JYU DISSERTATIONS 43

Joonas Hämäläinen

Improvements and Applications of the Elements of Prototype-Based Clustering

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION
TECHNOLOGY

JYU DISSERTATIONS 43

Joonas Hämäläinen

Improvements and Applications of the Elements of Prototype-Based Clustering

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella
julkisesti tarkastettavaksi yliopiston Agora-rakennuksen Gamma-salissa
joulukuun 14. päivänä 2018 kello 12.

Academic dissertation to be publicly discussed, by permission of
the Faculty of Information Technology of the University of Jyväskylä,
in building Agora, Gamma hall, on December 14, 2018 at 12 o'clock noon.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2018

Editors
Timo Männikkö
Faculty of Information Technology, University of Jyväskylä
Ville Korkiakangas
Open Science Centre, University of Jyväskylä

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-7621-7

ISBN 978-951-39-7621-7 (PDF)
URN:ISBN:978-951-39-7621-7
ISSN 2489-9003

ABSTRACT

Hämäläinen, Joonas
Improvements and Applications of the Elements of Prototype-Based Clustering
Jyväskylä: University of Jyväskylä, 2018, 58 p.(+included articles)
(JYU Dissertations
ISSN 2489-9003; 43)
ISBN 978-951-39-7621-7 (PDF)
Finnish summary
Diss.

Clustering or cluster analysis is an essential part of data mining, machine learning, and pattern recognition. The most popularly applied clustering methods are partitioning-based or prototype-based methods. Prototype-based clustering methods usually have easy implementability and good scalability. These methods, such as K-means clustering, have been used for different applications in various fields. On the other hand, prototype-based clustering methods are typically sensitive to initialization, and the selection of the number of clusters for knowledge discovery purposes is not straightforward. In the era of big data, in high-velocity, ever-growing datasets, which can also be erroneous, outlier intensive and sparse, research has arisen focused on the development of efficient prototype-based clustering methods for more challenging datasets. This collection of articles primarily focuses on developing prototype-based clustering for more scalable, efficient and reliable data processing. To achieve these goals, improvements and modifications have been made to prototype-based clustering in six included articles. Additionally an application of the prototype-based clustering to supervised learning in regression problems is also covered. In general, these efforts advance the knowledge discovery process towards more reliable data processing and big data.

Keywords: knowledge discovery, data mining, machine learning, prototype-based clustering, big data, parallel computing, robust clustering, clustering initialization, K-means, minimal learning machine, random projection

Author

Joonas Hämäläinen
Faculty of Information Technology
University of Jyväskylä
Finland

Supervisors

Professor Tommi Kärkkäinen
Faculty of Information Technology
University of Jyväskylä
Finland

Professor Tuomo Rossi
Faculty of Information Technology
University of Jyväskylä
Finland

Reviewers

Professor Martti Juhola
Faculty of Natural Sciences
University of Tampere
Finland

Professor Ajalmar Rêgo da Rocha Neto
Department of Teleinformatics
Federal Institute of Ceará
Brazil

Opponent

Professor Pasi Fränti
School of Computing
University of Eastern Finland
Finland

TIIVISTELMÄ (FINNISH ABSTRACT)

Hämäläinen, Joonas
Prototyyppipohjaisen klusteroinnin elementtien parannukset ja sovellukset
Jyväskylä: University of Jyväskylä, 2018, 58 s.(+artikkelit)
(JYU Dissertations
ISSN 2489-9003; 43)
ISBN 978-951-39-7621-7 (PDF)

Klusterointi eli klusterianalyysi on keskeinen osa-alue tiedonlouhinnassa, koneoppimisessa ja hahmontunnistuksessa. Sovelluksissa käytetyimpiä ovat osittavat eli prototyyppipohjaiset klusterointimenetelmät. Prototyyppipohjaiset klusterointimenetelmät ovat usein helposti toteutettavissa ja ne skaalautuvat hyvin. Näitä menetelmiä, kuten K-means-klusterointia, on hyödynnetty monissa eri sovelluksissa eri tutkimusaloilla. Toisaalta prototyyppipohjaiset klusterointimenetelmät ovat alustukselle herkkiä eikä klustereiden lukumäärän valinta ole suoraviivaista. Big datan aikakaudella nopeasti kasvavat tietomassat, jotka voivat myös olla virheellisiä, anomaliaintensiivisiä ja harvoja, ohjaavat tutkimusta tehokkaiden prototyyppipohjaisten klusterointimenetelmien kehittämiseen haastaville datajoukoille. Tämä artikkeliväitöskirja keskittyy pääasiassa kehittämään datan prosessointia prototyyppipohjaisella klusteroinnilla skaalautuvammaksi, tehokkaammaksi ja luotettavammaksi. Näiden tavoitteiden saavuttamiseksi kuudessa väitöskirjaan kuuluvassa artikkelissa on tehty parannuksia ja modifikaatioita prototyyppipohjaiseen klusterointiin. Lisäksi prototyyppipohjaisen klusteroinnin sovellusta ohjattuun oppimiseen regressio-ongelmissa on käsitelty yhdessä artikkelissa. Yleisesti väitöskirjan tulokset kehittävät tietämyksen muodostamisprosessia kohti luotettavampaa datan prosessointia ja skaalautuvampaa big datan prosessointia.

Avainsanat: tietämyksen muodostaminen, tiedonlouhinta, koneoppiminen, prototyyppipohjainen klusterointi, big data, rinnakkaislaskenta, robusti klusterointi, klusteroinnin alustaminen, K-means, minimaalinen oppimiskone, satunnaisprojektio

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Professor Tommi Kärkkäinen for all his support, guidance and help along the way. His genuine enthusiasm towards data mining and machine learning, scientific expertise, and heterogeneous pool of ideas have been priceless. I am also grateful to my supervisor Professor Tuomo Rossi whose support and scientific expertise in parallel computing have been valuable. It has been a pleasure to work with both of them.

For reviewing the dissertation manuscript and providing insightful comments I would like to thank Professor Martti Juhola and Professor Ajalmar Rêgo da Rocha Neto. I thank collaborators Professor João Paulo Pordeus Gomes, Dr. Mirka Saarela and Ms. Susanne Jauhiainen for fruitful research. I would also like to thank all my colleagues and particularly Dr. Mirko Myllykoski, Dr. Paavo Nieminen and Dr. Pekka Wartiainen for support and many interesting discussions. I would also like to thank Mr. Pertti Henttu who got me interested in data mining and machine learning.

I am grateful for the financial support of the Faculty of Information Technology at University of Jyväskylä and Ellen and Artturi Nyyssönen Foundation. I would also like to thank my family members, friends and relatives for their support outside research work. Finally, I am grateful to my beloved wife Susanna for her endless support along the way of my PhD studies, and patience for my unusual working hours, especially at the end of the dissertation work.

GLOSSARY

AI	Artificial intelligence
C-MLM	Cubic equation minimal learning machine
CVI	Cluster validation index
DB	Davies–Bouldin
DB-SCV	Distribution balanced stratified cross-validation
DOB-SCV	Distribution optimally balanced stratified cross-validation
ELM	Extreme learning machine
ESCS	Economic, social and cultural status
KDD	Knowledge discovery in databases
K-means||	Scalable K-means++
K-NN	K-nearest neighbors
KW	Kruskal–Wallis
MDCS	MATLAB distributed computing server
MLM	Minimal learning machine
MLP	Multi-layer perceptron
NN-MLM	Nearest neighbor minimal learning machine
ON	Opposite neighbors
PBM	Pakhira–Bandyopadhyay–Maulik
PCA	Principal component analysis
PCT	Parallel computing toolbox
PISA	Programme for international student assessment
RMSE	Root-mean-squared error
RQ	Research question
RS	Reference points selection
RT	Ray–Turi
SCV	Stratified-cross validation
SK-means||	Subset K-means||
SOR	Successive over-relaxation
SPMD	Single program multiple data
SRPK-means||	Subset random projection K-means||
SSE	Sum-of-squared error
UPGMA	Unweighted pair group method with arithmetic mean
WG	Wemmert–Gançarski

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Joonas Hämäläinen and Tommi Kärkkäinen. Initialization of Big Data Clustering using Distributionally Balanced Folding. *ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2016.

PII Joonas Hämäläinen, Tommi Kärkkäinen and Tuomo Rossi. Scalable initialization methods for clustering large datasets. *Pattern Recognition Letters (in revision)*, 2018.

PIII Joonas Hämäläinen, Tommi Kärkkäinen and Tuomo Rossi. Scalable robust clustering method for large and sparse data. *ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2018.

PIV Joonas Hämäläinen, Susanne Jauhiainen and Tommi Kärkkäinen. Comparison of Internal Clustering Validation Indices for Prototype-Based Clustering. *Algorithms, 10(3):105*, 2017.

PV Mirka Saarela, Joonas Hämäläinen and Tommi Kärkkäinen. Feature Ranking of Large, Robust, and Weighted Clustering Result. *PAKDD 2017 proceedings, Advances in Knowledge Discovery and Data Mining: 21st Pacific-Asia Conference*, 2017.

PVI Joonas Hämäläinen, Tommi Kärkkäinen and João P. P. Gomes. Clustering-Based Reference Points Selection for the Minimal Learning Machine. *Manuscript*, 2018.

1 INTRODUCTION

The general orientation of this thesis is to refine the KDD process and machine learning via contributions related to prototype-based clustering. These contributions advance the KDD process towards more reliable data processing and big data. Primarily these contributions provide methods with a random nature that are aimed for efficient and scalable data processing. In this chapter, background, and motivation for the thesis are given in Section 1.1. Section 1.2 gives the objectives of the thesis and research questions. Finally, Section 1.3 places the included six articles in the context of cluster analysis procedure and provides a structure for the thesis.

1.1 Background and motivation

Grouping or organizing objects into groups, in terms of how similar or dissimilar they are, is a natural way to summarize a collection of objects. This is referred to as clustering or cluster analysis, which is one of the essential parts of data mining [117], pattern recognition [61] and machine learning [14]. Particularly, clustering has a significant role in data mining [10, 107]. In data mining, clustering can be classified under the descriptive modeling task that aims to describe the data [57, pp. 12–15]. Other tasks of data mining, according to Hand et. al. [57, pp. 12–15], are exploratory data analysis (dimensionality reduction, visualization of the data), predictive modeling (classification, regression), discovery of patterns and rules (anomaly detection, association rules) and retrieval by content (finding similar items) [57, pp. 12–15].

More generally, data mining is one of the core steps in the knowledge discovery in databases (KDD) process [46] where the main aim is to extract knowledge for further usage from large volumes of data that are growing rapidly. An overview of the main steps of the KDD process is shown in Figure 1 where the steps are selection (subset of samples and variables are selected), preprocessing (e.g., noise removal, missing value imputation), transformation (selection or ex-

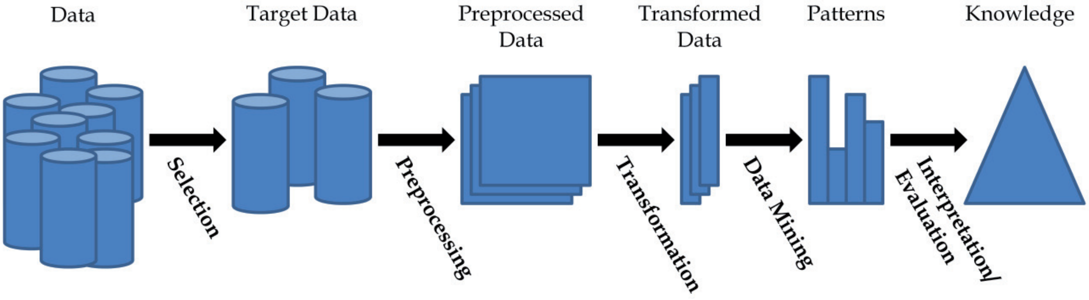


FIGURE 1 KDD process.

traction of features), data mining (extraction of patterns, for example with clustering, regression and classification) and interpretation/evaluation (e.g., visualization of the extracted patterns). The KDD process motivates the development of efficient and scalable data mining methods for processing large volumes of data and a further boost towards this direction is given by big data. The term big data refers to vast volumes of heterogeneous data growing at a rapid rate. Roughly, according to Chen et al. [24], big data can be considered as the datasets that common IT tools or software are not able to realize, manage or process in reasonable time. Therefore, there is an urgent need for researchers to clustering algorithms with improved speed and scalability [107]. Currently, processor and memory development cannot keep up with the rapid growth of data; therefore, multiple machines approaches relying on distributed and parallel computing are needed [107].

