than $\perp$ (the undefinedness character) in w. For example, if $w = 11\perp100\perp\perp\perp\ldots$, $|w| = 5$. For a set of sequences W, the size of W is defined by $|W| := \sum_{w \in W} |w|$.

An *embedding* of $\mathbb{R}^d$ is an injective function γ from $\mathbb{R}^d$ to Σ^ω. For a pair of infinite sequences p, q, we write $p \leqslant q$ if $p(i) = q(i)$ for all i with $p(i) \neq \perp$, where $p(i)$ denotes the ith position (including 0) of p. This means that q is more specific than p since $\perp$ denotes undefinedness. Moreover, if $w\perp^\omega \leqslant p$ for $w \in \Sigma^*$, we write $w \sqsubset p$ (w is a prefix of p). We define $\uparrow w := \{p \in \text{range}(\gamma) \mid w \sqsubset p\}$ for $w \in \Sigma^*$, and $\uparrow W := \{p \in \text{range}(\gamma) \mid w \sqsubset p \text{ for some } w \in W\}$ for $W \subseteq \Sigma^*$.

3 Minimum Code Length

The minimum code length, or MCL, is used to measure partitions under a fixed embedding γ. We define, for $p \in \text{range}(\gamma)$ and $P \subset \text{range}(\gamma)$,

$$\Phi(p \,|\, P) := \left\{ w \in \Sigma^* \;\middle|\; \begin{array}{l} p \in \uparrow w, \text{ and } P \cap \uparrow v = \emptyset \\ \text{for all } v \text{ with } |v| = |w| \text{ and } p \in \uparrow v \end{array} \right\}.$$

Every element in $\Phi(p \,|\, P)$ is a prefix of p that discriminates p from P. Trivially, $\Phi(p \,|\, P) = \emptyset$ if $p \in P$.

The MCL is introduced here in accordance with the above preparations.

Definition 1 (MCL). Given an embedding γ, for a partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ of a dataset X, we define

$$\text{MCL}(\mathcal{C}) := \sum_{i \in \{1,\ldots,K\}} L_i(\mathcal{C}),$$

where

$$L_i(\mathcal{C}) := \min \left\{ |W| \;\middle|\; \begin{array}{l} \gamma(C_i) \subseteq \uparrow W \text{ and} \\ W \subseteq \bigcup\limits_{x \in C_i} \Phi\left(\gamma(x) \mid \gamma(X \setminus C_i)\right) \end{array} \right\}.$$

Intuitively, this gives the code length of the *maximumly compressed representatives* of a given partition through discretization using fixed embedding γ since the following property holds: For a partition $\mathcal{C}$ of X, if we discretize each data point $x \in X$ into a finite sequence $c(x)$ with γ (*i.e.*, $c(x) \sqsubset \gamma(x)$) such that $\left|\bigcup_{x \in X} c(x)\right| < \text{MCL}(\mathcal{C})$, then there must exist a pair of data points $x, y \in X$ satisfying $\uparrow c(x) \cap \uparrow c(y) \neq \emptyset$ and $x \in C_i, y \in C_j$ with $i \neq j$. Therefore, we cannot discriminate x from y and thus cannot find the partition $\mathcal{C}$ from compressed codes $c(X)$.

Example 1. Suppose we use binary embedding γ_{B}. Assume a one-dimensional dataset $X = \{0.1, 0.2, 0.8, 0.9\}$ and partitions $\mathcal{C}_1 = \{\{0.1, 0.2\}, \{0, 8, 0.9\}\}$ and $\mathcal{C}_2 = \{\{0.1\}, \{0.2, 0.8\}, \{0.9\}\}$. Then, $\mathrm{MCL}(\mathcal{C}_1) = L_1(\mathcal{C}_1) + L_2(\mathcal{C}_1) = 1 + 1 = 2$ since $\gamma_{\mathrm{B}}([0, 0.5]) = \uparrow\mathtt{0}$ and $\gamma_{\mathrm{B}}((0.5,\ 1]) = \uparrow\mathtt{1}$, and $\mathrm{MCL}(\mathcal{C}_2) = L_1(\mathcal{C}_2) + L_2(\mathcal{C}_2) + L_3(\mathcal{C}_2) = 3 + (3 + 3) + 3 = 12$ because we have $\gamma_{\mathrm{B}}([0, 0.125]) = \uparrow\mathtt{000}$, $\gamma_{\mathrm{B}}((0.125, 0.25]) = \uparrow\mathtt{001}$, $\gamma_{\mathrm{B}}((0.75, 0.875]) = \uparrow\mathtt{110}$, and $\gamma_{\mathrm{B}}((0.875, 1]) = \uparrow\mathtt{111}$. Note that $\gamma_{\mathrm{B}}([0, 0.25]) = \uparrow\mathtt{00}$, hence 0.1 and 0.2 cannot be discriminated using code $\mathtt{00}$, and that $\gamma_{\mathrm{B}}((0.75, 1]) = \uparrow\mathtt{11}$, hence 0.8 and 0.9 cannot be discriminated using $\mathtt{11}$.

The MCL is calculated for $O(nd)$ time complexity by using a radix sort, where n is the size of X (*i.e.*, $n = \#X$), and d is the dimension of X. This is why if the discretized dataset $\{p(0)p(1)\dots p(k-1) \mid p \in \gamma(X)\}$ at level k is sorted in advance, each data point simply needs to be compared with the subsequent one for each dimension, and the MCL is obtained by checking from $k = 1, 2, 3, \dots$.

4 Minimizing MCL and Clustering

We formulate clustering using the MCL as a criterion and describe clustering algorithm COOL, which always finds the globally optimal partition that *minimizes* the MCL.

4.1 Problem Formulation

The clustering problem with the MCL is defined as follows.

