A Simulated Annealing-Based Maximum Margin Clustering Algorithm

Sattar Seifollahi* Adil Bagirov† Ehsan Zare Borzeshi‡
Massimo Piccardi§

Abstract

Maximum-margin clustering (MMC) is an extension of the support vector machine (SVM) to clustering. It partitions a set of unlabelled data into multiple groups by finding hyperplanes with the largest margins. Although existing algorithms have shown promising results, there is no guarantee of convergence of these algorithms to global solutions due to the non-convexity of the optimization problem. In this paper, we propose a simulated annealing-based algorithm that is able to address the issue of local minima in the MMC problem. The novelty of our algorithm is twofold: (1) it comprises a comprehensive cluster modification scheme based on simulated annealing, and (2) it introduces a new approach based on the combination of k-means++ and SVM at each step of the annealing process. More precisely, k-means++ is initially applied to extract subsets of the data points. Then, an unsupervised SVM is applied to improve the clustering results. Experimental results on various benchmark datasets (of up to over a million points) give evidence that the proposed algorithm is more effective at solving the clustering problem than a number of popular clustering algorithms.

1 Introduction

Clustering deals with the problem of organizing a data set into clusters based on a definition of similarity. Most clustering algorithms are based on two main lines of approaches, the hierarchical and the partitional [12]. Algorithms based on the hierarchical approach generate a dendrogram representing the nested grouping of patterns and similarity levels at which groupings change. Partitional clustering algorithms find the partition that optimizes a clustering criterion. In this paper, we introduce a new partitional clustering algorithm.

The similarity measure is fundamental to formulate a clustering problem. This measure, in particular, can be defined using distance(-like) functions. Clustering problems

*Faculty of Engineering and Information Technology, University of Technology Sydney, and Capital Markets Cooperative Research Centre (CMCRC), Sydney, NSW, Australia; Email: sseif@cmcrc.com

†Faculty of Science and Technology, Federation University Australia, VIC, Australia; Email: a.bagirov@federation.edu.au

‡Capital Markets Cooperative Research Centre (CMCRC), Sydney, NSW, Australia, Email: ehsan.zareborzeshi@gmail.com

§Faculty of Engineering and Information Technology, University of Technology Sydney, Sydney, NSW, Australia; Email: Massimo.Piccardi@uts.edu.au

using the Euclidean norm as similarity are called the minimum sum-of-squares clustering (MSSC) problems. Algorithms for solving these problems include the k-means algorithm and its variations (see, for example, [13, 16] and references therein). The global k-means algorithm and its various modifications are among the most efficient algorithms for solving the MSSC problem. [2, 3, 5, 17, 21].

There is a growing interest in applying support vector machine (SVM) techniques to clustering, motivated by their success in supervised learning. However, unlike large-margin supervised learning, large-margin unsupervised learning is a non-convex problem. Most algorithms for large-margin unsupervised learning are based on relaxation techniques (e.g., [18, 28]). These techniques allow applying convex optimization methods: however, they are only applicable to relatively small datasets. Therefore, it is imperative to develop methods which are both efficient and accurate at solving large-margin unsupervised learning problems on large datasets. In particular, such methods can be designed using opportunistic combinations of local and global search algorithms. In these methods, local search algorithms are used to determine local solutions while global search algorithms are used to escape from such local solutions and find better re-starting points for the local search algorithms.

In this paper, we develop a new algorithm for solving large-margin unsupervised learning problems based on the combination of simulated annealing, the k-means++ clustering algorithm and the SVM. The use of simulated annealing allows us to deal with the non-convexity of the problem while k-means++ and SVM are applied to solve adapted clustering problems with the aim to achieve a constructive combination from these two complementary approaches. The choice of the simulated annealing method is based on the fact that this method has an efficient mechanism to escape from local solutions which are different from the global one. Our approach consists of two stages: first, we find a subset of the points called the *representative points*; then, we iteratively apply $k-$means++ and SVM using only this subset. In this way, we are able to significantly reduce the inherent computational complexity while retaining high accuracy. After each iteration, the representative points are updated and the approach iterated to find an accurate solution for the MSSC problem. The proposed algorithm is tested using datasets of small to large size (¿ 1,000,000 points) and the results compared with those obtained using de-facto standard algorithms such as k-means++ [1], *mini-batch* k-means [22], the Dirichlet process Gaussian mixture model and fuzzy c-means.

The rest of this paper is organized as follows. The formulation of the clustering problem is given in Section 2. Section 3 provides a brief review of maximum-margin clustering and some related works. The proposed method is presented in Section 4. Numerical results are reported and discussed in Section 5, and Section 6 contains some concluding remarks.

2 Formulation of the Clustering Problem

In this section we present nonsmooth optimization formulations of the clustering problem. Let us consider a finite set X of points in the n-dimensional space R^n:

$$X = \{x^1, \ldots, x^m\}, \text{where } \ x^i \in R^n, i = 1, \ldots, m.$$

The data points x^i, $i = 1, \ldots, m$ are called *instances* and each instance has n *dimensions.*

The *hard unconstrained clustering problem* is the distribution of the points of the set X into a given number k of disjoint subsets X^j, $j = 1, \ldots, k$ with respect to predefined criteria such that

1. $X^j \neq \emptyset$, $j = 1, \ldots, k$
2. $X^j \bigcap X^l = \emptyset$, for all $j, l = 1, \ldots, k$, $j \neq l$.
3. $X = \bigcup_{j=1}^{k} X^j$.

Sets $X^j, j = 1, \ldots, k$ are called *clusters*. Each cluster X^j can be identified by its *center* $c^j \in R^n, j = 1, \ldots, k$. The problem of finding these centers is called the *k-clustering* (or k-partition) *problem*.

In this paper, the similarity measure is defined using the Euclidean norm (the L_2-norm)

$$d_2(c, x) = \left(\sum_{i=1}^{n} (c_i - x_i)^2 \right)^{1/2}.$$

The *nonsmooth optimization formulation of the MSSC problem* is [4]:

$$\begin{cases} \min & f_k(c) \\ \text{subject to} & c = (c^1, \ldots, c^k) \in R^{nk}, \end{cases} \tag{1}$$

where

$$f_k(c^1, \ldots, c^k) = \sum_{x \in X} \min_{j=1,\ldots,k} d_2(c^j, x). \tag{2}$$

The function f_k is called the *k-th clustering objective function*. For $k = 1$ this function is convex and for $k > 1$ it is both non-convex and nonsmooth.