Clustering aims to find structure in the data that is constructed with groups, called clusters, in which observations are similar to each other [62]. More formally, it is a task that aims to find K groups for a given presentation of N objects with a selected similarity measure so that in each group similarity is high between objects and similarity is low between objects from different groups (dissimilarity between groups is high) [61]. The clustering task is challenging: an ideal cluster can be defined as compact and isolated, but in practice, it is in the eye of the beholder [61]. According to Xu and Wunsch [118], the clustering process contains four main steps: 1) feature selection or extraction (dimensionality reduction), 2) clustering algorithm design or selection, 3) cluster validation and 4) interpretation of results. In Figure 2 this process is depicted following these four steps. Note that in this procedure workflow can also propagate backward and each step has a significant impact on the final results, similar to the KDD process. In the dimensionality reduction step, selecting a subset of features or transforming features to build new features simplifies data to a reduced representation and can improve the effectiveness of the clustering process. In the second step, a suitable clustering algorithm for dimensionally-reduced data is selected/designed and employed. The second step can include defining the clustering objective function and similarity/dissimilarity measure. In cluster validation, a clustering structure given by the selected clustering algorithm has to be evaluated. Most often, this is conducted with cluster validation indices (CVIs), which aim to give a quality measure of a clustering structure. Typically, different parameter settings and clustering algorithms are tested to obtain different clustering structures, and the best clustering structure according to the CVI(s) is selected for interpretation to extract knowledge.

In general, clustering algorithms can be divided into two branches, partitional and hierarchical, based on cluster structure properties [118]. Partitional clustering constructs a single-layer clustering structure whereas hierarchical generates a tree-type clustering structure with multiple layers of different groupings. Typically, partitional clustering methods are based on representing clusters with prototype points; therefore, partitional clustering is also referred to as prototype-based clustering [97, 6]. Due to the quadratic time and space complexity, classical hierarchical clustering methods are not suitable for large-scale datasets [118, 25], although efficient techniques have been proposed [122, 26]. In general, from the point of view of scalability partitional clustering methods are often better suited for large-scale datasets [40]. Besides this basic division of clustering methods, there exists a vast amount of different types of clustering approaches such as fuzzy clustering [88], spectral clustering [89], neural network-based clustering [74], density-based clustering [42], model-based clustering [18], and evolutionary computing-based clustering [94, 41].

A recently proposed randomized learning machine approach for supervised learning, the minimal learning machine (MLM) [32, 31] can be used to form classification or regression models of data. The MLM forms a linear regression model between input and output distance matrices which are computed concerning a subset of points referred to as reference points. The MLM has comparable performance to many state-of-the-art supervised learning methods, easy implementability and only one parameter that has to be optimized, the number of reference points, during the learning procedure [85]. Lately the MLM has been applied to human activity recognition [84], robotics [83] and cancer classification [96].

1.2 Research questions

This thesis mainly focuses on prototype-based clustering. The most central prototype-based clustering methods related to this thesis are the well-known K-means [81, 61] and a robust variant of it, K-spatialmedians [6, 70, 68, 7]. One of the main objectives is to improve KDD via contributions to prototype-based clustering regarding initialization, scalability and cluster validation. In addition, this work also covers application of prototype-based clustering in supervised learning especially in regression problems with the MLM.

The research questions of this thesis are as follows.

RQ1: What techniques are beneficial for improving prototype-based clustering towards more efficient, scalable and reliable data processing?

RQ2: Can K-spatialmedians be scaled to large-scale clustering with parallelization?

RQ3: What are beneficial clustering validation indices in the context of prototype based clustering?

RQ4: Can clustering-based reference points selection improve performance of the MLM in regression tasks?

1.3 Structure of the work

The relationship of the articles to different steps of the clustering procedure [118] is shown in Figure 2. The included articles are mainly focused on the second step, clustering algorithm design and selection. The article [PII] is partially related to the dimensionality reduction step. The articles [PIV] and [PVI] focus on cluster validation and interpretation of results, respectively. Moreover, the most common themes in the articles are prototype-based clustering initialization and scalability/big data.

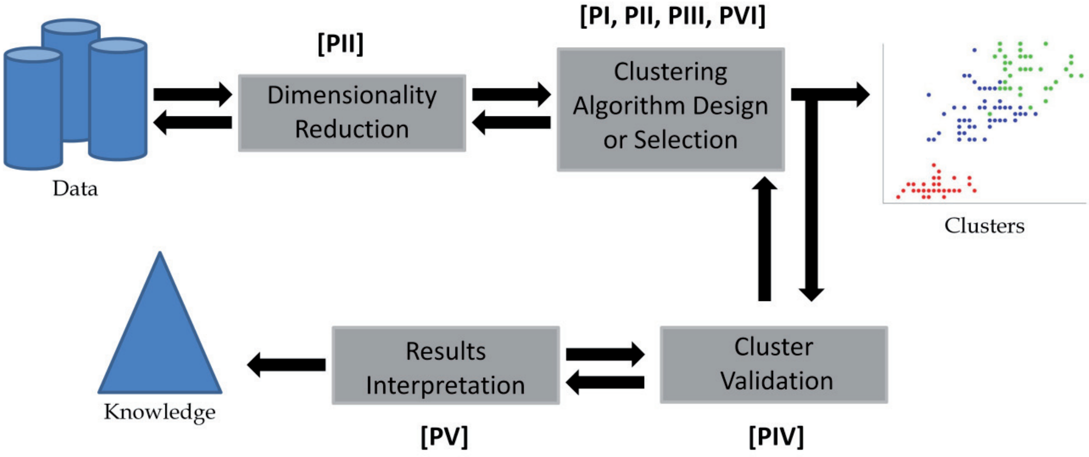


FIGURE 2 Included articles related to the steps of the cluster analysis procedure.

In Chapter 2, background related to clustering and prototype-based clustering is given. This chapter focuses on important topics related to prototype-based clustering. After a general description of prototype-based clustering, more specific descriptions and discussion of the K-means and K-spatialmedians clustering are given. Finally, topics related to cluster validation and big data clustering are discussed. Chapter 3 gives an introduction to supervised learning, mostly in terms of regression. The MLM is described and discussed in more detail. Chapter 4 summarizes the included articles and contributions of this thesis. Finally, Chapter 5 gives the conclusions of the thesis and the direction of future work.

2 CLUSTERING

First, this chapter gives general background to clustering in Section 2.1. In Section 2.2, a more detailed introduction to prototype-based clustering with the most relevant methods related to this thesis is given. In Section 2.3, cluster validation is briefly introduced. Finally, basic concepts and techniques of big data clustering are discussed in Section 2.4.

2.1 Background

Label information for objects is missing in clustering tasks, and clustering is usually performed in unsupervised manner. Sometimes labeling of a subset of observations is performed prior to clustering. Semi-supervised clustering refers to utilization of this prior information, a specification of pair-wise constraints (cannot-link and must-link) for cluster labels, in a clustering process [61]. Besides unsupervised and semi-supervised clustering, another high-level division, how clustering is performed, is the distinction of crisp (hard) and fuzzy (soft) clustering. In crisp clustering, each point is a member of one cluster only, in contrast to fuzzy clustering where each point has membership function values (between 0 and 1) for the similarity for each of the K clusters [12]. In this thesis, the focus is on unsupervised crisp clustering with similarity measures defined by individual cases of the Minkowski distance (the Euclidean, the squared Euclidean and city-block distances).

For a given dataset, clustering can be performed in many ways, because the best clustering structure depends on its further usage [62]. For example, the best clustering structure in a data mining task aiming to find meaningful groupings for the discovery of new knowledge [100], clustering-based anomaly detection [3], clustering application to supervised learning [PVI] and data compression with less expensive clustering method for more expensive clustering method [37] may all be profoundly different. In the first example, the best structure for finding a suitable grouping for the discovery of new trends or patterns can be based on some numerical measure describing how compact the clusters are and how well separated they are from other clusters. In the second example, the best clustering result may be the one which gives the highest performance for detecting anomalies which can be evaluated, for example, with a receiver operating characteristic curve analysis [45, 51]. In the third and fourth examples, the best clustering structure may be the one which minimizes the complexity of the model without compromising accuracy. In big data clustering, a desired clustering structure may even be the structure that can be constructed with the smallest computational cost. It is, therefore, no surprise that there exists a vast and heterogeneous pool of clustering algorithms that has been aggregated to its current form from initial algorithms dating back to the 1950s.

Partitional and hierarchical clustering differ in their resulting clustering structure and also they are built in different ways. One of the most well-known partitional clustering methods is the K-means [81], which forms a grouping by iteratively updating cluster prototypes based on mean location estimate and the Euclidean distance. Typically, the direction of propagation for hierarchical clustering is either from the bottom up (agglomerative), with each object forming a cluster initially, or, from the top to down (divisive), with all objects as members of one cluster initially. Typically, hierarchical clustering uses a proximity matrix to a form hierarchical clustering structure represented by a dendrogram or binary tree, where each leaf node represents data object, and the root represents the whole dataset [118]. Moreover, partitional clustering can also be performed hierarchically by iteratively using partitional clustering for each cluster [108, 115].

2.2 Prototype-based clustering

It is possible to obtain the single layer clustering structure given by partitional clustering that uses some objective function, by simply enumerating all possible groupings for K groups [118]. The number of different possible groupings of N points into K clusters, already for small datasets, is vast. The number of these groupings is given by the Stirling numbers of the second kind [39]

$$S = \frac{1}{K!}\sum_{i=0}^{K}(-1)^{K-i}\binom{K}{i}i^N. \tag{1}$$

An approximate value for (1) can be computed with $K^N/K!$ [39]. Therefore, using brute-force search to find optimal cluster structure is not an appropriate approach. For example, by applying the approximative formula, the number of different groupings for the classic iris dataset ($N = 150$) for $K = 3$ is $3^{150}/6 \approx 6.2 \times 10^{70}$; which is not far from the same order of magnitude as the number of atoms in the known universe.

In general, prototype-based clustering methods consist of two main steps [6, 56]: 1) selection of K initial prototypes and 2) iterative refinement of the prototypes until convergence (see Algorithm 1). In practice, the most typical approach is to employ a non-deterministic method in step 1, followed by refinement with step 2 with multiple restarts of step 1 [118]. Then the final clustering result, selected out of the multiple restarts, will correspond to the clustering result with the smallest clustering error.

Algorithm 1 General prototype-based clustering

```
1) Select $K$ initial prototypes.
2) Refine prototypes until convergence.
  2.1) Find the closest prototypes for $N^*$ points.
  2.2) Recompute cluster prototypes.
```

Step 2 can be broken down into two sub-steps: 2.1) search of the closest prototypes for N^* observations and 2.2) recomputation of K cluster prototypes. For sub-step 2.1, two main types can be distinguished, the case $N^* = N$ refers to a batch version and $N^* = 1$ refers to an incremental version. This thesis focuses on the batch version of Algorithm 1.