Definition 2. *Clustering of a dataset X under the MCL criterion* means finding the globally optimal partition that minimizes the MCL with more than K clusters; that is, finding $\mathcal{C}_{\mathrm{op}}$ such that

$$\mathcal{C}_{\mathrm{op}} \in \operatorname*{argmin}_{\mathcal{C} \in \mathcal{C}(X)_{\geqslant K}} \mathrm{MCL}(\mathcal{C}),$$

where $\mathcal{C}(X)_{\geqslant K} = \{\mathcal{C} \in \mathcal{C}(X) \mid \#\mathcal{C} \geqslant K\}$.

In this framework, we assume that a lower bound on the number of clusters K is given to avoid overgeneralization since, if we search for the optimal partition $\mathcal{C}_{\mathrm{op}}$ in $\mathcal{C}(X)$ (*i.e.*, all possible partitions) instead of $\mathcal{C}(X)_{\geqslant K}$, we always have the nonsense result $\mathcal{C}_{\mathrm{op}} = \{X\}$.

4.2 COOL Algorithm

Our COOL algorithm efficiently solves the optimization problem (Definition 2) by integrating the computation of MCL within the clustering step. By contrast, naïve approach that would compare the MCLs of all possible partitions would result in an algorithm with exponential time complexity. The pseudo-code of COOL is shown in Algorithm 1.

COOL is a *level-wise* clustering algorithm that finds the optimal partition $\mathcal{C}_{\mathrm{op}}$ by enumerating level-k partitions ($k = 1, 2, 3, \dots$).

Algorithm 1. COOL algorithm

Input: Dataset X, lower bound on number of clusters K, and noise parameter N
Output: Optimal partition $\mathcal{C}_{\mathrm{op}}$ and noise data

Function COOL(X, K, N)
1: Find partitions $\mathcal{C}^1_{\geqslant N}, \ldots, \mathcal{C}^m_{\geqslant N}$ such that $\#\mathcal{C}^{m-1}_{\geqslant N} < K \leqslant \#\mathcal{C}^m_{\geqslant N}$
2: $(\mathcal{C}_{\mathrm{op}}, \mathrm{MCL}) \leftarrow$ FINDCLUSTERS(X, K, $\{\mathcal{C}^1_{\geqslant N}, \ldots, \mathcal{C}^m_{\geqslant N}\}$)
3: **return** $(\mathcal{C}_{\mathrm{op}}, X \setminus \bigcup \mathcal{C}_{\mathrm{op}})$

Function FINDCLUSTERS(X, K, $\{\mathcal{C}^l, \ldots, \mathcal{C}^m\}$)
1: **if** $K = 1$ **then**
2: **return** $(\mathcal{C}^l, |W|)$, where $\gamma(\bigcup \mathcal{C}^l) \subseteq \uparrow W$ and $|w| = l$ for all $w \in W$
3: **end if**
4: Find k such that $\#\mathcal{C}^{k-1} < K \leqslant \#\mathcal{C}^k$
5: $\mathcal{C}_{\mathrm{op}} \leftarrow \mathcal{C}^k$
6: MCL $\leftarrow$ MCL($\mathcal{C}^k$)
7: **for each** C in $\mathcal{C}^l \cup \cdots \cup \mathcal{C}^k$
8: $L \leftarrow \min\{|W| \mid \gamma(C) \subseteq \uparrow W$ and $|w| = j$ for all $w \in W\}$, where $C \in \mathcal{C}^j$
9: $(\mathcal{C}, L') \leftarrow$ FINDCLUSTERS($X \setminus C$, $K-1$, $\{\mathcal{C}^j, \ldots, \mathcal{C}^k\}$)
10: **if** $L + L' <$ MCL **then**
11: $\mathcal{C}_{\mathrm{op}} \leftarrow C \cup \mathcal{C}$
12: MCL $\leftarrow L + L'$
13: **end if**
14: **end for**
15: **return** $(\mathcal{C}_{\mathrm{op}}, \mathrm{MCL})$

Definition 3 (Level-k partition). For a dataset X and an embedding γ, the *level-k partition* $\mathcal{C}^k$ is defined as follows: Every pair of data points $x, y \in X$ are contained in the same cluster if and only if $v = w$ for some $v \sqsubset \gamma(x)$ and $w \sqsubset \gamma(y)$ with $|v| = |w| = k$.

This means that if $x, y \in X$ are in the same cluster, there exists a *chain* of data points $z_1, z_2, \ldots, z_m$ ($m \geqslant 2$) such that, for all $i \in \{1, 2, \ldots, m-1\}$, $z_1 = x$, $z_m = y$, and $w_i = w_{i+1}$ for some $w_i \sqsubset \gamma(z_i)$ and $w_{i+1} \sqsubset \gamma(z_{i+1})$ with $|w_i| = |w_{i+1}| = k$. Obviously, the level-k partition is determined uniquely. The time complexity of finding the level-k partition is $O(nd)$, where n and d are the size and dimension of the dataset, respectively, since, if the discretized dataset $\{p(0)p(1)\ldots p(k-1) \mid p \in \gamma(X)\}$ at level k is initially sorted using a radix sort, clusters are constructed by comparing each data point to the next data point for each dimension.

The most important feature of the level-k partition is that the optimal partition $\mathcal{C}_{\mathrm{op}}$ in Definition 2 is obtained by searching for only clusters contained in the level-k partition.