3 Maximum-Margin Clustering and Related Works

Consider a set of training samples, $D = \{(x^i, y^i)\}_{i=1}^{m}$, where x^i is an instance and y^i its class label. For simplicity, let us assume that $y^i \in \{-1, 1\}$. The SVM finds a maximum-margin hyperplane $h(x) = w^T \phi(x) + b = 0$, where $\phi(x)$ is the mapping induced by a kernel and T is the vector transpose, by solving:

$$\begin{cases} \min_{w,b,\xi_i} & \|w\|^2 + 2\mathcal{C}\xi^T e \\ \text{subject to} & y^i(w^T \phi(x^i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \end{cases} \tag{3}$$

Here, the ξ_i's are slack variables for the errors, $\mathcal{C} > 0$ is a regularization parameter and $e = (1, \ldots, 1)^T$. The optimization problem (3) is convex. If the class labels, y^i, are unknown, the problem becomes a non-convex maximum-margin clustering problem (MMC) [29]:

$$\begin{cases} \min_y \min_{w,b,\xi_i} & \|w\|^2 + 2\mathcal{C}\xi^T e \\ \text{subject to} & y^i(w^T \phi(x^i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0. \\ & y^i \in \{-1, +1\}, \\ & -l \leq e^T y \leq l. \end{cases} \tag{4}$$

The last constraint, $-l \leq e^T y \leq l$, is a "balance" constraint preventing the convergence of the algorithm to a trivially "optimal" solution where all instances are assigned to the same class.

The dual problem of (4) is:

$$\begin{cases} \min_y \max_\alpha & 2\alpha^T e - \alpha^T (K \circ yy^T)\alpha \\ \text{subject to} & \alpha^T y = 0, \quad Ce \geq \alpha \geq 0, \\ & y^i \in \{-1, +1\}, \\ & -l \leq e^T y \leq l. \end{cases} \tag{5}$$

where α is the dual parameter, $\alpha = [\alpha_1, \ldots, \alpha_m]^T$, K is the kernel function and "$\circ$" is the element-wise product between matrices.

It is shown in [27] that (5) is equivalent to an NP-hard convex integer program. Since its introduction, it has been extended in many ways, for instance, by choosing different loss functions (e.g., Zhang et al. [29], Gieseke et al. [8]), incorporating additional constraints such as pairwise links (Hu et al. [11]) and manifold smoothness (Wang et al., [25]), or adding a feature weighting mechanism (Zhao et al., [30]).

Xu et al. [27] and Valizadegan and Rong [24] have reformulated the original problem as a semi-definite programming (SDP) problem. Zhang et al. [29] have employed alternating optimizations - finding labels and optimizing a support vector regression (SVR). Li et al. [18] iteratively generate the most violated labels, and combine them via multiple kernel learning. Note that the above methods can only solve binary-cluster clustering problems. To handle the multi-cluster case, Xu and Schuurmans [28] have extended the SDP method of [27]. Zhao et al. [31] have proposed a cutting-plane method which uses the constrained convex-concave procedure to relax the non-convex constraint. Gopalan and Sankaranarayanan [9] have examined data projections to identify the maximum margin. More recently, Karnin et al. [14] have proposed a global solution of polynomial complexity; however, its soft-margin formulation differs from the conventional soft-margin SVM based on the hinge loss.

Overall, the existing methods for MMC can be categorized as either relaxation or alternating methods [29, 31]. For the relaxation methods, one can refer to the papers of [27], [18] as notable examples. These methods aim to find best approximating functions for the non-convex formalization of (5) and solve the relaxed problem instead, hoping that the solution will be close enough to the actual one. However, in general, the solution found is not bounded compared to the actual solution.

In contrast to the relaxation methods, alternating methods solve the original problem in two steps [29]. They first initialize the labels using a clustering technique such as k-means and then they solve a supervised problem, often using SVM. The procedure can be iterated in various ways until no further improvement is achieved. Our proposed method falls in this group since we use an iterative method leveraging a combination of $k-$means++ and SVM at each iteration. However, our method differs from existing methods in that we solve a sub-problem at each step rather than considering the whole problem at once.

4 The proposed method

We design a method to address two main issues of the MMC method: (1) its computational complexity and (2) the inherent risk of falling into local solutions. To address the first problem, we perform most of the computation over only a subset of the points, hereafter called *representative points*, instead of the whole data. To address the second problem, we use a scheme employing simulated annealing and a combination of k-means++ and SVM to find "deeper" local solutions. The details for the initial process, post-processing, and the steps of the algorithm are presented in the following sub-sections.

4.1 Initial clusters

We generate an initial set of clusters using k-means++. These clusters are meant to act as "coarse-scale" points in the final clustering algorithm. Their number has to be very high so as to not over-simplify the problem, yet significantly smaller than the number of the original points to be approachable by MMC-like algorithms. An adequate value for the number of such initial clusters is likely to depend on the number of points, the number of attributes and the closeness of the points to each other. Herewith, we choose it as a random value proportional to the number of points. The initialization procedure can be summarized as follows:

Algorithm 1 Computing the set $\mathcal{R}$ of representative points.

Step 1: Select a number $\delta \in (0, 1)$ and the final number of clusters k to be computed.
Step 2: Compute k clusters $X^1, \ldots, X^k$ using the $k-$means++ algorithm.
Step 3: For each cluster $X^j,\ j = 1, \ldots, k$ compute the weight $w_j = |X^j|/m$ and define the number of initial clusters as $k_j = \delta w_j m$.
Step 4: For each cluster $X^j,\ j = 1, \ldots, k$ compute k_j clusters using $k-$means++.
Step 5: Define the set $\mathcal{R}$ of representative points as the set of cluster centers computed in Step 4.

4.2 Post-processing of the initial clusters

The set $\mathcal{R}$ of representative points obtained using Algorithm 1 is used as the input for a post-processing phase to generate the final clusters. To this aim, the points are grouped using a combination of k-means++ and SVM, labelled as KMSVM. We use an iterative cluster modification scheme based on simulated annealing in which we update the representative points at each iteration using two steps: (1) perturbing a randomly-selected representative point to another point in the same cluster, and (2) splitting an initial cluster into two new clusters if it exhibits high bi-modality when applying a Gaussian Mixture Model on the corresponding points.