2.2.1 K-means

The K-means method, especially the batch version, has a wide variety of applications, including vector quantization [72], anomaly detection [51], image clustering [52, 90] and image segmentation [95, 91]. Due to the K-means' effectiveness, it is often employed for preprocessing or initialization of more expensive algorithms [125, 37]. Moreover, K-means is identified as one of the top 10 most influential data mining algorithms [117]. The K-means algorithm can serve as a prototype method of partitional clustering. Hence, it is often used to demonstrate and compare new partitional clustering ideas.

The K-means clustering originates from the optimization problem in which the goal is to find K cluster prototypes which minimize sum-of-squared error (SSE), a sum of the squared Euclidean distances (L_2 norms) of the points to their closest cluster prototypes. For the dataset $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ where $\mathbf{x_i} \in \mathbb{R}^M$ and for the set of cluster prototypes $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$ where $\mathbf{c_k} \in \mathbb{R}^M$, the objective function, SSE, for the K-means clustering problem is defined as

$$SSE(\mathbf{C}) = \sum_{\mathbf{x} \in \mathbf{X}} \min_{\mathbf{c} \in \mathbf{C}} \|\mathbf{c} - \mathbf{x}\|^2. \tag{2}$$

It can be proved that the best prototype for minimizing the SSE objective function within a cluster is the sample mean of the points within a cluster [97]. The most popular heuristic to solve clustering problem (2) approximately is Lloyd's algorithm, which is the standard batch version of the K-means algorithm. Lloyd's algorithm follows the Algorithm 1 skeleton with $N^* = N$ and represents the sample mean of the points within a cluster as a prototype.

There also exist many variants of K-means for step 2.1 in Algorithm 1 where N^* can vary between $1 \leq N^* \leq N$ which are based on, such as pruning distance computations [38, 54, 36] or sampling [104]. Computational requirements of the batch version of the K-means algorithm mainly build on the distance computations in step 2.1 [54, 55] which contain a large portion of redundant distance computations after few iterations. Therefore, the distance pruning can speed up the computations 30–50 times with respect to Lloyd's algorithm without affecting the clustering results at all [55].

A large portion of the clustering effort in K-means, in terms of clustering error, is performed in the first few iterations [15, 20]. This is demonstrated in Figure 3, which shows relative SSE improvement with respect to the previous iteration's SSE ($SSE_r = (SSE_{t-1} - SSE_t)/SSE_{t-1}$, where t is iteration) as a function of iterations for the S1-S4 datasets (description of the datasets in [50]) by employing K-means++ with $K = 15$. Note that SSE_r is averaged over 100 clustering results. From Figure 3, one can observe that the first iteration improves the clustering error most significantly, $30\% - 40\%$, and for iterations $2 - 4$ improvement is around 5%. Overall, improvement to clustering error decreases rapidly as a function of iterations. This characteristic was utilized in [PII] to speed up initialization of K-means clustering.

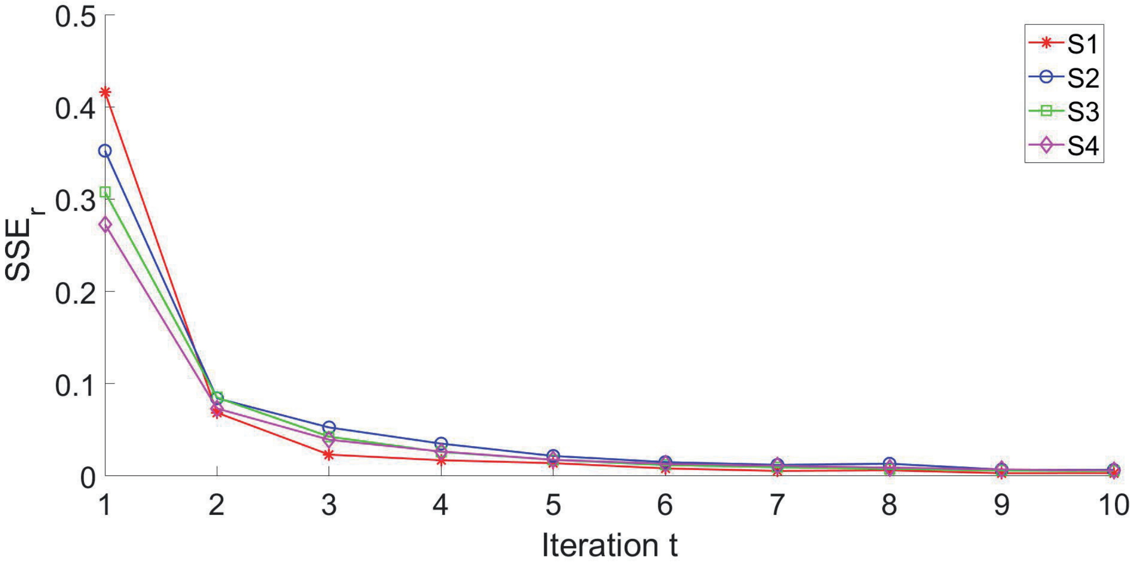


FIGURE 3 Demonstration of relative clustering error improvement as a function of iterations for K-means clustering.

2.2.2 Initialization of K-means

It is a well-known fact that the K-means is sensitive to the selection of the initial prototypes or partitions [92, 39]. A large and heterogeneous pool of initialization methods proposed in frequently cited papers further confirms this issue [49, 82, 53, 19, 5, 9]. Moreover, a September 2018 Google Scholar search for "K-means initialization", for the year 2017 alone, gives 11,200 results. Currently, the most popular initialization strategy is K-means++ [5] that selects initial prototypes iteratively based on a probability distribution using the squared Euclidean distance that is updated after each selection. K-means++ is currently the state-of-the-art method for the initialization [8], it is widely implemented in various platforms [76] and it is an active research topic [9, 119, 8, 23, 101]. In this thesis, the articles [PI, PII, PIII, PIV, PVI] are related to the K-means++ initialization.

It is well-known that Lloyd's algorithm is guaranteed to converge to a local minimum that is deterministically mapped from the initialization to the final solution [92]. Hence, initialization strategies that explore possible good local minimums are essential (and possibly end up finding the global minimum), because the number of unique local minimums can be large already for small datasets [109]. A poor initialization may cause various undesired effects: there is a higher probability for the final clustering of get stuck in a bad local minimum, a possibility to end up clustering with empty clusters, and the convergence time of the algorithm might be increased [39].

A good initialization strategy can reduce the need for multiple restarts, and if a deterministic initialization approach can be used, restarting is not needed. Concerning the final clustering result for small datasets, when it is possible to do thousands of restarts, the choice of random-based initialization method does not usually matter [39]. However, restarting clustering for large-scale datasets can change the running time requirement for clustering from hours to days or weeks, from reasonable to unreasonable. Therefore, especially for large-scale datasets, initialization for prototype-based clustering matters. Moreover, Steinley [109] noted that by increasing the number of clusters also number of different local minimums also seems to increase. The formula (1) indicates that similar behavior could be possible. Note that the different initializations can end up with the same final solution.

One of the first proposed initialization approaches for K-means are random methods [49, 82]. An initialization method proposed by Forgy [49] assigns all data points randomly to K clusters. A drawback of Forgy's initialization is that the initial centroids computed from the initial partitions cause the centroids to locate very close to each other, which causes bad scalability with respect to K [76]. In [82], MacQueen proposed an initialization method (the second method that is order-invariant) that selects initial prototypes at random from the data points, which is one the most used initialization methods for the K-means. Note that this initialization ensures that there are no empty clusters in the first iteration. MacQueen's initialization method combined with the batch K-means search phase (Lloyd's iterations) is often referred to as the standard K-means. MacQueen's initialization method's main idea is that random selection of the initial prototypes will most often select initial points from the dense regions [39]. On the other hand, this can cause a selection of initial prototypes that are close to each other in the same dense cluster [PVI]. These random methods are not recommended for the initialization, because they often have poor performance [39].

The K-means variant K-means++ [5] utilizes a better initialization technique: it selects initial cluster prototypes so that they are better separated in space than in McQueen's initialization. The K-means++ initialization is depicted in Algorithm 2. After the first prototype is selected uniformly random, the rest of the prototypes are sampled at random, one by one, by utilizing probability distribution

Algorithm 2 K-means++ initialization

```
Input: Dataset $\mathbf{X}$ and #clusters $K$.
Output: Initial prototypes $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$.
1: $\mathbf{c}_1 \leftarrow$ select point uniformly random from $\mathbf{X}$.
2: for $i = 2, i = i + 1, i \leq K$
3:     $\mathbf{c}_i \leftarrow$ select point $\mathbf{x} \in \mathbf{X}$ with probability $\min_{k=1,...,i-1} \|\mathbf{c}_k - \mathbf{x}\|^2 / \text{SSE}(\{\mathbf{c}\}_{j=1}^{i-1})$.
4: end
```

based on squared Euclidean distances with respect to those already selected. In step 3, a probability that a new point is selected is equal to its contribution to SSE divided by the SSE; therefore K-means++ is likely to select a new point from the regions that contribute to SSE significantly, for example, from an isolated cluster. The K-means++ initialization, more or less, balances goals from the MacQueen's and maximin [53] initialization methods: the initial prototypes are more likely to be selected from the dense regions but also favor separation in a selection process. K-means++ has proven guarantees for the final SSE with respect to the optimal solution. In general, K-means++ improves the quality of the clustering results and convergence rate with respect to Forgy's and McQueen's initialization methods. The K-means++ initialization has linear time complexity with respect to N. It is highly desirable that initialization methods for the K-means be linear or superlinear with respect to N, because K-means search is linear with respect to N and it is tricky to justify initialization methods with higher time complexity. A generalized form of the K-means++ initialization algorithm for L_p^q norms is depicted in [PIV].

A major drawback of the K-means++ initialization is that it has an inherently sequential nature [9] while the K-means search can be easily parallelized. The parallelizable version of K-means++, K-means$\|$ or scalable K-means++ [9], samples points with similar fashion, but by sampling $\mathcal{O}(Tl)$ points in T iterations (in practice already 5 iterations with $l = 2K$ is sufficient [9]) that forms a weighted sample that can be clustered with weighted K-means++ (see Algorithm 1 in [8]) to determine the K initial prototypes. This procedure, due to a much smaller number of iterations (the weighted K-means++ procedure can be neglected, because the weighted sample is usually much smaller for large-scale datasets than the whole dataset) compared to $K - 1$ iterations of the K-means++ initialization, enables efficient parallelization. Similarly to K-means++, K-means$\|$ produces provably good clustering results [8]. The computational cost for the K-means$\|$ initialization is larger than it is for the K-means++ initialization, but K-means$\|$ can be parallelized efficiently to enable solving large-scale clustering problems. In the context of big data clustering, a K-means$\|$ implementation is utilized for K-means clustering, for example, in the Spark platform[1].

[1] http://spark.apache.org/

2.2.3 K-spatialmedians

K-means aims to minimize the SSE objective function and because of this, K-means is easily disturbed by outliers in data. Moreover, K-means is also sensitive to missing values in the data [6]. A natural way to handle outliers is to take advantage of a more robust cluster prototype or cluster location estimate that is less prone to be disturbed by noise/outliers than the mean cluster prototype. This treatment allows conducting clustering without outlier detection as a preprocessing step prior to clustering.