Lemma 1. *For every cluster $C \in \mathcal{C}_{\mathrm{op}}$, C is contained in some level-k partition, that is, $C \in \mathcal{C}^k$ for some $k \in \mathbb{N}$.*

Proof. Let $\mathcal{C}$ be a partition such that, for every $C \in \mathcal{C}$, $C \in \mathcal{C}^k$ for some k, and a pair of clusters $C, C' \in \mathcal{C}$ is fixed. Then, from the definition of the level-k partition, the

following condition holds: For all pairs of clusters D, D' such that $D \cup D' = C \cup C'$ and $D \cap D' = \emptyset$, we have $\mathrm{MCL}(\mathcal{C}) \leqslant \mathrm{MCL}(\mathcal{C}')$, where $\mathcal{C}' = (\mathcal{C} \setminus \{C, C'\}) \cup \{D, D'\}$. Therefore, for the optimal partition $\mathcal{C}_{\mathrm{op}}$, $C \in \mathcal{C}^k$ with $k \in \mathbb{N}$ for all $C \in \mathcal{C}_{\mathrm{op}}$. □

The level-k partition has a *hierarchical* structure: For each cluster $C \in \mathcal{C}^k$, there must exist a set of clusters $\mathcal{D} \subseteq \mathcal{C}^{k+1}$ such that $\bigcup \mathcal{D} = C$. Thus, COOL works through divisive hierarchical clustering. The MCL of the level-k partition used in line 6 of the function FindClusters in Algorithm 1 can thus be easily calculated: Let $\mathcal{C}^k$ be a set of clusters $\{C_1, \ldots, C_K\}$. For each C_i and for the minimum level l such that $C_i \in \mathcal{C}^l$,

$$L_i(\mathcal{C}^k) = \min\{|W| \mid \gamma(C_i) \subseteq \uparrow W \text{ and } |w| = l \text{ for all } w \in W\}$$

holds. This means that we can obtain the MCL of the level-k partition by checking only sequences with length l.

Next we show that COOL can solve the optimization problem in Definition 2.

Proposition 1. *The* COOL *algorithm* (*Algorithm 1*) *always outputs the globally optimal partition* $\mathcal{C}_{\mathrm{op}}$.

Proof. Let $\#\mathcal{C}_{\mathrm{op}} = K$. Then there must exist $k \in \mathbb{N}$ such that $K \leqslant \#\mathcal{C}^k$ and $\#\mathcal{C}^{k'} < K$ for all $k' < k$ since the number of clusters in the level-k partition $\#\mathcal{C}^k$ increases monotonically with respect to increase of k. Fix a cluster $C \in \mathcal{C}_{\mathrm{op}}$, and let $\mathcal{C}'_{\mathrm{op}}$ be the optimal partition for the dataset $X \setminus C$. Then we can easily check that $\mathcal{C}'_{\mathrm{op}} \cup \{C\}$ coincides with $\mathcal{C}_{\mathrm{op}}$. Moreover, from Lemma 1 and the definition of the level-k partition, for all $C \in \mathcal{C}_{\mathrm{op}}$, $C \in \mathcal{C}^l$ for some $l \in \{1, \ldots, k\}$. Thus, COOL finds the optimal partition $\mathcal{C}_{\mathrm{op}}$ by recursive computing in Algorithm 1 (lines 4 - 7) with fixing each cluster in $\mathcal{C}^1 \cup \cdots \cup \mathcal{C}^k$. □

COOL can find the globally optimal partition $\mathcal{C}_{\mathrm{op}}$ efficiently, and its time complexity is $O(nd)$ and $O(nd + K!)$ in the best and worst cases, respectively, since finding m partitions in the first line of the function Cool takes $O(nd)$, and the function FindClusters takes $O(K!)$ in the worst case. Usually, $K \ll n$ holds, so complexity becomes $O(nd)$.

Furthermore, noise is directly removed by COOL using a lower bound on the size of each cluster N, which we call the *noise parameter*. For a partition $\mathcal{C}$, we denote the set $\{C \in \mathcal{C} \mid \#C \geqslant N\}$ by $\mathcal{C}_{\geqslant N}$. For example, let a dataset $X = \{0.1, 0.4, 0.5, 0.6, 0.9\}$ and $\mathcal{C} = \{\{0.1\}, \{0.4, 0.5, 0.6\}, \{0.9\}\}$. Then, $\mathcal{C}_{\geqslant 2} = \{\{0.4, 0.5, 0.6\}\}$, and two data points, 0.1 and 0.9, are detected as noise.

5 G-COOL: COOL with Gray Code

We use Gray code embedding for COOL, and show its powerful clustering ability by theoretical analysis. We call COOL with Gray code embedding G-COOL.

5.1 Gray Code Embedding

Gray code embedding is illustrated in Figure 2. Its rich mathematical properties are described elsewhere [29]. Gray code was originally simply binary encoding of natural

numbers, as mentioned in introduction. For example, natural numbers $1, 2, \ldots, 8$ are represented in Gray code as 000, 001, 011, 010, 110, 111, 101, 100, whereas, in binary code, they are represented as 000, 001, 010, 011, 100, 101, 110, 111. The importance of Gray code is that only one bit differs between one code and its successor, that is, the Hamming distance between them is always one. Here $\mathcal{I}$ denotes the unit interval $[0,1] \times \cdots \times [0,1] \subset \mathbb{R}^d$, and $\Sigma^{\omega}_{\perp,d}$ denotes the set of infinite sequences for which, in each sequence, at most d positions are $\perp$. For example, if $\Sigma = \{0, 1\}$ and $d = 2$, then $0\perp 100\cdots \in \Sigma^{\omega}_{\perp,d}$, $\perp\perp 110\cdots \in \Sigma^{\omega}_{\perp,d}$, and $0\perp 1\perp\perp 0\ldots \notin \Sigma^{\omega}_{\perp,d}$. In the following, we consider only real vectors in $\mathcal{I}$.