We exploit simulated annealing to find a "deeper" local solution for the clustering problem. Simulated annealing comprises two main iterations: the outer and inner iterations. In the outer iteration the temperature, T, which is analogous to the temperature in the physical process of annealing, is updated. To this aim, we take an arbitrary initial value T_0 for the temperature and a number $r \in (0, 1)$ and use the following schedule for the temperature update: $T_{i+1} = r \times T_i, i = 0, 1, 2, \ldots$. In the inner iteration the current state

is modified to generate a new solution based on a proposal step. If the proposal reduces the value of the objective function, the transformation to the new state is accepted. If it increases the value of the objective function, the transformation is accepted with an acceptance probability:

$$p = \min\left(1, \exp\left(\frac{-\Delta f}{T}\right)\right), \tag{6}$$

where $\Delta f = f^{new} - f^{old}$, f^{old} is the function value in the previous state and f^{new} is the function value based on the perturbed configuration. More precisely, a random number u from the uniform distribution $U[0, 1]$ is generated. If $p \geq u$, the perturbed configuration is accepted as a new solution; otherwise the inner iterations are repeated. For more details on the simulated annealing method see Kirkpatrick [15] and Seifollahi et al. [23].

4.3 The overall algorithm

Algorithm 2 describes the main steps of the overall algorithm. Step 1 provides the set of representative points as presented in Section 4.1. Step 2 applies a step of local search (KMSVM), consisting of a predetermined number of alternating steps of k-means and SVM. Step 3 splits certain clusters into two, conditional to a bimodality test that is explained hereafter. Step 4 perturbs the representative points as presented in Section 4.2. Eventually, Step 5 tests the overall termination conditions.

In Step 3, we leverage bi-modality information to decide whether the initial clusters are appropriate. The idea is to split a candidate initial cluster so that it substantially contains only one mode. The presence of two distinct modes in a cluster can be identified by fitting a Gaussian mixture model (GMM) with two components. After the model is fit and the two components identified, the Kullback-Leibler divergence (KLD) can be applied to measure the difference between their distributions. By referring to the two Gaussian components as P and Q, KLD can be expressed in closed-form as [7]:

$$\begin{aligned} D_{KLD}(P|Q) = \frac{1}{2}\Big(&tr(\Sigma_q^{-1}\Sigma_p) + \\ &(\mu_q - \mu_p)^T\Sigma_q^{-1}(\mu_q - \mu_p) - \\ &n + \ln\big(\frac{\det\Sigma_q}{\det\Sigma_p}\big)\Big) \end{aligned} \tag{7}$$

where μ_p and μ_q are the means of P and Q, Σ_p and Σ_q are their covariance matrices and n is the dimension of the vector space.

Algorithm 2 Simulated annealing-based maximum-margin clustering (SAMMC).

Step 1: (Initialization). Compute k_0 initial clusters using Algorithm 1, where $k_0 = \sum_{i=1}^{k} k_i$ and compute the set $\mathcal{R}$ of representative points. Choose the final number of clusters, $k < k_0$.

Step 2: (Computation of clusters). Apply KMSVM to the set $\mathcal{R}$ to find k clusters.

Step 3: (Splitting clusters based on GMM). Apply the GMM on each initial cluster to find two new clusters for each one. Select an initial cluster with the highest KLD value as a candidate. Split the candidate cluster into two new clusters using $k-$means++. Set $k_0 = k_0 + 1$ and update the set $\mathcal{R}$. Apply KMSVM on the set $\mathcal{R}$. Accept or reject the proposal based on (6).

Step 4: (Perturbation of representative points). Select a representative point at random and perturb it to another point (at random) within the corresponding initial cluster. Recompute clusters using KMSVM on new representative points, and accept or reject the proposal based on (6).

Step 5: (Termination of algorithm). Repeat steps 3-4 until termination conditions are met.

As shown in Algorithm 2, ratio (6) is used twice for accepting or rejecting a new proposal: the first time (at Step 3) in the bimodality test, and the second time (Step 4) for a random walk. More precisely, i) in Step 3 the cluster having the highest KLD value is conditionally split into two subject to test (6), while ii) in Step 4, the random perturbation of a representative point is accepted subject to test (6). In this way, every point in the cluster can potentially switch cluster. In both Steps 3 and 4, clusters with only one point are discarded as they will cause no change to the objective function.

4.4 Convergence of the proposed algorithm

The convergence of the proposed algorithm to the global solutions follows from the convergence of the simulated annealing algorithm. It is well-known that under some mild assumptions the simulated annealing method convergence to global solutions of continuous global optimization problems with probability 1 (see, [20], for details).

The proposed algorithm is the combination of a local search and the simulated annealing method. A local search algorithm is applied to find stationary points of problem (4), and the simulated annealing method is applied to escape such points and find points which are located in deeper "basins" of the objective function.

The objective function in problem (4) has a finite number of local minimizers. Since the simulated annealing method escapes such points with probability 1, we get that the proposed algorithm converges to the set of global minimizers of problem (4) with probability 1.

5 Numerical Results

We compare the performance of the proposed algorithm with a pool of algorithms widely adopted for clustering: 1) the $k-$means++ algorithm (in the implementation of the scikit-learn Python machine learning library) [1]; 2) *mini-batch* k-means++ (initialized with

$k-$means++) [22]; 3) a Dirichlet process Gaussian mixture model (DPGMM) algorithm, from the same library [10]; and 4) fuzzy c-means, from the fuzzy logic scikit Python machine learning library [26]. Mini-batch k-means++ [22] was proposed as an alternative to the k-means algorithm for clustering very large datasets. The advantage of this algorithm is a reduction of the computational load deriving from the use of sub-samples of fixed size. DPGMM is an infinite mixture model with a Dirichlet Process as a prior distribution over the number of clusters. Fuzzy c-means [26] is a clustering algorithm in which each data point is assigned to multiple clusters in membership grades. For hardening the final assignment, we assign the point to the cluster with the highest grade.

For ease of reference, hereafter we refer to the proposed algorithm as simulated-annealing MMC, or SAMMC for short; $k-$means++ as KM; *mini-batch* k-means++ as MBKM; the Dirichlet process Gaussian mixture model as DPGMM; and fuzzy c-means as FCM. All algorithms were implemented in Python 3.5 on a PC with a 5-core CPU and 8 GB RAM.