The minimization of slightly different objective function than (2) gives the K-spatialmedians clustering problem. The objective function for K-spatialmedians clustering is defined as

$$J(\mathbf{C}) = \sum_{\mathbf{x}\in\mathbf{X}} \min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{c} - \mathbf{x}\|, \tag{3}$$

where $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$ and $\mathbf{c_k} \in \mathbb{R}^M$. Similar to K-means clustering, K-spatialmedians clustering is also based on the general prototype-based clustering algorithm skeleton (Algorithm 1). Both the K-means and K-spatialmedians methods use the Euclidean distance in step 2.1 so the main difference between them is the cluster prototype update in step 2.2. The best prototype for the minimization of a sum of the Euclidean distances, instead of the sum of the squared Euclidean distances (as it is for K-means), of the points within a cluster can be shown to be the spatial median [6], which is a statistically robust location estimate and represents cluster prototypes in K-spatialmedians clustering. The spatial median is also sometimes referred to as Weber point [6]. However, referring to the spatial median is well described, since the spatial median for one-dimensional data gives the same result as median and it is computed by collectively taking account of all dimensions for multidimensional data, while the sample mean and the median location estimate is calculated from the marginal distributions.

Contrary to the sample mean, solving the spatial median requires minimization of a non-smooth optimization (see [71]) problem that requires an iterative method to be computed [68]. The successive over-relaxation (SOR) accelerated iterative algorithm is an efficient method for computing the spatial median for large-scale clustering purposes [68]. Moreover, the number of iterations of the SOR algorithm is independent with respect to the dimension of the data [68, 6].

K-spatialmedians can also a tolerate large amount of missing data [6]. The objective function of K-spatialmedians for data with missing values can be defined by utilizing the available data strategy [99]. The available data strategy refers to an approach which uses all existing values from the data via a set of M dimensional projection vectors (or projection matrix) $\{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}$ where $(\mathbf{p}_i)_j = 1$ if $(\mathbf{x}_i)_j$ has a value or $(\mathbf{p}_i)_j = 0$ if value is missing for $(\mathbf{x}_i)_j$. The objective function (3) with missing values reads as

$$J(\mathbf{C}) = \sum_{i=1}^{N} \min_{\mathbf{c}\in\mathbf{C}} \|\text{Diag}(\mathbf{p}_i)(\mathbf{c} - \mathbf{x}_i)\|, \tag{4}$$

where $\text{Diag}(\mathbf{p}_i)$ transforms projection vector $\mathbf{p}_i$ into $M \times M$ diagonal matrix.

Figure 4 illustrates that K-spatialmedians clustering is more robust than K-means clustering. This figure was published initially in [PIII], where a modified S2 dataset with noise and missing values were used to compare the proposed method (K-spatialmedians||) and K-means||. Note that the final prototypes for the methods were selected out of 200 runs based on their corresponding objective function values with available data strategy.

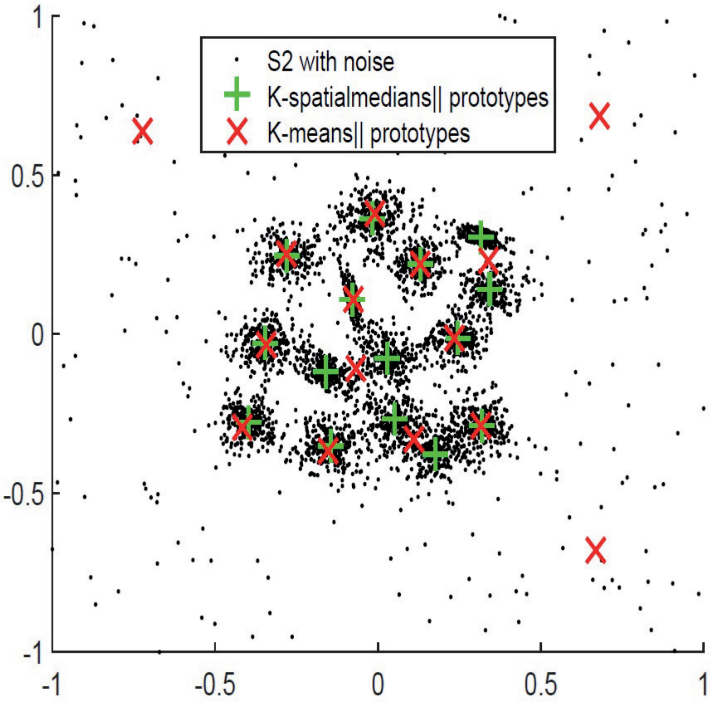


FIGURE 4 Illustration of robustness of K-spatialmedians clustering with respect to K-means clustering for a synthetic dataset with noise and missing values (10%). This figure was originally published in [PIII].

K-spatialmedians can also be employed for large-scale data clustering. However, the spatial median for cluster prototypes requires an iterative method which forms a second iterative layer for the clustering algorithm; therefore, the scalability is more restricted concerning K-means. Nevertheless, K-spatialmedians is an appealing clustering method for large-scale data mining when datasets are sparse and noisy [99]. The article [PIII] proposes an SOR accelerated parallel K-spatialmedians algorithm for large-scale clustering with noise and missing values. In addition, a K-means||-based robust initialization approach is used to initialize K-spatialmedians.

Similar to K-means clustering, the clustering quality and convergence speed of K-spatialmedians depends on the selection of initial prototypes. Äyrämö [6] proposed and tested initialization methods for K-spatialmedians clustering which were based on the well-known K-means initialization methods. In [6], the main idea of the proposed initialization methods was to replace the mean by the spatial median to obtain robust initialization of K-spatialmedians. According to [6] and [PIII], usage of robust elements in the initialization can improve the quality of clustering results.

2.3 Cluster validation

One of the crucial tasks in clustering, in the context of the KDD process, is cluster validation. Clustering results validation is often conducted with clustering validation indices (CVI). A diverse set of CVIs has been developed to validate clustering results [105, 110, 63, 123, 29, 21, 95]. CVIs can be divided into three main groups [98]: internal, external and relative. Internal CVIs do not use prior knowledge from data to suggest an appropriate number of clusters contrary to external CVIs [PIV]. Internal CVIs are more suitable for clustering result validation in a practical sense because in many real-world cases data has no label information, e.g., a knowledge discovery task aiming to find meaningful groupings from data. This can especially occur in big data clustering problems. However, in clustering algorithm comparisons, when the label information is available, external CVIs can be applied to compare the performance of the clustering algorithms of how well they discover the known grouping of data. Relative CVIs are based on comparing clustering structures in cluster validation. In the article [PIV], a comparison of internal CVIs for different distance measures in the context of prototype-based clustering was conducted.

Internal CVI gives suggestions for selecting the number of clusters, which is typically given as input for partitional clustering algorithms. The suggestion is usually based on the ratio of the two values [PIV]: intra (compactness of the clusters) and inter (separability of the clusters). The goal is to minimize intra value (ideal clusters are compact) and maximize inter value (ideal clusters are isolated). Depending on how the ratio is selected, the minimum (when the ratio is intra/inter) or maximum (when the ratio is inter/intra) ratio value defines the internal CVI's suggestion for selecting the number of clusters K.

2.4 Big data clustering

2.4.1 Characteristics of big data

The term "big data" is usually associated with a vast amount of data in terms of the number of observations and variables. This massive amount of data is referred to in big data characterization as volume, which is one of the characteristics presented in [77] along with velocity and variety. Velocity refers to an enormous flow of data that induces challenges to real-time processing. Variety refers to heterogeneous data, data that is typically unstructured and comes in various formats. A fourth characteristic added later, veracity, refers to erroneous and uncertain data (described, e.g., in [33]). However, the three Vs model – volume, velocity, and variety – forms a core characterization of big data [30, 43, 107, 24]. De Mauro et al. [30] define big data as "the information asset characterized by such a high volume, velocity, and variety to require specific technology and analytical methods for its transformation into value."

2.4.2 Scaling clustering methods for big data

The proposed methods in the literature for big data clustering are mainly focused on tackling the first characteristic, volume. In [25], it was reported that [80] contains the most extensive clustering experiments that have been published in the literature and it still seems to be so. Liu et al. [80] clustered about 1.5 billion data points in 104-dimensional space with the distributed nearest neighbor method into 50 million clusters. This clustering task took about 10 hours with 2000 central processing units.

Shirkhorshidi et al. [107] divide big data clustering methods into single machine and multiple machine clustering techniques. Single machine techniques are further divided into sampling-based and dimensionality reduction techniques and multiple machine techniques are further divided into parallel clustering and MapReduce-based clustering. A basic workflow in multiple machine clustering techniques is to divide the dataset into smaller subsets which are distributed to multiple machines. Each machine processes its subset of data in parallel then local results are aggregated to get intermediate clustering results, and after iterating this procedure, the final clustering result is achieved. The main difference between parallel clustering and MapReduce based clustering is that in MapReduce-based clustering data distribution and fault tolerance is handled automatically; thus, parallel clustering creates more challenges for developers [107].

On the other hand, MapReduce does not support efficient iterative data processing [28]. Crotty et al. [27] argue that popular big data frameworks such as Hadoop and Spark are designed for massive data processing (petabytes of data), on large clusters for large companies, such as Facebook and Google, and therefore do not meet the needs of typical users. Typically, large-scale data processing is done with datasets of up to several terabytes on small clusters [27].

Besides K-means having many sequential variants it has also been widely studied in the context of distributed and parallel computing [34, 66, 44, 124, 121, 9, 28, 93, 59]. For big data platforms such as Hadoop[2], Spark [3], Flink[4], and H2O[5], K-means variants are typically implemented in machine learning libraries.

2.4.3 Clustering with random projections

A shift from a low-dimensional data space to high-dimensional space causes various contradictory phenomena for distance measures [2, 113]. Many data mining and machine learning algorithms are based on the concept of the nearest neighbors with the chosen distance measure. Finding the nearest neighbors for high-dimensional data was studied in [11] by Beyer et al. theoretically and empirically. Beyer et al. demonstrated that with certain assumptions about underlying data distributions, the distance to nearest neighbor approaches the distance to the far-

[2] http://hadoop.apache.org/

[3] http://spark.apache.org/

[4] https://flink.apache.org/

[5] http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html

thest neighbor when dimensionality increases, and differences in the distances between data points diminish. These results imply that in high-dimensional space distances can become meaningless [11].

Over the years, various feature extraction methods have been combined with clustering algorithms [17, 75, 47, 4] to handle the curse of dimensionality [113]. Dimensionality reduction has an essential role in data mining, since real data, in many cases, contains a large set of variables that restricts the selection of the methods for data [13]. High-dimensional data come from images, videos and text. Dimensionality reduction aims to transform higher-dimensional data into appropriate lower-dimensional data representation in the way that meaningful information from data is preserved [112]. Dimensionality reduction can be divided into two main branches [16]: feature selection and feature extraction. Feature selection aims to find a subset of meaningful variables from data, and feature extraction aims to extract meaningful variables via projecting data into lower dimensional space.

The main idea to use random projection in dimensionality reduction comes from the Johnson-Lindenstrauss lemma [64], which states that random projection approximately preserves distances when dimensionality reduction is conducted from high-dimensional space to lower-dimensional space. The distance preservation is an appealing characteristic for distance-based clustering methods. The random projection method is combined with clustering methods in several papers [17, 47, 22, 111, 87, 102, 22, 16, 23]. Lately, from the perspective of high-dimensional big data processing, the random projection method has raised interest [107, 111, 114, 87, 102, 103], because it is computationally more efficient compared to traditional methods [13] such as principal component analysis (PCA) [65]. More precisely, the random projection method [1] is dimensionality reduction method that is based on extracting features from the data with an $M \times P$ matrix $\mathbf{R}$ where elements R_{ij} are randomly generated. For one of the following distributions:

$$R_{ij} = \begin{cases} 1, & \text{with probability } 1/2 \\ -1, & \text{with probability } 1/2 \end{cases} \tag{5}$$

$$R_{ij} = \begin{cases} \sqrt{3}, & \text{with probability } 1/6 \\ 0, & \text{with probability } 2/3 \\ -\sqrt{3}, & \text{with probability } 1/6 \end{cases} \tag{6}$$

the random projection reduced $N \times P$ data matrix $\widetilde{\mathbf{X}}$ can be computed as

$$\widetilde{\mathbf{X}} = \frac{1}{\sqrt{P}}\mathbf{X}\mathbf{R}. \tag{7}$$

Utilization of the distribution in (6) is referred to as sparse random projections, which can give a threefold speedup because multiplication with $\sqrt{3}$ can be done after computing $\mathbf{XR}$ without using floating-point arithmetic [79]. There are more suggestions to generate $\mathbf{R}$ in [79].