Definition 4 (Gray code embedding). (*One-dimensional*) *Gray code embedding* is an injective function, $\gamma_{\mathrm{G}} : \mathcal{I} \to \Sigma^{\omega}_{\perp,d}$ $(d = 1)$, that maps $x \in \mathcal{I}$ to an infinite sequence $p(0)p(1)p(2)\ldots$: For each i, $p(i) := 1$ if

$$2^{-i}m - 2^{-(i+1)} < x < 2^{-i}m + 2^{-(i+1)}$$

holds for an odd number m, $p(i) := 0$ if the same holds for an even number m, and $p(i) := \perp$ if $x = 2^{-i}m - 2^{-(i+1)}$ for some integer m.

Moreover, by using the *wrapping function*

$$\varphi(p_1, \ldots, p_d) := p_1(0)\ldots p_d(0)p_1(1)\ldots p_d(1)p_1(2)\ldots p_d(2)\ldots,$$

we can define *d-dimensional Gray code embedding* $\gamma^d_{\mathrm{G}} : \mathcal{I} \to \Sigma^{\omega}_{\perp,d}$ as

$$\gamma^d_{\mathrm{G}}(x_1, \ldots, x_d) := \varphi(\gamma_{\mathrm{G}}(x_1), \ldots, \gamma_{\mathrm{G}}(x_d)).$$

We abbreviate d of γ^d_{G} if it is understood from the context.

Example 2. For one-dimensional data points $x = 0.2$, $y = 0.5$, and $z = 0.7$, we have $\gamma_{\mathrm{G}}(x) = 0010\ldots$, $\gamma_{\mathrm{G}}(y) = \perp 100\ldots$, and $\gamma_{\mathrm{G}}(z) = 1110\ldots$ with Gray code embedding, while $\gamma_{\mathrm{B}}(x) = 0001\ldots$, $\gamma_{\mathrm{B}}(y) = 0111\ldots$, and $\gamma_{\mathrm{B}}(z) = 1011\ldots$ with binary embedding. For a two-dimensional data point (x, y), we have $\gamma_{\mathrm{G}}(x, y) = 0\perp 011000\ldots$, and for a three-dimensional data point (x, y, z), $\gamma_{\mathrm{G}}(x, y, z) = 0\perp 1011101000\ldots$ with Gray code embedding.

5.2 Theoretical Analysis of G-COOL

Here we show that G-COOL achieves internal cohesion and external isolation without any distance calculation or data distribution. In the following, we measure the distance between $x, y \in \mathbb{R}^d$ by using the L_∞ metric, where the distance is defined by

$$d_\infty(x, y) := \max_{i \in \{1,\ldots,d\}} |x_i - y_i| .$$

Lemma 2. *For the level-k partition $\mathcal{C}^k$ of a dataset X with Gray code embedding γ_{G}, two data points $x, y \in X$ are in the same cluster if $d_\infty(x, y) < 2^{-(k+1)}$ and are not in the same cluster only if $d_\infty(x, y) \geqslant 2^{-(k+1)}$.*

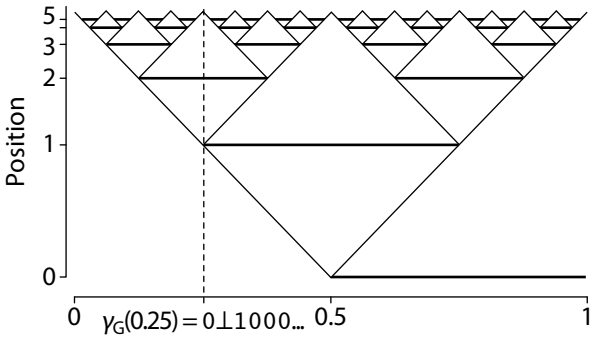

Fig. 2. Gray code embedding γ_{G}. Position i is `1` if it is on the line, $\perp$ if on the end point, and `0` otherwise. Diagonal lines are auxiliary lines. For example, if $p = \gamma_{\mathrm{G}}(0.25)$, $p = \texttt{0}\perp\texttt{1000}\ldots$ because position 0 is not on the line, 1 is on the end point, 2 is on the line, and every $i \geqslant 3$ is not on the line.

Proof. From the definition of Gray code embedding, if $d_\infty(x, y) < 2^{-(k+1)}$ for $x, y \in X$, there must exist a finite sequence w with $|w| = k$ such that $w \sqsubset \gamma_{\mathrm{G}}(x)$ and $w \sqsubset \gamma_{\mathrm{G}}(y)$. Thus, x and y are in the same cluster in the level-k partition $\mathcal{C}^k$. Moreover, this means that, if x and y are in the different clusters in $\mathcal{C}^k$, $d_\infty(x, y) \geqslant 2^{-(k+1)}$. □

Informally, the *redundancy* of Gray code embedding enables the powerful property described in the above lemma, that is, for an infinite sequence $p = \gamma_{\mathrm{G}}(x)$, there may be two prefixes, $v_1 \sqsubset p$ and $v_2 \sqsubset p$ with $|v| = |w|$.

Example 3. Let us consider the situation illustrated in Figure 3, where we have five one-dimensional data points: $x_{\mathrm{a}} = 0.14$, $x_{\mathrm{b}} = 0.48$, $x_{\mathrm{c}} = 0.51$, $x_{\mathrm{d}} = 0.73$, and $x_{\mathrm{e}} = 0.77$. In binary embedding, the unit interval $\mathcal{I} = [0, 1]$ is divided into two intervals $[0, 0.5]$ and $[0.5, 1]$ at level-1, while it is divided into three intervals $[0, 0.5]$, $[0.25, 0.75]$, and $[0.5, 1]$ in Gray code embedding. Thus, there are three overlapping clusters $\{x_{\mathrm{a}}, x_{\mathrm{b}}\}$ (encoded as `0`), $\{x_{\mathrm{b}}, x_{\mathrm{c}}, x_{\mathrm{d}}\}$ (encoded as $\perp$`1`), and $\{x_{\mathrm{c}}, x_{\mathrm{d}}, x_{\mathrm{e}}\}$ (encoded as `1`). Actually, there is only one cluster $\{x_{\mathrm{a}}, x_{\mathrm{b}}, x_{\mathrm{c}}, x_{\mathrm{d}}, x_{\mathrm{e}}\}$ since they are merged. At level-2, we have four clusters with binary embedding although some data points such as x_{b} and x_{c} are close. On the other hand, we have two *natural* clusters $\{x_{\mathrm{a}}\}$ and $\{x_{\mathrm{b}}, x_{\mathrm{c}}, x_{\mathrm{d}}, x_{\mathrm{e}}\}$ with Gray code embedding.