5.1 Datasets

To test and compare the proposed algorithm, we have carried out experiments with sixteen datasets. A brief description of these datasets is given in Table 1, while a more detailed description can be found in [19], with the exception of D15 which is described hereafter.

Dataset D15 is from the Transport Accident Commission (TAC) which is a major accident compensation agency of the Victorian Government in Australia. It consists of a collection of 593,433 phone calls from 13,937 single TAC clients recorded by various operators over 5 years. The phone calls are made for different purposes including, but not limited to: compensation payments, recovery and return to work, different type of services, medications and treatments, pain, solicitor engagement and mental health issues. We refer to this data set as "Phone Calls".

The following preprocessing steps have been applied to D15 before its use in the experiments: 1) removal of numbers, punctuation, symbols and "stopwords"; 2) synonyms and misspelled words have been replaced with the base and actual words; 3) infrequently occurring words have been removed; 4) as common in text mining, we have also removed the most frequently occurring words such as names and addresses based on a predefined list. The data have then been projected to a vector space by using the popular term frequency-inverse document frequency (tf-idf) scheme [6].

All datasets contain only numeric features and do not have missing values. In brief, the datasets were chosen so that i) the number of attributes would range from very few (2) to many (4,696); and ii) the number of data points would range from thousands (smallest: 2,310) to millions (largest: 4,178,504). Their diversity provides a thorough base for evaluation and comparison.

5.2 Implementation and settings

To implement Algorithm 1, one has to choose parameter δ at Step 1. Values of δ close to one significantly increase computational time. Therefore, we have decreased the value of this parameter with increasing numbers of data points. For the small datasets (D1-D11), we have set it between 0.05 and 0.20 and decreased it to 0.01 for the very large ones (D12-D16).

Table 1: Dataset summary (m is the number of points and n the number of dimensions).

Name	Datasets	m	n
D1	Image Segmentation	2,310	19
D2	Page Blocks	5,473	10
D3	Gas Sensor Array Drift	13,910	128
D4	EEG Eye State	14,980	14
D5	D15112	15,112	2
D6	Online News Popularity	39,797	58
D7	KEGG Metabolic Relation Network	53,413	20
D8	Shuttle Control	58,000	9
D9	Sensorless Drive Diagnosis	58,509	48
D10	MiniBooNE particle identification	130,065	49
D11	Skin Segmentation	245,057	3
D12	3D Road Network	434,874	3
D13	Cover Type	581,012	10
D14	Poker Hand	1,025,010	10
D15	Phone Calls	593,433	4,696
D16	Gas sensor array under dynamic gas mixtures	4,178,504	19

For the implementation of the two alternating steps of KMSVM in Algorithm 2, we have used MBKM and linear SVM from the scikit-learn module in Python. Algorithm 2 terminates if one of the following criteria is satisfied:

- (Temperature drop). If the temperature parameter in simulated annealing drops to a minimum user-defined value. We have set the minimum temperature to 10^{-5}.
- (Number of iterations). If the number of iterations reaches a maximum number defined by the user. We have set the maximum number of iterations to $20,000$.
- (Number of unsuccessful iterations). If the number of unsuccessful iterations exceeds a user-defined value (an unsuccessful iteration is an iteration that does not decrease the objective). It was set to $1,000$.
- (Time consumption). If the CPU time spent exceeds a pre-defined value. The maximum CPU time used by any algorithm is limited to: two hours for datasets Cover Type and Gas sensor array under dynamic gas mixtures; and half an hour for all the other datasets.

Results for all the compared clustering algorithms are reported in Tables 4 and 5.These results are the best output out of 20 runs with different random initializations. In these tables, we have adopted the following notations:

- k is the number of final clusters;
- f_{best} (scaled by the number shown immediately after the name of the dataset) is the best value of the clustering objective function (2);

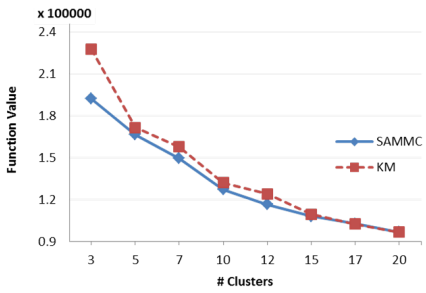
- E_A is the *error* (in %) of algorithm A calculated as follows:

$$E_A = \frac{\bar{f} - f_{best}}{f_{best}} \times 100$$

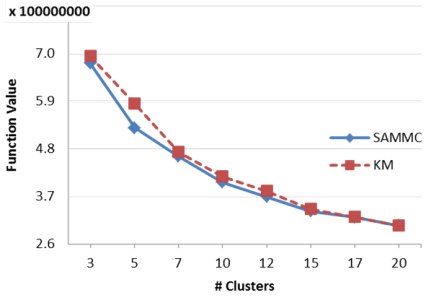
where $\bar{f}$ is the value of the objective function obtained by algorithm A.

5.3 Results and discussion

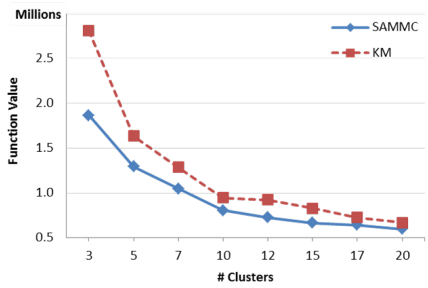
For all datasets, we have computed up to 20 clusters. Since the proposed algorithm, SAMMC, requires an initial KM step in all cases, we have decided to report the objective function values before and after the use of SAMMC (i.e., $f_{initial}$ and f_{final}) in Tables 2 and 3. Tables 4 and 5 instead report the errors of all the compared algorithms.