2.4.4 Parallel clustering with MATLAB

Distributing a dataset horizontally (observations) or vertically (features) to multiple machines' distributed main memory, followed by parallel clustering, is a natural way to scale clustering methods for large-scale datasets that do not fit a single machine's main memory. In the MATLAB environment, this can be implemented with the single program multiple data[6] (SPMD) paradigm by utilizing parallel computing toolbox[7] (PCT) (see [106]). This enables scaling of clustering methods for large-scale datasets via MATLAB distributed computing server[8] (MDCS). A basic workflow is first to develop a parallel implementation on a local desktop computer with PCT and then scale it to a computer cluster with MDCS, which is responsible for job scheduling. During an execution of a batch job in MDCS, one machine runs as a client (master worker) and one or more machines run worker processes (In MDCS, these are also referred to as labs). In the context of SPMD, a program running (single program) on the client controls which workers perform *spmd*-block statement(s) on their local data (multiple data). The workers can communicate with each other using MATLAB's message passing functions, which are based on the well-known message passing interface standard.

[6] http://se.mathworks.com/help/distcomp/spmd.html

[7] http://se.mathworks.com/help/distcomp/index.html

[8] http://se.mathworks.com/help/mdce/index.html

3 SUPERVISED LEARNING

In supervised learning, the aim is to give a prediction for a given input based on a set of inputs and their corresponding known output values. Contrary to unsupervised learning, the supervised learning process is guided by desired output values, and the performance of the model is determined by the training accuracy (performance in the training set) and generalization accuracy (performance in the unseen test set). Supervised learning tasks can be divided into two main categories, regression and classification, based on the output type, quantitative and qualitative [58]. In this chapter, cross-validation approaches are first introduced in Section 3.1. Supervised learning methods are described in Section 3.2. Finally, Section 3.3 describes the minimal learning machine and concerns reference points selection.

3.1 Cross-validation

A typical approach is to include model validation/selection for separate validation data that involves tuning parameters to optimize generalization ability of the model. The most common approaches for validation are based on cross-validation in which the dataset is first divided into J disjoint groups. Then the learning algorithm is applied to $J-1$ groups (training data) and validated with one group (validation data). This procedure is repeated so that each group is once allocated as a validation data and $J-1$ times as learning data. As a result, this procedure gives an estimate of the learning algorithm's generalization performance (e.g., in classification, it may be classification accuracy). Cross-validation strategies enable full data usage in the learning process, and the repetition of the learning and validation increases reliability. Note that repeating the entire cross-validation with different groups/folds gives better estimates [73].

Cross-validation approaches differ in how they create the disjoint groups of data. In the conventional cross-validation approach, k-fold cross-validation, validation creates k approximately equal-sized subsets of data at random [73]. In leave-one-out cross-validation, k-fold cross validation is performed for $k = N$. In classification problems, k-fold stratified-cross validation (SCV) aims to keep a balance of different classes between folds so that each fold has an approximately similar distribution of classes to the entire dataset. The SCV is the most popularly applied cross-validation method in the literature [86].

In [120], Zeng and Martinez proposed a more advanced strategy for cross-validation, the distribution balanced stratified cross-validation (DB-SCV). The DB-SCV extends SCV to approximately balance the class-wise distribution of the inputs. The DB-SCV aims to keep the distribution of the input data similar within the folds [86]. The distribution optimally balanced stratified cross-validation (DOB-SCV) method improves DB-SCV with more careful placement of the data points into folds. The DOB-SCV is typically employed with classification tasks; however, it can also be used to create distributionally balanced folds for regression tasks by treating the whole dataset as one-class [PVI]. This enables data distribution to remain approximately similar between the folds for regression.

Moreover, DOB-SCV, as one-class case, can be utilized to create similar subsets of the data for divide-and-conquer type of approaches, as is done in [PI]. Based on the DOB-SCV algorithm presentation in [67], the DOB-SCV algorithm for the one-class case is depicted in Algorithm 3. Different from DB-SCV, in step 4, the randomly selected point is always assigned to fold $\mathbf{F}_1$ and ith nearest neighbor is assigned to fold $\mathbf{F}_{i+1}$.

Algorithm 3 One-class DOB-SCV

```
Input: Dataset $\mathbf{X}$ and the number of folds $k$.
Output: Folds $\mathbf{F}_1, \mathbf{F}_2, ..., \mathbf{F}_k$.
1: while $\mathbf{X}$ has more than $k$ points do
2:     $\mathbf{e}_1 \leftarrow$ select a point uniformly at random from $\mathbf{X}$.
3:     $\{\mathbf{e}_2, ..., \mathbf{e}_k\} \leftarrow$ find $k-1$ nearest neighbors of $\mathbf{e}_1$ from $\mathbf{X}$.
4:     $\mathbf{F}_i \leftarrow \mathbf{F}_i \cup \mathbf{e}_i$ for each $i = \{1, 2, ..., k\}$.
5:     $\mathbf{X} \leftarrow \mathbf{X} \setminus \{\mathbf{e}_1, ..., \mathbf{e}_k\}$.
6: Set rest of the points from $\mathbf{X}$ into different folds.
```

3.2 Supervised methods

K-nearest neighbors (K-NN) [116] is one of the simplest supervised methods for classification. A given point is classified based on the K closest neighbors' labels. Typically, it is determined by majority voting, the most common class label among the neighbors. Majority voting is usually sensitive for the selection of K; a better voting technique is weighted voting based on distance [117]. Besides its simplicity, K-NN is also one of the state-of-art methods in unsupervised anomaly detection, especially in global outlier detection [51].

In the context of deep learning, feedforward neural networks, such as multilayer perceptron (MPL) neural networks, are typically employed for supervised learning problems where internal structure of the data lies in high-dimensional spaces, which is the case in many areas of science [78]. The MLP conducts prediction for a given input by propagating the input through a hierarchical structure of few layers where non-linear functions (activation functions) and layer connections (weights) aggregate the output. The most popular approach for the training of feedforward neural networks is backpropagation, where the synaptic weights are modified with gradient descent by propagating from the output layer through the hidden layer(s) to minimize error with respect to desired outputs. In deep neural networks, neural network architecture contains multiple layers which allow the formation of multiple abstraction levels of the data, thus enabling raw data processing [78].

The extreme learning machine (ELM) [60] is a randomized feedforward neural network. It is an efficient and straightforward supervised method for regression and classification. Differently from other feedforward neural networks, the ELM contains only single hidden layer, and the hidden layer nodes are randomly initialized and fixed prior to training. Moreover, it contains only one metaparameter, the weights between the hidden layer and the output layer, that must be optimized during the training.

3.3 Minimal learning machine

The minimal learning machine (MLM) was proposed in [32]. Similar to the ELM, the MLM is a simple randomized learning machine that requires only one hyperparameter to be tuned. The MLM consist of two steps [31]: 1) construction of a linear regression model between the distance matrices and 2) output estimation of an input point based on the distances the reference points in the input space and estimated distances based on the regression model in the output space. Although MLM is based on using linear regression, it is able to form nonlinear regression models for multidimensional outputs. The MLM training step 1 can be broken down into two substeps [85]: 1.1) selection of the reference points and 1.2) linear regression model fitting for the distance matrices. The actual output value estimation based on the estimated distances in the output space can be treated as a multilateration optimization problem, which can be solved with several approaches [31].

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ input data and $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$ corresponding output data where $\mathbf{x}_i \in \mathbb{R}^M$ and $\mathbf{y}_i \in \mathbb{R}^S$. Let $\mathbf{R} = \{\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_K\}$ be a subset of points, reference points, from $\mathbf{X}$ so that $\mathbf{r_i} \in \mathbb{R}^M$. Similarly for the output space, let $\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, ..., \mathbf{t}_K\}$ be a subset of points from $\mathbf{Y}$ so that $\mathbf{t_i} \in \mathbb{R}^S$. Then, the learning step 1) of the MLM finds coefficients for the following linear regression model [31]

$$\mathit{\Delta}_{\mathbf{y}} = \mathbf{D}_{\mathbf{x}}\mathbf{B} + \mathbf{E}, \tag{8}$$

where $\mathbf{D_x}$ is $N \times K$ input distance matrix, $\Delta_\mathbf{y}$ is $N \times K$ corresponding output distance matrix, $\mathbf{B}$ is $K \times K$ matrix containing regression model coefficients and $\mathbf{E}$ is $N \times K$ matrix that contains regression residuals. A matrix element $D_{ij} = \|\mathbf{x}_i - \mathbf{r}_j\|$ of $\mathbf{D_x}$ for $i = 1, ..., N$ and $j = 1, ..., K$. Correspondingly, a matrix element $\Delta_{ij} = \|\mathbf{y}_i - \mathbf{t}_j\|$ of $\Delta_\mathbf{y}$ for $i = 1, ..., N$ and $j = 1, ..., K$. Note that linear relation between input and output distance matrices is assumed in the MLM. The least squares estimate of $\mathbf{B}$

$$\hat{\mathbf{B}} = \begin{cases} (\mathbf{D_x}^T\mathbf{D_x})^{-1}\mathbf{D_x}^T\Delta_\mathbf{y} \text{ with } N < K, \\ \mathbf{D_x}^{-1}\Delta_\mathbf{y} \text{ with } N = K, \end{cases} \tag{9}$$

can be used in the linear regression model 8. In output estimation for new data point $\mathbf{x} \in \mathbb{R}^M$, distances with respect to the reference points $\mathbf{R}$ are first computed as $\mathbf{d_x} = \|\mathbf{x} - \mathbf{r}_1\|, ..., \|\mathbf{x} - \mathbf{r}_K\|$. Then the linear regression model of the two distance matrices is used to compute estimated distances

$$\hat{\delta} = \mathbf{d_x}\hat{\mathbf{B}}. \tag{10}$$

Finally, actual output value, $\mathbf{y}$, for $\mathbf{x}$ is given by a solution of the following objective function minimization

$$J(\mathbf{y}) = \sum_{i=1}^{K}(\|\mathbf{y} - \mathbf{t}_i\|^2 - \hat{\delta}_i^{\,2})^2. \tag{11}$$

For an example, Levenberg-Marquadt [31] or the classical Newton's method [PV] can be employed for solving minimization of equation 11. The objective function minimization of equation 11 is also referred to as multilateration problem [31].

The MLM can be easily adapted to classification problems by representing output vectors with binary variables [85]. For example, output vector $\mathbf{y}_i$ corresponding to the jth class has components otherwise 0, but jth component is 1. Classification of a new input $\mathbf{x}$ into one of the S classes, i.e., assigning class label $l \in \{G_1, ..., G_S\}$, is determined by the predicted output vector $\mathbf{y}$. The predicted class label is G_{s^*}, where s^* corresponds to the largest component of $\mathbf{y}$.