Intuitively, this lemma theoretically supports the claim that G-COOL finds *natural* clusters. For a data point $x \in X$, we say that a data point $y \in X$ is the *nearest neighbor* of x if $y \in \mathrm{argmin}_{x' \in X} d_\infty(x, x')$.

Theorem 1 (main theorem). *The optimal partition $\mathcal{C}_{\mathrm{op}}$ of a dataset X generated by* G-COOL *has the following property: For every data point $x \in C$ with $C \in \mathcal{C}_{\mathrm{op}}$ and $\#C \geqslant 2$, its nearest neighbor $y \in C$.*

Proof. From Lemma 1, every cluster $C \in \mathcal{C}_{\mathrm{op}}$ is contained in $\mathcal{C}^k$ for some k. Thus, from Lemma 2, trivially, any $x \in C$ with $C \in \mathcal{C}_{\mathrm{op}}$, $\#C = 1$ or its nearest neighbor $y \in C$. □

id	Value	Level 1		Level 2	
		Binary	Gray	Binary	Gray
a	0.14	0	0	00	00
b	0.48	0	0, ⊥1	01	01, ⊥10
c	0.51	1	1, ⊥1	10	11, ⊥10
d	0.73	1	1, ⊥1	10	11, 1⊥1
e	0.77	1	1	11	10, 1⊥1

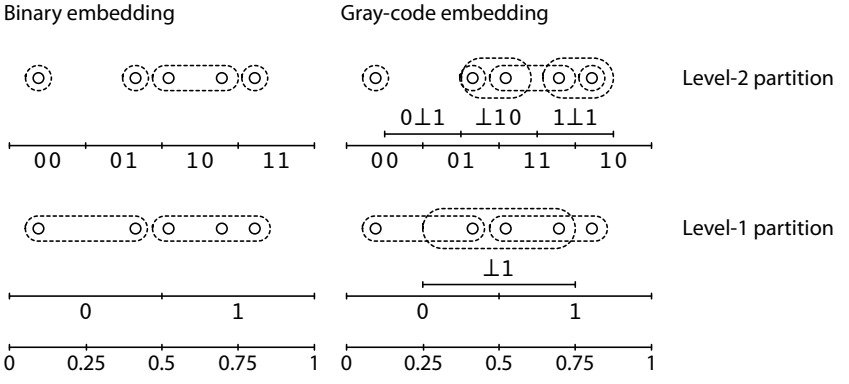

Fig. 3. Examples of level-1 and 2 partitions with binary and Gray code embedding

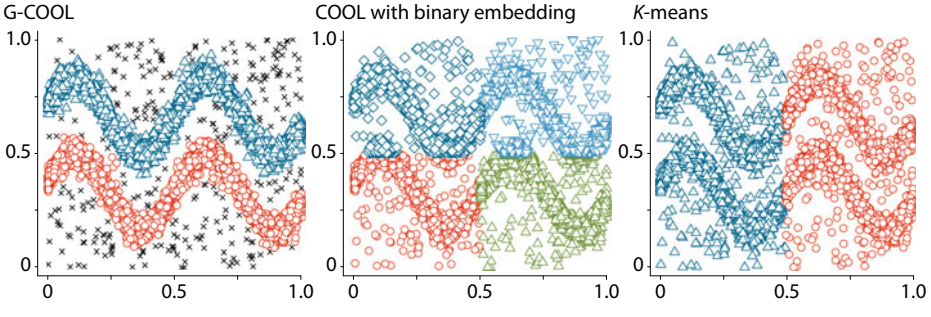

Fig. 4. Clustering results for G-COOL and COOL with binary embedding ($K = 2, N = 50$) and K-means ($K = 2$). Dataset size is 10,500 (where 500 points are noise). G-COOL detects two natural clusters and noise. The other two methods cannot find such clusters.

This property of Gray code (Lemma 2) enables clusters with the condition in Theorem 1 to be quickly found, whereas the naïve solution results in more than $O(n^2)$. Figure 4 illustrates the results of G-COOL for a two-dimensional dataset for which K-means could not find natural clusters. We can see that COOL with binary embedding also failed to find such clusters.

6 Experiments

We analyze G-COOL empirically and evaluate the effectiveness of G-COOL and the proposed measure, MCL. We use low-dimensional synthetic and real datasets, which are common in spatial clustering setting.

6.1 Methods

Environment. G-COOL was implemented in R version 2.12.2 [24], and all experiments were performed in R. We used Mac OS X version 10.6.5 with two 2.26-GHz Quad-Core Intel Xeon CPUs and 12 GB of memory.

Datasets. The synthetic datasets were used to evaluate robustness against the number of clusters, the size of the datasets, and noise existence. They were randomly generated using the R `clusterGeneration` package [23], and the parameters were set as follows: $sepVal = 0.34$, $numNonNoisy = 2$, $numNoisy = 1$, $numOutlier = 500$, and $rangeN = c(1000, 2000)$. We generated 20 datasets to obtain the mean and s.e.m. (standard error of the mean). The size of each cluster was around 1,500, so the size of the datasets varied from ∼3,000 to ∼10,500. Each dataset was three-dimensional, where one dimension was composed of noise and about 500 data point were added to each dimension as noise.