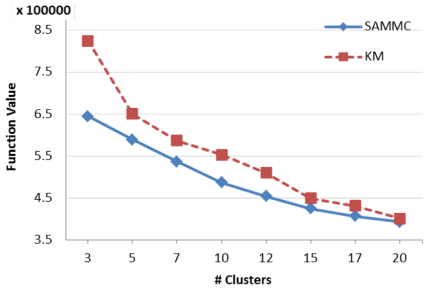
(a) Image Segmentation



(b) Gas Sensor Array Drift



(c) KEGG Relation Network

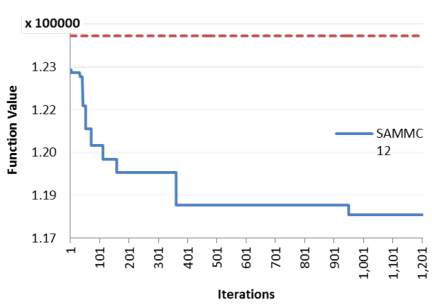


(d) Sensorless Drive Diagnosis

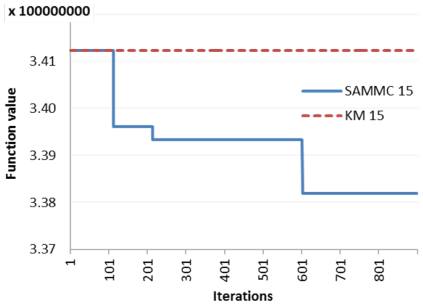
Figure 1: Objective function values for SAMMC and KM with varying number of clusters (subset of four datasets).

Results presented in Tables 4 and 5 show that the proposed SAMMC algorithm has proved the most accurate among all algorithms, followed by KM, FCM, MBKM and DPGMM, respectively. It has been able to find better values of the objective function for most datasets; with particularly significant improvements over Image Segmentation, Page Block, KEGG Metabolic Relation Network, Shuttle Control, Sensorless Drive Diagnosis and 3D Road Network.

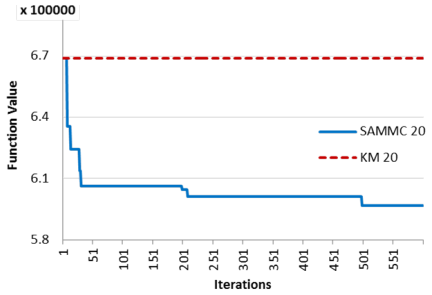
The datasets can be divided in two groups based on their dimensionality. The first group contains the datasets with small dimensionality ($\leq$ 10): Page Blocks, D15112, Shuttle Control, Skin Segmentation, 3D Road Network, Cover Type and Poker Hand.



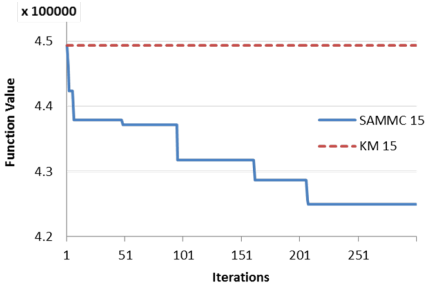
(a) Image Segmentation



(b) Gas Sensor Array Drift



(c) KEGG Relation Network



(d) Sensorless Drive Diagnosis

Figure 2: Objective function values for SAMMC and KM with varying number of iterations (subset of four datasets); notations "SAMMC k" and "KM k" stand for SAMMC and KM with k clusters.

The number of points in these datasets ranges from $15,112$ to $1,025,010$. Results presented in Tables 4 and 5 show that SAMMC improves the results for most of them, with a particularly strong improvement on Page Blocks, Shuttle Control and 3D Road Network. The second group contains datasets with larger number of attributes: Image Segmentation, Gas Sensor Array Drift, EEG Eye State, Online News Popularity, KEGG Metabolic Relation Network, Sensorless Drive Diagnosis, MiniBooNE particle identification, Phone Calls and Gas sensor array under dynamic gas mixtures. The number of dimensions in these datasets ranges from 14 to 4,696. Results presented in Tables 4 and 5 show that SAMMC has been able to improve over the other algorithms and that this improvement is remarkable in most cases.

For further analysis, Figures 1a-1d show the objective function values for algorithms SAMMC and KM over a subset of the datasets when the number of clusters varies. Since SAMMC is initialized with KM, they also show how much SAMMC has been able to improve over its initial KM clustering. Figures 2a-2d show the objective function values for algorithms SAMMC and KM over the same subset when the number of iterations varies. As shown by these figures, the proposed method significantly decreases the objective within the first 500 iterations.

6 Conclusion

In this paper, we have proposed a novel clustering algorithm for clustering under a minimum sum-of-squares objective. The proposed algorithm leverages simulated annealing to escape local minima and a combination of $k-$means++ and SVM to provide high-quality local minima. By exploiting a two-stage organization, the proposed algorithm has been able to mollify the computational complexity of maximum-margin clustering and prove suitable for large-scale data. In the experiments, we have tested it on real-world datasets with number of points ranging from thousands to millions and number of dimensions ranging from a few to thousands. The experimental results have provided clear evidence that the proposed method is able to achieve significant improvement of the objective function in comparison to popular clustering algorithms such as k-means++, mini-batch $k-$means++, DPGMM and fuzzy c-means.

References

[1] D. Arthur and S. Vassilvitskii. K-means++: the advantages of careful seeding. In *Proc. of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.

[2] A.M. Bagirov. Modified global k-means algorithm for minimum sum-of-squares clustering problems. *Pattern Recognition*, 41(10):3192–3199, 2008.

[3] A.M. Bagirov, J. Ugon, and D. Webb. Fast modified global k-means algorithm for incremental cluster construction. *Pattern Recognition*, 44(4):866–876, April 2011.

[4] A.M. Bagirov and J. Yearwood. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *European Journal of Operational Research*, 170(2):578–596, 2006.

[5] L. Bai, J. Liang, C. Sui, and Ch. Dang. Fast global k-means clustering based on local geometrical information. *Information Sciences*, 245:168 – 180, 2013.

[6] J. Beel, B. Gipp, S. Langer, and C. Breitinger. Research-paper recommender systems: a literature survey. *International Journal on Digital Libraries*, 17:305–338, 2016.

[7] J. Duchi. Derivatieves for linear algebra and optimisation. Available in web page <URL: `http://www.cs.berkeley.edu`>, 2007.

[8] F. Gieseke, T. Pahikkala, and O. Kramer. Fast evolutionary maximum margin clustering. In *Proc. of International Conference on Machine Learning (ICML)*, pages 361–368, 2009.

[9] R. Gopalan and J. Sankaranarayanan. Max-margin clustering: Detecting margins from projections of points on lines. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

[10] D. Görür and C.E. Rasmussen. Dirichlet process gaussian mixture models: Choice of the base distribution. *Journal of Computer Science and Technology*, 25:653 – 664, 2010.