The nearest neighbor minimal learning machine (NN-MLM) [85] utilizes straightforward labeling strategy. NN-MLM assigns class labels based on the class label of the nearest reference point. The nearest reference point is determined by the predicted distances $\hat{\delta}$, thus avoiding the minimization of equation 11. Cubic equation minimal learning machine (C-MLM) [85] can be used for regression problems with one-dimensional outputs. In C-MLM, an assumption of one-dimensional outputs simplifies the objective function 11 so that $\mathbf{y}$ can be solved analytically. This requires solving a cubic equation that always has at least one real root when all coefficients are real.

The training phase of the MLM has time complexity order of $\mathcal{O}(NK^2)$, which is similar to the ELM training if K is the same as the number of hidden layer nodes [31]. In its original formulation, computing prediction with the MLM for a given input is slightly more expensive compared to the ELM [31]. However, the MLM variants proposed in [85], NN-MLM and C-MLM, provide a faster prediction for the classification problems and one-dimensional regression problems.

Currently, different distance measures and methods for the reference points selection are topical directions to extend and improve the MLM. Distances in the MLM are typically measured with the Euclidean distance. However, other distance measures can also be used [69, 96]. Dias et al. [35] proposed the reference points selection method for the MLM classification model building based on the opposite neighbors (ON) method. The main idea of the ON method is to exclude a subset of points from the data that lie in class-overlapping zones. Finally, reference points are selected randomly out of the class-overlapping zones from a subset of points. The ON method applies K-NN for building the subset of points from the class-overlapping zones. This idea is further improved with fuzzy clustering for classification tasks in [48].

In the article [PVI], it was demonstrated that clustering-based methods could be beneficial for the MLM in regression tasks. Figures 5–7 demonstrate and illustrate differences of the reference points selection methods studied in [PVI] for the MLM in non-linear regression tasks. An illustration from the results of [PVI] for the clustering-based reference points selection methods for a non-linear regression task (modification of the S1 [50] dataset) is shown in Figure 5, where the root-mean-squared error (RMSE) for a test set is plotted as a function of the number of reference points. The RS-maximin method has the best RMSE curve for this dataset, and the Random method has the worst. The difference is larger between the methods when the number of reference points is small.

Figure 7 shows the predicted output (z-axis) as a function of two-dimensional input. The MLM was trained with the entire S1 regression dataset (see [PV] for a description of the dataset) with $K = 50$, and then predictions were computed with a two-dimensional grid. In Figure 6c, the ground truth output is shown with respect to this grid. By comparing prediction surfaces for the Random (Figure 6a) and RS-maximin (Figure 6b) built MLM regression models, one can observe that randomly selected (Random) reference points create more distortions to output surface than well-separated reference points (RS-maximin).

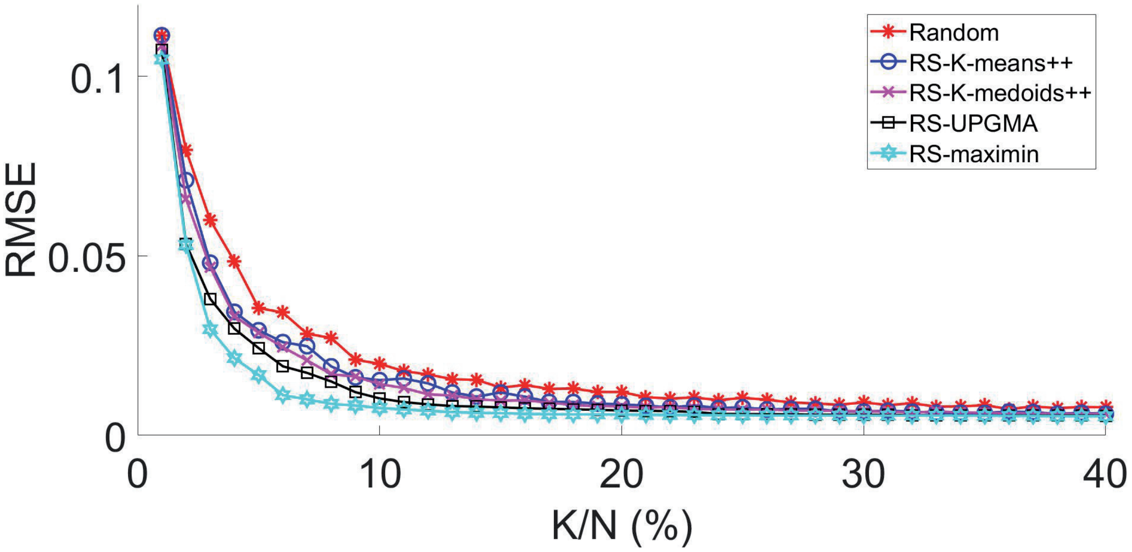
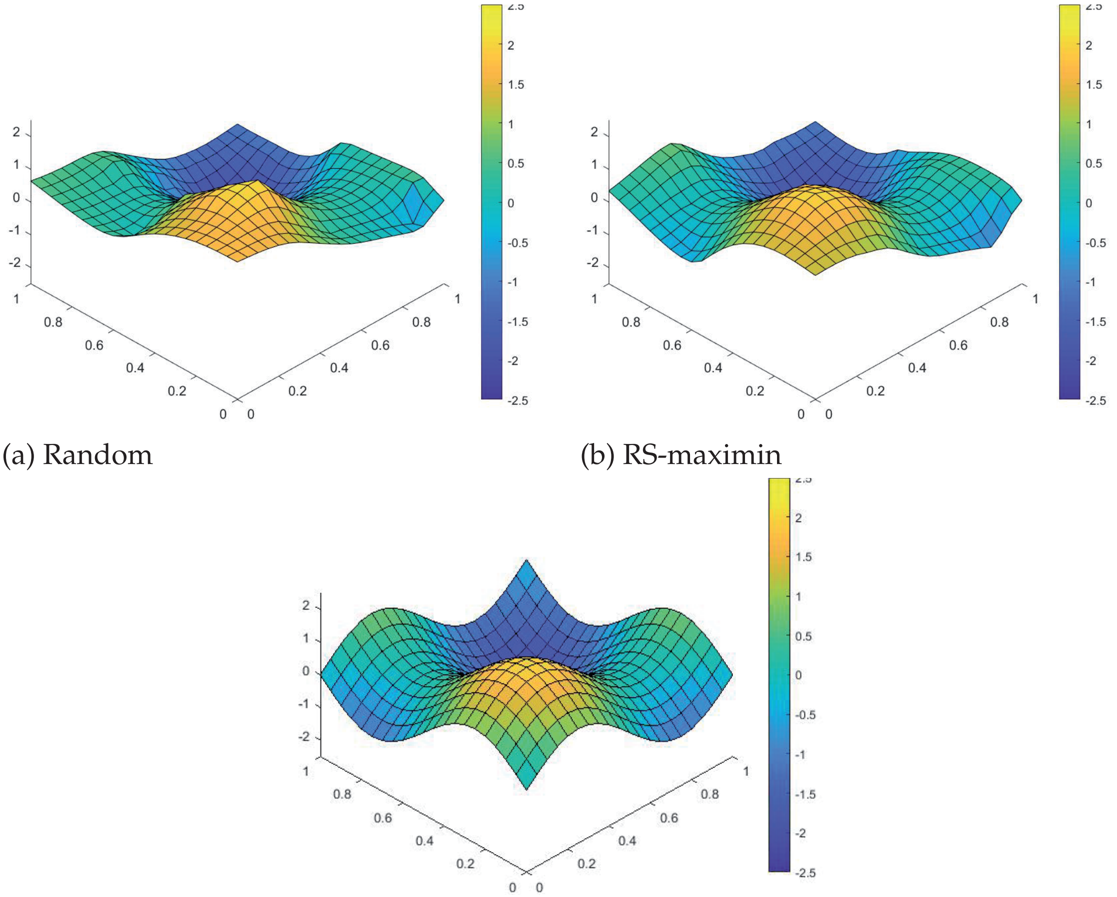


FIGURE 5 Demonstration of differences between the clustering-based methods [PVI] and the random selection concerning a regression model quality (RMSE), and model complexity (number of reference points *K*). For a small number of reference points, it matters how the reference points are selected. The results shown here were computed in [PVI].

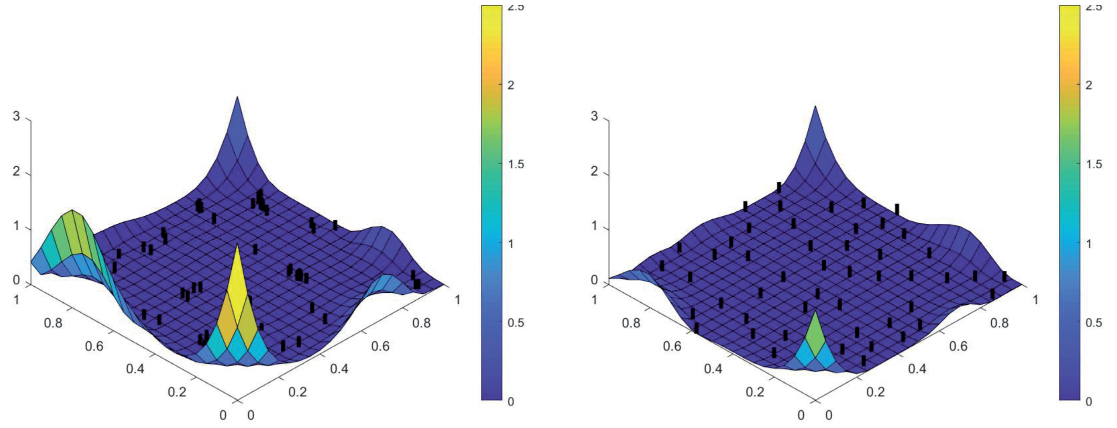


(a) Random

(b) RS-maximin

(c) Ground truth

FIGURE 6 Prediction surface of the MLM for a synthetic regression dataset with (a) Random and (b) RS-maximin reference points selection methods. The ground truth output surface is shown in (c).



(a) Random

(b) RS-maximin

FIGURE 7 Squared error surface of MLM with respect to the ground truth for (a) Random and (b) RS-maximin reference points selection methods. Reference point locations in the input space are indicated with black bars.

4 SUMMARY OF THE INCLUDED ARTICLES

4.1 [PI]: Initialization of big data clustering using distributionally balanced folding

The article [PI] was published in the proceedings of the 24th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning in 2016.

This article proposed a clustering initialization strategy for the K-means clustering method. As already discussed in Section 2.1, the K-means clustering method is highly sensitive for the initial selection of the prototypes. This article proposed a method that combines ideas from two known effective initialization strategies and an advanced sampling approach. The proposed approach is based on using distributionally optimally balanced stratified cross-validation (DOB-SCV) created distinct subsets of data which are clustered with the K-means++ method. Finally, the initialization strategy selects prototypes corresponding to the smallest SSE as an initialization. Note that in practice the DOB-SCV folding is required to be performed only once for each dataset.

The proposed method was implemented in parallel and experimented with a real dataset that was artificially extended to mimic the volume of a big data case. A parallel SPMD MATLAB implementation of the proposed method was tested utilizing MDCS. For DOB-SCV, sequential MATLAB implementation was used. Based on the experiments, the clustering quality for the proposed initialization method was around the same level as for *R* times repeated K-means++, when *R* was the same as the number of folds in the proposal. The main benefits of the proposed initialization method, compared to the inherently sequential K-means++, are that the proposed method can be easily implemented in parallel with several parallel computation models and it scales for large-scale data.

4.2 [PII]: Scalable initialization methods for clustering large datasets

The article [PII] has been submitted to Pattern Recognition Letters (in revision).