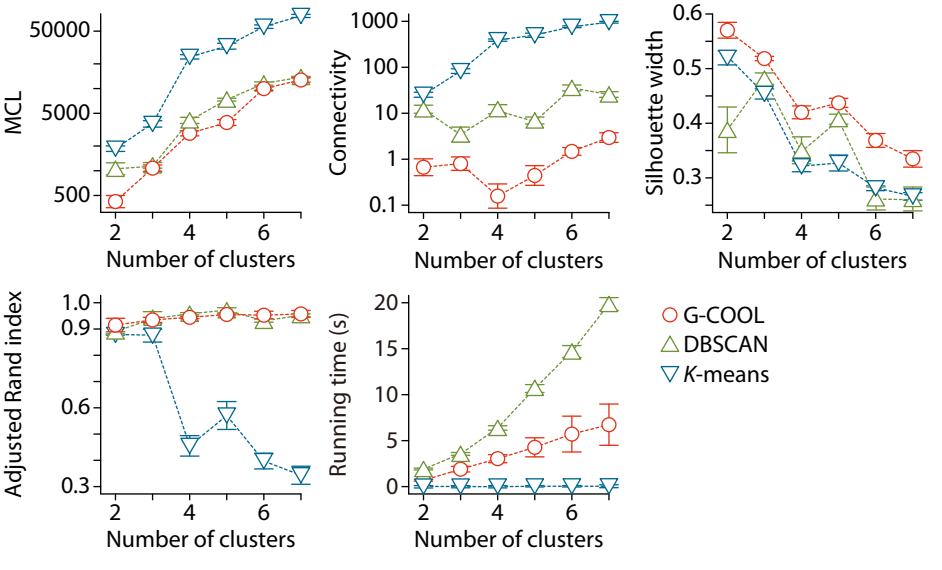

Fig. 5. Experimental results for synthetic datasets. MCL and connectivity should be minimized, and Silhouette width and adjusted Rand index should be maximized. Data show mean ± s.e.m.

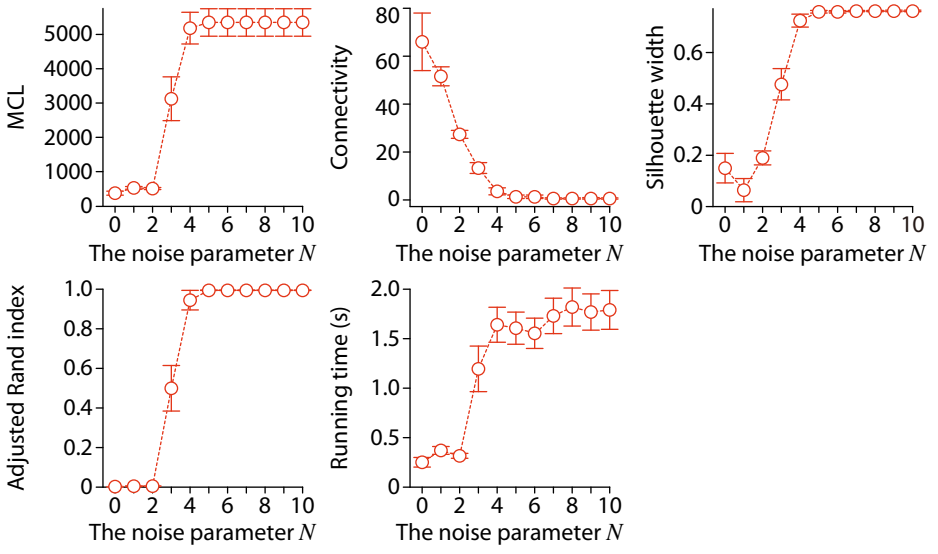

Fig. 6. Clustering speed and quality for G-COOL and synthetic datasets. MCL and connectivity should be minimized, and Silhouette width and adjusted Rand index should be maximized. G-COOL shows robust performance if the noise parameter N is large enough. Data show mean $\pm$ s.e.m.

Five real datasets were collected from the Earth-as-Art website[1], which contains geospatial satellite images (see Table 1 and Figure 7). Similar datasets were used in experiments with the state-of-the-art spatial clustering method [5]. Each image was pre-processed using ImageJ software [25]; they were reduced to 200×200 pixels and translated into binary images.

With G-COOL, each dataset was translated using min-max normalization [10] so that the dataset was in the unit interval $\mathcal{I}$, where each value x of ith dimension X_i of a dataset X was mapped to $x' = (x - \min X_i)/(\max X_i - \min X_i)$ and the runtime for the translation was included in the G-COOL running time.

Control Methods. As control methods, we used K-means and DBSCAN because K-means is the standard clustering algorithm and DBSCAN is a typical method for finding arbitrarily shaped clusters, and their source codes are publicly available. DBSCAN was executed using the R `fpc` package. We tuned the parameters of DBSCAN to obtain the best results.

Evaluation. With the synthetic datasets, performance was evaluated using internal and external measures. As internal measures, we used the MCL (with Gray code), the connectivity (takes values in $[0, \infty]$, to be minimized) [11], and the Silhouette width (takes values in $[-1, 1]$, to be maximized) [26]. As an external measure, we used the adjusted Rand index (takes values in $[-1, 1]$, to be maximized) [13], which takes into account the

[1] `http://eros.usgs.gov/imagegallery/`

Table 1. Running time (in seconds) and MCL for real datasets. In table, n and K denote the number of data points and clusters, respectively. Clustering results are shown in Figure 7.