[11] Y. Hu, J. Wang, N. Yu, and X. S. Hua. Maximum margin clustering with pairwise constraints. In *Proc. of the 8th IEEE international Conference on Data Mining (ICDM)*, pages 253–262, 2008.

[12] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.

[13] A.K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):651–666, 2010.

[14] Z.S. Karnin, E. Liberty, S. Lovett, R. Schwartz, and O. Weinstein. Unsupervised svms: On the complexity of the furthest hyperplane problem. In *COLT 2012 - The 25th Annual Conference on Learning Theory*, pages 2.1–2.17, 2012.

[15] S. Kirkpatrick, C.D. Gelatt-J, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[16] J. Kogan. *Introduction to Clustering Large and High-dimensional Data*. Cambridge University Press, 2007.

[17] J.Z.C. Lai and T.-J. Huang. Fast global k-means clustering using cluster membership and inequality. *Pattern Recognition*, 43(5):1954 – 1963, 2010.

[18] Y.-F. Li, I.W. Tsang, J.T.-Y. Kwok, and Z.-H. Zhou. Tighter and convex maximum margin clustering. In *Proc. of the 12th international conference on artificial intelligence and statistics*, pages 344–351, 2009.

[19] M. Lichman. UCI machine learning repository. Available in web page <URL: `http://archive.ics.uci.edu/ml`>, University of California, Irvine, School of Information and Computer Sciences, 2001. (April 8th, 2016).

[20] M. Locatelli. Simulated annealing algorithms for continuous global optimization: convergence conditions. *Journal of Optimization Theory and Applications*, 104:121 – 133, 2000.

[21] B. Ordin and A.M. Bagirov. A heuristic algorithm for solving the minimum sum-of-squares clustering problems. *Journal of Global Optimization*, 61:341–361, 2015.

[22] D. Sculley. Web-scale k-means clustering. In *Proc. of the 19th international conference on World wide web*, pages 1177–1178, 2010.

[23] S. Seifollahi, P.A. Dowd, and C. Xu. An enhanced stochastic optimization in fracture network modelling conditional on seismic events. *Computers and Geotechnics*, 16:85–95, 2014.

[24] H. Valizadegan and R. Jin. Generalized maximum margin clustering and unsupervised kernel learning. In *B. Schölkopf, J. Platt and T. Hoffman (Eds.), Advances in Neural Information Processing Systems 19, (2007)*, pages 1417–1424. MIT Press, Cambridge, 2006.

[25] F. Wang, X. Wang, and T. Li. Maximum margin clustering on data manifolds. In *International Conference on Data Mining*, 2009.

[26] R. Winkler, F. Klawonn, and R. Kruse. Fuzzy c-means in high dimensional spaces. *International Journal of Fuzzy System Applications*, 1(1):1–16, 2011.

[27] L. Xu, J. Neufeld, B. Larson, and D. Schuurmans. In *In L. K. Saul, Y. Weiss and L. Bottou (Eds.), Advances in neural information processing systems.*

[28] L. Xu and D. Schuurmans. Unsupervised and semi-supervised multi-class support vector machines. In *Proc. of the 20th National Conference on Artificial Intelligence*, pages 192–199, Pittsburgh, PA, 2005.

[29] K. Zhang, I.W. Tsang, and J.T. Kwok. Maximum margin clustering made practical. In *Proc. of the 24th International Conference on Machine Learning (ICML)*, pages 1119–1126, Corvallis, OR, 2007.

[30] B. Zhao, J. Kwok, F. Wang, and C. Zhang. Unsupervised maximum margin feature selection with manifold regularization. In *Computer Vision and Pattern Recognition*, 2009.

[31] B. Zhao, F. Wang, and C. Zhang. Efficient multiclass maximum margin clustering. In *Proc. of the 25th international conference on Machine Learning (ICML)*, pages 1248–1255, 2008.

Table 2: Objective function values for the SAMMC algorithm over datasets 1-8

k	$\#itr$	$f_{initial}$	f_{final}	$\#itr$	$f_{initial}$	f_{final}
	D1 ($\times 10^5$); $k_0 = 203$			D2 ($\times 10^6$); $k_0 = 849$		
3	2999	2.27535	1.92150	2000	6.92090	4.86011
5	2999	1.71600	1.66362	2000	4.21305	3.30935
7	2000	1.57640	1.49572	2000	3.01114	2.47169
10	2000	1.31979	1.27118	2000	2.76200	1.96377
12	2000	1.24097	1.16674	2000	2.17831	1.78075
15	2000	1.09656	1.08162	2999	2.06897	1.61311
17	2000	1.02597	1.02597	2000	1.73571	1.54352
20	2000	0.96981	0.96691	2000	1.63124	1.41833
	D3 ($\times 10^8$); $k_0 = 1974$			D4 ($\times 10^6$); $k_0 = 2250$		
3	2999	6.94459	6.77215	2000	1.69264	1.69264
5	2999	5.84274	5.28765	2000	1.13563	0.98754
7	2000	4.72804	4.62881	2000	0.87802	0.85799
10	2000	4.02730	4.02730	2000	0.75413	0.75413
12	2000	3.83332	3.70030	2000	0.71172	0.70948
15	2999	3.41227	3.36523	2000	0.67419	0.67418
17	2000	3.23031	3.22069	2000	0.65220	0.64999
20	2000	3.03252	3.03028	2000	0.62949	0.62949
	D5 ($\times 10^7$); $k_0 = 1975$			D6 ($\times 10^9$); $k_0 = 1763$		
3	2999	5.54539	5.54065	1907	3.57554	3.55978
5	2000	4.03324	4.03281	1845	2.56188	2.28574
7	2000	3.41973	3.41973	1826	2.05394	190084
10	2999	2.86699	2.86601	1789	1.68326	1.56733
12	2000	2.62155	2.62132	1764	1.47500	1.42740
15	2000	2.33309	2.33309	1717	1.34470	1.27022
17	2000	2.18249	2.18249	1691	1.26208	1.19329
20	2000	2.02283	2.02283	1642	1.13187	1.12003
	D7 ($\times 10^6$); $k_0 = 1930$			D8 ($\times 10^6$); $k_0 = 1646$		
3	1567	2.80643	1.86366	709	2.51994	1.90781
5	1244	1.63321	1.29238	647	2.46973	1.99779
7	1184	1.28721	1.04727	629	2.40504	1.60107
10	973	0.94686	0.80424	584	1.79075	1.51039
12	963	0.94071	0.74028	571	1.75150	1.39149
15	884	0.82894	0.66603	566	1.72041	1.20582
17	852	0.71239	0.63615	528	1.50015	1.23395
20	843	0.66856	0.59669	548	1.39315	1.11643