The goal of this article was to develop efficient parallel clustering initialization methods for large datasets based on K-means|| type of divide-and-conquer approach and dimensionality reduction. Compared to the method proposed in [PI], the computational cost is reduced by using random subsets, approximative SSE evaluation, a limited number of subset clustering iterations and an efficient dimensionality reduction method. For the dimensionality reduction, the random projection method was selected, because it is computationally fast, e.g., compared to PCA. Moreover, the computational cost for the K-means|| initialization mainly accrues from the distance computations. Utilizing K-means|| instead of K-means++ for clustering subsets enables fixing the number subsets. To illustrate for eight subsets, parallel implementation using 64 processing cores can be implemented so that each subset is clustered in parallel with eight processing cores instead of 1 processing core. Thus, the local SSE (clustering error for subset) based selection of the initial prototypes can be used to speed up SSE computation compared to the approach proposed in the article [PI].

In this article, two parallel K-means initialization methods (SK-means|| and SRPK-means||) were proposed for large-scale datasets. Both of the proposed methods are based on clustering subsets with K-means||, but differ in that the SRPK-means|| method utilizes random projections in the initialization. The proposed methods were compared to K-means++ and K-means||. Experiments were conducted with 15 datasets of various sizes. The seven largest datasets were clustered with parallel implementations. Parallel MATLAB implementations of the proposed methods and K-means|| were implemented with SPMD blocks and message passing functions. The parallel implementations were tested utilizing MDCS and Taito computing cluster.

In general, both proposals achieved clearly smaller initialization clustering errors than K-means++ and K-means||, and the final clustering error was better or equal compared to K-means++ and K-means||. In addition, the proposed methods usually required a smaller number of clustering iterations until convergence in the search phase than K-means++ and K-means||. The running time of K-means|| was around $60\% - 80\%$ of the running time of SK-means||, but this is more than compensated with faster convergence and better clustering quality. Overall, the proposals performed well in the scalability tests compared to K-means||. The scalability experiments for a large and high-dimensional dataset showed that SRPK-means|| can perform initialization up to 7–8 times faster than K-means||. For very high-dimensional data, the speedup was improved when K was increases for SRPK-means||.

According to the experimental results, SK-means|| was better suited for datasets with dimension less than 100 and SRPK-means|| for datasets with dimension more than 100. SRPK-means was clearly faster than SK-means|| when dimension is higher than 100. Furthermore, the final clustering error statistics also favored SRPK-means|| compared to SK-means|| for datasets with dimension larger than 100. The clustering quality of the proposed methods was already quite good after the initialization. Therefore, the proposed methods could also be used as standalone clustering algorithms for large-scale datasets.

4.3 [PIII]: Scalable robust clustering method for large and sparse data

The article [PIII] was published in the proceedings of the 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning in 2018.

Clustering datasets which are large, sparse and contain noise/outliers require specialized methods. K-spatialmedians uses available data strategy to handle missing values in data. The available data strategy considers all existing values from the data without discarding anything. This approach does not make any assumptions about the missing values for clustering. Unlike K-means that uses mean as a location estimate for the cluster prototype, K-spatialmedians uses spatial median, which is more tolerant for missing values and outliers than mean. Employing K-spatialmedians for large-scale clustering tasks with parallel computing is not trivial because the computation of the location estimate requires an iterative method to be solved. In this article, the primary goal was to tackle the K-spatialmedians clustering problem for large-scale data via parallel computing.

This article proposed a parallel implementation of K-spatialmedians, K-spatialmedians||, for datasets that are sparse, noisy and large. Secondly, a robust initialization strategy, based on integrating K-means|| initialization and weighted K-spatialmedians, that can also be parallelized, was described. The empirical comparison for a synthetic dataset with missing values and noise showed that the proposed method outperforms the K-means|| in terms of clustering quality. Similar to articles [PI] and [PII], parallelization of the proposal was implemented with MATLAB SPMD blocks and message passing functions, and it was tested via MDCS. The scalability experiments for a large real dataset ($N = 16,334,970$, $M = 128$) with missing values show that the implementation scales well. Thus, it was demonstrated that the K-spatialmedians could be adapted to large-scale parallel clustering purposes in a computer cluster environment.

4.4 [PIV]: Comparison of internal clustering validation indices for prototype-based clustering

The article [PIV] was published in Algorithms journal in 2017.

In the article [PIV], the main goal was to compare clustering validation indices' enlargements of three popular L_p^q norms (the Euclidean, the squared Euclidean and city-block) with their corresponding prototype-based clustering methods (K-spatialmedians, K-means, K-medians). Same distance measures of the clustering methods' objective functions were used to compute the indices correspondingly. For example, for the K-means clustering result, the squared Euclidean distance was used to compute the indices. In addition to clustering validation index comparison, convergence properties of the prototype-based clustering methods were studied experimentally and theoretically.

Based on the extensive empirical evaluation of the internal clustering validation indices for 56 synthetic datasets, on the one hand, the WG index outperforms other indices for all three distance metrics. Therefore, the WG index appears to be an appealing index for further study. On the other hand, the WG index is not the best for all types of data. For the sim5D10 and sim5D2 datasets, which have unbalanced clusters with noise, the WG index failed to suggest the correct number of clusters, while the PBM index (with the Euclidean and city-block distances), the KCE index (with the squared Euclidean distance) and the WB index (with the squared Euclidean distance) all succeeded in these tasks. The article [PIV] further confirms previous studies findings that there exists no superior internal clustering index that adapts for all types of data.

Concerning different distance measures, it was discovered that some indices are better suited for specific distances. For example, the PBM index is better suited for the Euclidean and city-block distances than for the squared Euclidean distance. Also, the WB index seems to perform better with robust distance measures. Based on the results, the DB and RT indices are not recommended for cluster validation since there exist clearly better performing CVIs with a reasonable computational complexity with respect to prototype-based clustering. The difference to other CVIs is especially clear when the number of clusters increases. Cluster indices often suggest quite a small number of clusters for high-dimensional data. Hence, utilization of prototype-based clustering in a hierarchical fashion is recommended for discovering clustering structures at different levels. Moreover, prototype-based clustering methods that use the batch-type update of the prototypes have a theoretical basis for convergence as shown in [PIV], in which Algorithm 1 decreases the clustering error in each iteration until convergence. The most distinct difference between the clustering methods in the experiments, in terms of convergence, was that K-medians tended to require more iterations for convergence than K-means and K-spatialmedians when dimension increased.

4.5 [PV]: Feature ranking of large, robust, and weighted clustering result

The article [PV] was published in the proceedings of 21st Pacific-Asia Conference on Knowledge Discovery and Data Mining in 2017.

In this article, a general aim was to automate clustering result interpretation for large and weighted clustering result which can contain a significant amount of missing values. In [100], large-scale international student assessment data (PISA 2012 dataset) was clustered with hierarchical K-spatialmedians variant [115]. The article [PV] continued this study by interpreting clustering result labels concerning input data (data used in clustering) and additional data (data not used in clustering) achieved in [100]. More specifically, the aim was to determine variable ranking in terms of how well they explain the clustering structure.

Two approaches based on the Kruskal-Wallis H test were proposed for automating the interpretation of the student assessment data clustering result. The first proposal utilizes the Kruskal-Wallis H test statistic with the bootstrapping strategy (Bootstrap KW). The second proposal is an analytical formula derived from the Kruskal-Wallis H test statistic (Analytic KW). The proposed approaches can process real value-weighted clustering result. In addition to the PISA 2012 clustering result, the classical iris dataset was artificially modified with real value weights, and it was used to demonstrate the proposals. Bootstrap KW was implemented in parallel with Matlab PCT. Based on the experiments, both methods were found to have a highly similar ranking of variables.

Moreover, the results are consistent concerning the analysis conducted in [100]. Analytic KW is suitable for fast ranking of variables. The Bootstrap KW approach takes more information into account when forming the ranking, and it can produce confidence intervals for the ranking. The Bootstrap KW approach uses more computing resources than the Analytic KW approach, but this can be compensated for by parallelizing the Bootstrap KW approach vertically (with respect to variables).

Variables that were used in the clustering explain the clustering structure generally more than additional variables. However, some of the additional variables had fairly high rankings. The highest ranked variable, according to both methods, was economic, social and cultural status (ESCS) of the students.

4.6 [PVI]: Clustering-based reference points selection for the minimal learning machine

The MLM is based on mapping input and output distance matrices, which can be used to built regression and classification models. The distance matrices are formed using a subset of points, referred to as reference points. The default technique is to select these reference points at random. The goal of the article [PVI] was to improve the MLM model by applying clustering-based reference points selection instead of the random selection method. The focus here was to build regression models with the MLM.

In this article, four modifications of clustering methods for reference points selection were proposed. A joint procedure of the methods is to carry out clustering or clustering initialization in the input space to form a subset of points. Then corresponding points are selected from the output space. Based on the empirical evaluation with 13 regression datasets, the proposed methods can improve the regression model RMSE values compared to random selection. This was especially noticeable for a small/moderate number of reference points. In overall, the best performing reference selection method was RS-maximin. This method is based on the maximin K-means initialization method. The reference point selection method RS-UPGMA, achieved similar RMSE values as RS-maximin. However, the computational complexity for RS-UPGMA is $\mathcal{O}(N^2)$ due to hierarchical clustering whereas RS-maximin has linear time complexity with respect to N. Note that these deterministic methods must be applied only once for each dataset because they construct the set of reference points incrementally. In addition, a new initialization heuristic for Newton's method was proposed. Newton's method was used to solve the multilateration optimization problem. Based on the results, the initialization heuristic seems to be a valid approach for the initialization. The random method selects the reference points so that they have higher probabilities of being located near each other. It seems that selected reference points should be well separated from each other, which is the way RS-maximin constructs a set of reference points.

4.7 Summary of contributions

A summary of the contributions and the articles' relationship to the research questions (given in Section 1.2) are shown in Table 1. This thesis proposed new parallelizable initialization methods in the articles [PI,PII]. The proposed initialization methods are scalable, and they are intended for large-scale clustering purposes. In the article [PII], the second proposal is an efficient method for high-dimensional and large-scale clustering. Moreover, the initialization method for the K-spatialmedians clustering, described in [PIII], is a promising approach for large-scale datasets that contain noise/outliers. In the article [PVI], the maximin K-means initialization-based proposal was the best performing reference points selection method for the MLM, even though maximin is not a recommended method in the literature for initializing K-means.

In addition to the parallelizable initialization methods, parallelization of the K-spatialmedians was described in [PIII]. The proposed method was found to achieve good clustering results and scale for large-scale data. Moreover, it can handle both missing values with the available data strategy and noise/outliers with a robust cluster location estimate during the clustering procedure. Article [PIV] compared internal CVIs which were generalized for the L_p norms of the q-th power to correspond with different prototype clustering objective functions. Some of the indices performed quite differently with different distance measures. Article [PV] proposed two approaches for ranking important variables that explain a given clustering structure. These approaches automate result interpretation for large and weighted clustering result. Article [PVI] proposed clustering-based reference points selection for the MLM in the regression tasks. It was found that clustering-based methods can improve MLM regression models concerning accuracy and model complexity.

TABLE 1 Summary of contributions.