Name	n	K	Running time (s)		MCL	
			G-COOL	K-means	G-COOL	K-means
Delta	20748	4	1.158	0.012	4010	4922
Dragon	29826	2	0.595	0.026	3906	7166
Europe	17380	6	2.404	0.041	2320	12210
Norway	22771	5	0.746	0.026	1820	6114
Ganges	18019	6	0.595	0.026	2320	12526

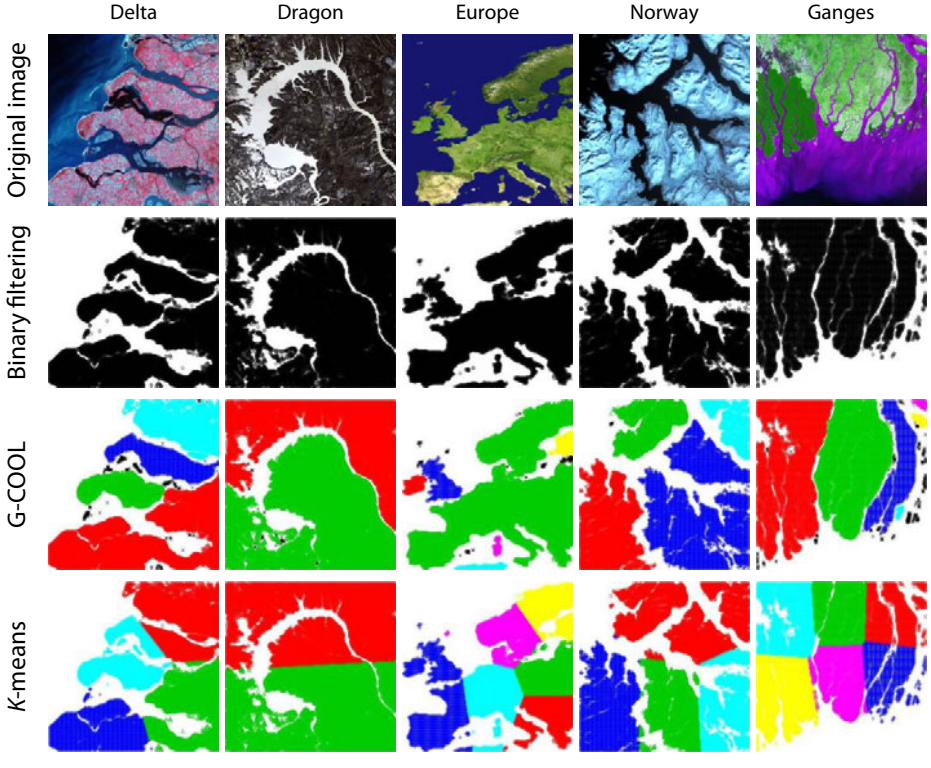

Fig. 7. Results for real datasets obtained from satellite images by G-COOL and K-means. G-COOL finds all natural clusters whereas K-means does not.

ground truth and is popular in the clustering literature. The measures were calculated using the R `clValid` [2], `cluster` [22], and `clues` [3] packages, respectively. For the real datasets, we used the MCL and simply show scatter plots of the results since we had no information on the ground truth.

6.2 Results and Discussion

The results obtained with the synthetic datasets (Figure 5) show that the quality of clusters obtained with G-COOL is significantly higher for three of the four quality measures (determined by paired t-test) and is competitive for the other one (adjusted Rand index). Moreover, it is faster than DBSCAN. These results show that the MCL works reasonably well as a measure of cluster quality compared to existing ones.

Note that we need to input only the lower bounds for the number and size of the clusters in G-COOL, whereas we have to tune the parameters carefully in DBSCAN and other shape-based (spatial) clustering algorithms. Therefore, G-COOL is more efficient and effective than existing clustering algorithms. Moreover, as shown in Figure 6, cluster quality is stable with respect to the noise parameter N (*i.e.*, lower bound on cluster size) even if the dataset contains noise, when N is large enough. If N is too small, then each noise is detected as a cluster. Thus, when clustering using G-COOL, all we have to do is set the parameter large enough, meaning that G-COOL is equally useful as K-means.

For all the real datasets, G-COOL finds natural clusters (N was set as 50 for all the datasets), as shown in Figure 7, whereas K-means results in inferior clustering quality (we did not perform DBSCAN since it takes too much time and needs manual tuning of the input parameters). Moreover, MCL of clustering results for G-COOL is much smaller than those for K-means (Table 1). These results show that G-COOL is robust and that it can find arbitrarily shaped clusters without careful tuning of the input parameters.

7 Conclusion

We have proposed an internal measure, the minimum code length (MCL), to evaluate the results of clustering and presented a clustering algorithm COOL that always finds the globally optimal partition; *i.e.*, clusters that have the minimum MCL. Intuitively, COOL produces the maximally compressed clusters using a fixed encoding scheme and does not take optimization of encoding into account. Moreover, Gray code is used for the encoding, resulting in an algorithm called G-COOL. Theoretically and empirically, G-COOL has been shown to be noise tolerant and to have the ability to find arbitrarily shaped clusters efficiently. The result is an effective solution to two essential problems, how to measure the goodness of clustering results and how to find good clusters.

Many types of *shape-based clustering*, or *spatial clustering*, methods have been proposed for finding arbitrarily shaped clusters, including partitional algorithms [4,5], the mass-based clustering algorithm [28], density-based clustering algorithms (*e.g.*, DBSCAN [7] and DENCLUE [12]), agglomerative hierarchical clustering algorithms (*e.g.*, CURE [8], CHAMELEON [15]), and grid-based algorithms (*e.g.*, STING [30] and Wave Cluster [27]). However, most of them are not practical. Their clustering results are sensitive to the input parameters, which have to be tuned manually, so they work well only if all parameters are tuned appropriately by the user. As a result, these methods are not well suited for users who are not specialized in machine learning. Furthermore, most of them are not scalable: their time complexity is quadratic or cubic with respect to data size. Compared to these methods, G-COOL is robust to the input parameters

and always finds the globally optimal clusters under the MCL criterion. Moreover, G-COOL is usually faster than most of these methods since the time complexity is linear with respect to data size.