Table 3: Objective function values for the SAMMC algorithm over datasets 9-16

k	$\#itr$	$f_{initial}$	f_{final}	$\#itr$	$f_{initial}$	f_{final}
	D9($\times 10^5$); $k_0 = 1889$			D10($\times 10^7$); $k_0 = 3195$		
3	482	8.24640	6.44864	453	4.98413	4.90467
5	333	6.50427	5.89485	456	3.90094	3.88245
7	331	5.86409	5.36534	451	3.25558	3.22204
10	331	5.53054	4.86941	450	2.82944	2.77394
12	326	5.10210	4.54399	447	2.62172	2.62172
15	315	4.49256	4.24934	440	2.54134	2.43630
17	318	4.31557	4.06649	439	2.37069	2.34742
20	321	4.01503	3.93085	442	2.25009	2.23405
	D11($\times 10^7$); $k_0 = 2892$			D12($\times 10^6$); $k_0 = 3526$		
3	363	1.17959	1.17271	151	2.53221	2.45714
5	338	0.87830	0.87783	124	1.57130	1.55483
7	325	0.71552	0.71552	121	1.19048	1.15124
10	302	0.57940	0.57381	120	0.89404	0.85363
12	295	0.53372	0.53317	120	0.759370	0.735879
15	293	0.49592	0.49114	119	0.63639	0.62200
17	291	0.45631	0.44613	117	0.73587	0.56791
20	287	0.41331	0.41206	115	0.52566	0.51077
	D13($\times 10^8$); $k_0 = 3568$			D14($\times 10^6$); $k_0 = 4219$		
3	322	6.84525	6.82126	223	7.73012	7.73012
5	315	5.31869	5.31693	224	7.10038	7.09909
7	314	4.73661	4.73574	227	6.66651	6.66651
10	316	4.12908	4.12908	251	6.15894	6.15712
12	315	3.88909	3.88898	264	5.95588	5.94458
15	313	3.55111	3.52572	250	5.63555	5.63519
17	310	3.41428	3.41428	242	5.49580	5.49443
20	309	3.23589	3.22004	283	5.35294	5.35294
	D15($\times 10^4$); $k_0 = 1558$			D16($\times 10^{10}$); $k_0 = 3812$		
3	383	5.27089	5.27052	41	2.19410	2.19334
5	384	5.25709	5.25586	40	1.75273	1.75209
7	380	5.25277	5.24668	40	1.52607	1.52607
10	376	5.23029	5.22822	38	1.25024	1.25024
12	379	5.22126	5.21523	38	1.15669	1.14279
15	375	5.20914	5.20049	38	1.00832	0.99651
17	362	5.20057	5.20057	39	0.95343	0.92735
20	349	5.18291	5.17676	39	0.85912	0.85175

Table 4: Clustering errors obtained with the compared algorithms over datasets 1-8; for compactness of notation, E_1, E_2, E_3, E_4 and E_5 are the errors obtained using KM, MBKM, DPGMM, FCM and SAMMC, respectively

k	f_{best}	E_1	E_2	E_3	E_4	E_5	f_{best}	E_1	E_2	E_3	E_4	E_5
	D1 ($\times 10^5$)						D2 ($\times 10^6$)					
3	1.92065	18.46	0.34	35.42	0.00	0.04	4.86011	42.40	7.06	29.41	11.57	0.00
5	1.66362	2.83	3.07	22.57	1.29	0.00	3.30935	27.30	1.84	67.69	5.55	0.00
7	149572	5.39	7.81	31.29	1.01	0.00	2.47169	21.82	18.03	90.63	18.67	0.00
10	1.27072	3.86	0.00	49.03	1.76	0.03	1.96377	40.64	32.27	109.14	13.35	0.00
12	1.16674	6.36	1.00	55.16	3.31	0.00	1.78075	22.32	10.16	108.26	13.85	0.00
15	1.08162	1.38	2.95	58.82	2.02	0.00	1.61311	28.26	46.55	142.47	1.19	0.00
17	1.02360	0.23	0.00	62.20	5.55	0.23	1.54352	12.45	52.57	88.48	0.54	0.00
20	0.96691	0.30	0.35	89.45	4.74	0.00	1.41833	15.01	86.40	123.33	0.62	0.00
	D3 ($\times 10^8$)						D4 ($\times 10^6$)					
3	6.77215	2.55	0.07	56.14	1.52	0.00	1.69264	0.00	153.30	24.26	0.00	0.00
5	5.28765	10.50	2.32	59.50	2.79	0.00	0.98754	14.99	182.78	15.92	0.18	0.00
7	4.62881	2.14	1.75	77.33	1.78	0.00	0.85799	2.33	377.50	11.05	2.47	0.00
10	4.02730	3.35	4.37	64.53	3.74	0.00	0.74556	0.40	336.08	53.54	3.95	0.00
12	3.70030	3.59	6.99	66.01	3.62	0.00	0.70948	0.32	159.20	24.02	6.55	0.00
15	3.36523	1.40	0.01	66.60	3.41	0.00	0.67418	0.00	284.78	69.80	9.92	0.00
17	3.22069	0.30	6.80	43.15	6.42	0.00	0.64999	0.34	513.23	35.70	11.16	0.00
20	3.03252	0.07	10.25	66.34	2.64	0.00	0.62949	0.00	306.83	81.85	11.81	0.00
	D5 ($\times 10^7$)						D6 ($\times 10^9$)					
3	5.53855	0.13	0.07	24.29	0.00	0.04	3.55978	0.59	7.50	37.91	0.00	0.15
5	4.02884	0.11	1.64	142.09	0.00	0.10	2.28574	12.08	13.27	144.65	0.14	0.00
7	3.41973	0.00	1.77	14.89	1.37	0.00	1.90084	8.05	5.17	128.04	1.69	0.00
10	2.86601	0.03	1.73	240.54	2.06	0.00	1.56733	7.39	0.64	222.86	8.29	0.00
12	2.62132	0.01	1.66	11.66	0.25	0.00	1.42740	3.33	2.88	227.17	10.70	0.00
15	2.32964	0.15	1.33	318.44	0.00	0.15	1.27022	5.86	2.93	298.94	7.63	0.00
17	2.17353	0.41	2.54	9.45	0.00	0.41	1.19329	5.76	6.11	239.74	12.15	0.00
20	2.00347	0.97	3.55	383.42	0.00	0.97	1.12003	1.06	4.24	335.47	13.85	0.00
	D7 ($\times 10^6$)						D8 ($\times 10^6$)					
3	1.86366	50.58	8.30	10.66	15.14	0.00	1.90781	34.47	2.91	20.36	0.00	2.39
5	1.29238	26.37	5.36	25.04	14.04	0.00	1.77075	42.88	5.87	14.64	0.00	3.43
7	1.04727	22.911	11.421	88.220	16.039	0.00	1.60107	50.21	12.03	30.62	0.43	0.00
10	0.80424	17.73	12.62	59.07	21.79	0.00	1.51039	18.56	4.61	23.33	4.19	0.00
12	0.74028	27.07	18.37	136.27	17.39	0.00	1.39149	25.87	10.68	39.76	9.84	0.00
15	0.66603	24.46	26.33	74.73	16.09	0.00	1.20582	42.68	9.65	34.08	24.37	0.00
17	0.63615	11.98	22.19	128.04	13.34	0.00	1.23395	21.57	9.50	53.30	18.83	0.00
20	0.59669	12.04	25.43	101.41	14.94	0.00	1.11643	24.78	25.16	53.49	29.26	0.00