Many cluster validity methods have been proposed for quantitative evaluation of clustering results [11]. These measures are usually divided into two categories: internal (*e.g.*, connectivity and Silhouette width) and external (*e.g.*, F-measure and Rand index). The internal measures are intrinsic to actual clustering while the external measures need information that may not be available in an actual situation. Our proposed measure, MCL, can be categorized as an internal measure. Its effectiveness has been shown experimentally (see Section 6).

G-COOL's results are robust to changes in the input parameters, and does not assume a data distribution and does not need a distance calculation. Thus, it can be effectively applied to other machine learning tasks, such as anomaly detection. Theoretical analysis of relationship between admissibility of encoding schemes in computing real numbers and the ability of clustering to detect arbitrarily shaped clusters is necessary future work.

Acknowledgments. We would like to thank Marco Cuturi for his helpful comments. This work was partly supported by Grant-in-Aid for Scientific Research (A) 22240010 and for JSPS Fellows 22·5714.

References

1. Berkhin, P.: A survey of clustering data mining techniques. Grouping Multidimensional Data, 25–71 (2006)
2. Brock, G., Pihur, V., Datta, S., Datta, S.: clValid: An R package for cluster validation. Journal of Statistical Software 25(4), 1–22 (2008)
3. Chang, F., Qiu, W., Zamar, R.H., Lazarus, R., Wang, X.: clues: An R package for nonparametric clustering based on local shrinking. Journal of Statistical Software 33(4), 1–16 (2010), http://www.jstatsoft.org/v33/i04/
4. Chaoji, V., Hasan, M.A., Salem, S., Zaki, M.J.: SPARCL: An effective and efficient algorithm for mining arbitrary shape-based clusters. Knowledge and Information Systems 21(2), 201–229 (2009)
5. Chaoji, V., Li, G., Yildirim, H., Zaki, M.J.: ABACUS: Mining arbitrary shaped clusters from large datasets based on backbone identification. In: Proceedings of 2011 SIAM International Conference on Data Mining, pp. 295–306 (2011)
6. Cilibrasi, R., Vitányi, P.M.B.: Clustering by compression. IEEE Transactions on Information Theory 51(4), 1523–1545 (2005)
7. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, vol. 96, pp. 226–231 (1996)
8. Guha, S., Rastogi, R., Shim, K.: CURE: An efficient clustering algorithm for large databases. Information Systems 26(1), 35–58 (1998)
9. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of Intelligent Information Systems 17(2), 107–145 (2001)
10. Han, J., Kamber, M.: Data Mining, 2nd edn. Morgan Kaufmann, San Francisco (2006)
11. Handl, J., Knowles, J., Kell, D.B.: Computational cluster validation in post-genomic data analysis. Bioinformatics 21(15), 3201 (2005)

12. Hinneburg, A., Keim, D.A.: An efficient approach to clustering in large multimedia databases with noise. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining, pp. 58–65 (1998)
13. Hubert, L., Arabie, P.: Comparing partitions. Journal of Classification 2(1), 193–218 (1985)
14. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
15. Karypis, G., Eui-Hong, H., Kumar, V.: CHAMELEON: Hierarchical clustering using dynamic modeling. Computer 32(8), 68–75 (1999)
16. Keogh, E., Lonardi, S., Ratanamahatana, C., Wei, L., Lee, S.H., Handley, J.: Compression-based data mining of sequential data. Data Mining and Knowledge Discovery 14, 99–129 (2007)
17. Knuth, D.E.: The Art of Computer Programming. Fascicle 2: Generating All Tuples and Permutations, vol. 4. Addison-Wesley Professional, Reading (2005)
18. Kontkanen, P., Myllymäki, P.: An empirical comparison of NML clustering algorithms. In: Proceedings of Information Theory and Statistical Learning, pp. 125–131 (2008)
19. Kontkanen, P., Myllymäki, P., Buntine, W., Rissanen, J., Tirri, H.: An MDL framework for data clustering. In: Grünwald, P., Myung, I.J., Pitt, M. (eds.) Advances in Minimum Description Length: Theory and Applications. MIT Press, Cambridge (2005)
20. Li, M., Badger, J.H., Chen, X., Kwong, S., Kearney, P., Zhang, H.: An information-based sequence distance and its application to whole mitochondrial genome phylogeny. Bioinformatics 17(2), 149–154 (2001)
21. MacQueen, J.: Some methods for classification and analysis of multivariate observations. In: Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, pp. 281–297 (1967)
22. Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M.: Cluster analysis basics and extensions (2005)
23. Qiu, W., Joe, H.: Generation of random clusters with specified degree of separation. Journal of Classification 23, 315–334 (2006)
24. R Development Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2011), http://www.R-project.org
25. Rasband, W.S.: ImageJ. U. S. National Institutes of Health, Bethesda, Maryland, USA (1997–2011), http://imagej.nih.gov/ij/
26. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of Computational and Applied Mathematics 20, 53–65 (1987)
27. Sheikholeslami, G., Chatterjee, S., Zhang, A.: WaveCluster: A multi-resolution clustering approach for very large spatial databases. In: Proceedings of the 24th International Conference on Very Large Data Bases, pp. 428–439 (1998)
28. Ting, K.M., Wells, J.R.: Multi-dimensional mass estimation and mass-based clustering. In: Proceedings of 10th IEEE International Conference on Data Mining, pp. 511–520 (2010)
29. Tsuiki, H.: Real number computation through Gray code embedding. Theoretical Computer Science 284(2), 467–485 (2002)
30. Wang, W., Yang, J., Muntz, R.: STING: A statistical information grid approach to spatial data mining. In: Proceedings of the 23rd International Conference on Very Large Data Bases, pp. 186–195 (1997)
31. Weihrauch, K.: Computable Analysis: An Introduction. Springer, Heidelberg (2000)