Table 5: Clustering errors obtained with the compared algorithms over datasets 9-16; for compactness of notation, E_1, E_2, E_3, E_4 and E_5 are the errors obtained using KM, MBKM, DPGMM, FCM and SAMMC, respectively

k	f_{best}	E_1	E_2	E_3	E_4	E_5	f_{best}	E_1	E_2	E_3	E_4	E_5
	D9 ($\times 10^5$)						D10 ($\times 10^7$)					
3	6.44864	30.29	2.85	6.05	0.00	2.40	4.90467	1.62	49.06	93.91	41.52	0.00
5	5.65453	15.03	0.00	11.23	0.73	4.25	3.88245	2.29	67.57	107.18	0.00	1.80
7	5.11723	14.59	0.00	27.83	1.30	4.84	3.22204	1.04	84.64	43.40	97.54	0.00
10	4.86941	13.58	3.01	20.64	4.68	0.00	2.77394	2.00	105.57	179.43	123.64	0.00
12	4.49953	13.39	0.00	38.67	3.47	0.98	2.62172	0.00	131.61	61.21	133.11	0.00
15	4.24934	5.724	0.12	15.91	15.31	0.00	2.43630	4.31	124.53	215.33	148.27	0.00
17	4.06649	6.12	0.42	41.43	6.68	0.00	2.31230	2.52	0.00	76.76	160.27	1.51
20	3.93085	2.142	1.86	14.36	21.44	0.00	2.23405	0.72	138.92	204.42	167.56	0.00
	D11 ($\times 10^7$)						D12 ($\times 10^6$)					
3	1.17271	0.59	2.36	16.11	0.59	0.00	2.45714	3.06	0.24	1.61	2.11	0.00
5	0.87783	0.05	2.52	12.32	0.01	0.00	1.55483	1.06	6.56	0.93	0.66	0.00
7	0.71553	2.66	2.59	65.33	2.57	0.00	1.15124	3.40	0.44	188.85	2.36	0.00
10	0.57382	0.97	13.07	37.44	3.08	0.00	0.85363	4.73	3.66	14.28	2.33	0.00
12	0.53317	0.10	4.42	51.60	2.61	0.00	0.73587	3.19	3.67	457.07	1.50	0.00
15	0.47046	5.71	0.00	65.30	2.75	1.32	0.62200	2.31	1.04	35.41	1.30	0.00
17	0.45402	4.79	0.41	48.42	3.86	0.00	0.56791	2.27	2.70	607.59	1.05	0.00
20	0.41271	1.31	4.52	69.23	6.373	0.00	0.51077	2.91	1.70	48.86	0.85	0.00
	D13 ($\times 10^8$)						D14 ($\times 10^6$)					
3	6.82126	0.35	0.74	51.93	0.28	0.00	7.73012	0.00	0.56	0.88	1.07	0.00
5	5.30732	0.21	3.91	95.28	0.00	0.18	7.09909	0.02	0.47	3.92	1.44	0.00
7	4.72561	0.24	2.36	118.52	0.00	0.19	6.66651	0.00	1.48	7.80	8.00	0.00
10	4.12908	0.00	2.59	50.03	0.33	0.00	6.15712	0.03	0.90	13.65	10.83	0.00
12	3.88312	0.16	1.80	165.64	0.00	0.09	5.94458	0.19	0.93	15.12	10.52	0.00
15	3.52572	0.72	1.21	192.69	2.08	0.00	5.63519	0.01	1.78	19.47	11.70	0.00
17	3.41428	0.00	2.99	191.43	1.04	0.00	5.49443	0.03	1.77	19.94	17.41	0.00
20	3.22004	0.49	1.98	220.24	1.71	0.00	5.35294	0.00	0.98	22.20	20.46	0.00
	D15 ($\times 10^4$)						D16 ($\times 10^{10}$)					
3	5.27052	0.01	0.22	0.11	0.10	0.00	2.19334	0.03	0.25	109.71	0.08	0.00
5	5.25586	0.03	0.49	0.08	0.30	0.00	1.75209	0.04	0.25	157.66	0.05	0.00
7	5.24668	0.12	0.46	0.19	0.40	0.00	1.52578	0.02	0.00	170.02	0.62	0.02
10	5.22822	0.04	0.77	0.23	0.60	0.00	1.25024	0.00	0.87	266.32	1.45	0.00
12	5.21523	0.12	0.67	0.22	0.76	0.00	1.14279	1.21	0.86	147.42	1.18	0.00
15	5.20049	0.17	1.16	0.26	1.02	0.00	0.99651	1.19	1.99	330.68	4.74	0.00
17	5.19723	0.05	1.08	0.00	0.95	0.05	0.92735	2.81	4.08	183.29	2.55	0.00
20	5.17676	0.12	1.15	0.43	1.32	0.00	0.85175	0.87	0.43	418.83	3.83	0.00