P	RQ	Contribution
[PI]	RQ1	1) Proposed K-means++-based scalable large-scale clustering initialization method; 2) Showed that the combination of divide-and-conquer type of K-means++ strategy and the smallest SSE-based selection could be beneficial for initializing the K-means clustering; 3) The proposed method utilizes subsets created by DOB-SCV
[PII]	RQ1	1) Proposed K-means‖-based scalable large-scale clustering initialization method; 2) Proposed K-means‖-based scalable and efficient large-scale clustering initialization for high-dimensional data; 3) Showed that random projections could be beneficial for K-means‖; 4) The proposed methods reduce clustering error compared to K-means‖; 5) The proposed initialization methods can be used as standalone to obtain approximate clustering solution
[PIII]	RQ1, RQ2	1) Parallelization of K-spatialmedians for large-scale, robust clustering; 2) Scalable, robust initialization method for K-spatialmedians utilizes a modification of K-means‖; 3) The proposed method outperforms K-means‖ in terms of clustering quality for noisy and sparse data
[PIV]	RQ1, RQ3	1) Described internal CVIs' enlargements for the L_p^q norms and conducted a comparison of them; 2) Some CVIs' performance depended on what distance measure was used; 3) The WG index was the best performing index in the comparison; 4) The DB and RT indices are not recommended for clustering validation; 5) Theoretical and experimental study of the convergence properties of prototype-based clustering; 6) Described general K-means++ initialization for the L_p^q norms
[PV]	RQ1	1) Proposed parallelizable feature ranking approach for large-scale weighted clustering result based on Kruskal-Wallis H test and bootstrapping; 2) Proposed analytical feature ranking formula for large-scale weighted clustering result based on Kruskal-Wallis H test; 3) Showed that economic, social and cultural status (ESCS) of students is the best explaining feature for the PISA clustering result in [100]

Continued on next page

Table 1 – *Continued from previous page*

P	RQ	Contribution
[PVI]	RQ4	1) Proposed clustering-based reference points selection methods for the MLM; 2) Clustering-based methods are valid alternatives for the random selection of reference points; 3) Linearly scaling RS-maximin is the best performing method for reference points selection; 4) Newton's method with a new initialization heuristic was utilized to solve the multilateration optimization problem; 5) Demonstrated that well separated reference points improve MLM in regression tasks

4.8 Author's contribution to the included articles

Origins of the new methods

The proposed method in [PI] originated from Tommi Kärkkäinen as well as the idea to utilize dimensionality reduction in [PII] otherwise, ideas regarding the proposed methods, SK-means|| and SRPK-means|| in [PII] originate from the author. The idea to adapt K-spatialmedians with parallel computing for large-scale clustering came from Tommi Kärkkäinen. The K-spatialmedians|| algorithm in [PIII] is given by the author. The initialization strategy of K-spatialmedians in [PIII] originated from the author. The idea to enlarge K-means++ initialization and the internal CVIs in [PIV] to L_p^q distances was Tommi Kärkkäinen's. The proposed methods in [PV] originated from Tommi Kärkkäinen. The utilization of clustering methods and Newton's method with new initialization heuristics in MLM in [PVI] should be credited to Tommi Kärkkäinen. The idea to employ UPGMA and maximin methods in reference points selection in [PVI] came from the author.

Implementations

All parallel implementations of the methods in [PI, PII, PIII, PV] and all implementations in [PII] and [PIII] were written by the author. An SOR algorithm parallel implementation in [PIII] was written by the author based on a sequential implementation of the SOR made by Tommi Kärkkäinen. In [PIV], implementations were mainly written by Tommi Kärkkäinen and the author. In [PVI], the author wrote a majority of the clustering-based reference points method implementations and made minor modifications to the MLM implementation.

Experimental design and carrying out experiments

The author and Tommi Kärkkäinen mainly designed the experiments in the articles [PI, PII, PIII]. The experimental design in [PIV, PV] was conducted by Tommi Kärkkäinen. The experiments in [PVI] were designed by all authors. The author carried out the experiments in the articles [PI, PII, PIII, PV, PVI]. In [PIV], experiments were performed by the author and Susanne Jauhiainen.

Writing the articles

In the articles [PII, PIII, PVI], the author’s contribution to writing in each article was major. In [PI, PIV], the authors’ contributions to writing were close to equal. In [PV] the author wrote a majority of Section 4. Moreover, the author was the corresponding author in the articles [PI, PII, PIII, PIV, PVI].

5 CONCLUSIONS AND FUTURE WORK

This dissertation is composed of six articles, which are related to the essential aspects of prototype-based clustering. Using the proposed modifications and improvements allows one to improve prototype-based clustering algorithms and, therefore, the whole KDD process. The general goal was to advance KDD process towards more scalable, efficient and reliable data processing via contributions to prototype-based clustering. This objective is motivated by modern demands of the KDD process arising from big data and the popularity of prototype-based clustering methods such as K-means. Articles [PI], [PII] and [PIII] improve prototype-based clustering methods related to initialization and scalability. Clustering result interpretation approaches for weighted and large-scale clustering results in [PV] makes an effort to automate the cluster analysis chain further. The article [PIV] gives insights into internal CVIs with different distance measures in the context of prototype-based clustering. In the article [PVI], clustering methods are applied to improve regression model building in the MLM.

Based on the articles [PI] and [PII], a divide-and-conquer type of strategy combined with SSE-based selection of the initial prototypes can be beneficial for prototype-based clustering initialization for large-scale clustering purposes. The article [PI] is a proof of concept for this idea, and the proposed parallelizable initialization method utilizes K-means++. In [PII] the idea is further developed concerning efficiency and scalability. Efficiency and scalability of the two proposed parallelizable initialization methods, SK-means|| and SRPK-means||, proposed in [PII] arise from the K-means|| method, divide-and-conquer type of strategy, random projections, approximative SSE evaluation and tightly limited subset clustering iterations. Clustering quality for this type of strategy is equal or better than state-of-the-art for parallel K-means initialization. Moreover, SK-means|| and SRPK-means|| clearly achieve better clustering results in the initialization than K-means||, which demonstrates that these clustering initialization methods can be utilized as standalone to get approximate clustering results for large-scale data.

The parallelization of K-spatialmedians in [PIII] provides an enlargement of K-spatialmedians clustering to large-scale datasets that contain noise/outliers and missing values. Moreover, the initialization method variant of K-means|| for K-spatialmedians described in [PIII] is an appealing approach. Based on the articles [PIII] and [PIV], it seems that it is a good practice to modify the K-means initialization methods to correspond to the same distance measure that the prototype-based clustering method uses. This could be understood in the robust initialization of K-spatialmedians [PIII] that using the Euclidean distance-based sampling in K-means|| and then weighted K-spatialmedians aim to give a rough estimation of the same objective function minimization than the refinement phase of K-spatialmedians. In future work, combining ideas from [PII] and available data strategy could further improve the robust initialization for the K-spatialmedians for large-scale data.

In the article [PIV], comparison of the internal CVIs' extensions to three different distance measures and their corresponding prototype-based clustering methods shows that when a clustering chain including initialization, refinement of the initial prototypes and cluster validation (based on internal CVIs) utilizes a fixed distance measure, some indices are better suited for different distances. In practice, this means that selection of distance measures for internal CVIs should be considered in the clustering procedure. Even though the WG index had an excellent performance for many of the synthetic datasets, the results of the article [PIV] reinforce previous studies' observations about CVIs that no superior CVI for all types of data exists. The proposed methods in [PV] can be utilized to determine the best explaining variables of the clustering structure without manual or visual interpretation, thus directing the clustering procedure and the KDD towards more automated and efficient data processing. Moreover, based on the articles [PIII], [PIV] and [PV], the parallelization of K-spatialmedians, the WG and PBM indices, and the Kruskal-Wallis-based feature ranking approaches seem an appealing, robust clustering combination for large-scale knowledge discovery.

Four clustering-based reference points selection methods proposed in the article [PVI] improve the MLM model building, especially when the aim is to build the regression model with a small number of reference points. This can be the case for large datasets where the size of the model may be limited based on computing resources. Based on the article [PVI], the recommended method for the selection of reference points in the regression tasks is the RS-maximin method. The RS-maximin is an efficient method for reference points selection, it has linear time complexity with respect to N, and it finds a set of points from data that are well separated from each other. The MLM has similar characteristics to prototype-based clustering due to distance-based learning; therefore emerging ideas from prototype-based clustering, e.g., for missing data handling and utilization of different distance measures, could further improve the MLM.

Quality, effectiveness, and scalability of the KDD process chain are based on multiple formulations and selections of the KDD steps, all the way starting from target data selection and preprocessing. In general, the contributions of this thesis scale the KDD towards big data and make the whole chain more reliable and efficient. In future work, the author will continue research in a project involving structure prediction of hybrid nanoparticles via artificial intelligence (HNP-AI).

This project aims to develop AI-based methods and software for predicting structures of metal nanoparticles based on known structures in the literature. The possibility to adapt the MLM to this problem will be evaluated. Moreover, during the course of this work, the author did some preliminary work related to unsupervised ensemble-based anomaly detection. An application of prototype-based clustering for unsupervised ensemble-based anomaly detection and a parallelization of the MLM will be studied later.

YHTEENVETO (FINNISH SUMMARY)

Prototyyppipohjaisen klusteroinnin elementtien parannukset ja sovellukset

Tämän kuuden artikkelin väitöskirjan pääasiallisena tavoitteena on kehittää tietämyksen muodostamisprosessia tehokkaammaksi, skaalautuvammaksi ja luotettavammaksi parantamalla prototyyppipohjaisen klusteroinnin eri osa-alueita. Klusterointi on yksi keskeisimmistä osa-alueista tiedonlouhinnassa, koneoppimisessa ja hahmontunnistuksessa. Datan jäsentäminen ryhmiin on luonnollinen lähestymistapa tietämyksen hankintaan tai tiivistetyn rakenteen muodostamiseen datasta. Prototyyppipohjaiseen klusterointiin perustuvat menetelmät, kuten K-means, ovat käytetyimpiä ja sovelletuimpia klusterointimenetelmiä, koska ne ovat yleensä helposti toteutettavissa ja vaativat usein vähemmän laskentaresursseja verrattuna muun tyyppisiin klusterointimenetelmiin. Suurten datamassojen käsittely lisää tarvetta tehokkaammille klusterointimenetelmille, jotka perustuvat useita laskentaytimiä hyödyntävään rinnakkaislaskentaan ja hajautetun muistin käyttöön. Tutkimuksen keskeisiä teemoja ovat klusteroinnin alustus ja skaalautuvuus, klustereiden validointi sekä klusteroinnin soveltaminen ohjatussa oppimisessa.

Alustuksella voi olla merkittäviä vaikutuksia prototyyppipohjaisen klusteroinnin tulosten laatuun ja konvergoimiseen. Ensimmäiset kaksi artikkelia esittelevät skaalautuvia alustusmenetelmiä K-means-menetelmälle. Esitetyt menetelmät pystytään rinnakkaistamaan helposti, ja ne skaalautuvat suurille datamassoille. Toisessa artikkelissa esitetty satunnaisprojektio-menetelmään perustuva rinnakkaistuva alustusmenetelmä on tehokas ja skaalautuva menetelmä korkeaulotteiselle big datalle, ja sitä voidaan myös käyttää yksinään klusterointirakenteen muodostamiseen. K-spatialmedians soveltuu hyvin harvan ja häiriöisen datan käsittelyyn. Kolmannessa artikkelissa esitetty K-spatialmedians-rinnakkaistus K-means-tyyppisellä alustuksella on laajennus K-spatialmedians-menetelmästä suurten datamassojen käsittelyyn.

Klusteroinnin validointi tehdään usein hyödyntäen validointi-indeksejä, jotka mittaavat kuinka kompakteja klusterit ovat (ideaalit klusterit ovat tiiviitä) ja kuinka hyvin klusterit ovat erillään toisistaan (ideaalit klusterit ovat isoloituneet). Neljännessä artikkelissa tutkittiin sisäisiä validointi-indeksien laajennuksia eri etäisyysmetriikoille prototyyppipohjaisen klusteroinnin kontekstissa. Tämän artikkelin perusteella osa sisäisistä validointi-indekseistä ehdottaa sopivia klusterointirakenteita paremmin eri etäisyysmetriikoilla. Viidennessä artikkelissa kehitettiin automaattisia klusterointituloksen tulkintamenetelmiä, jotka pystyvät löytämään klusterointituloksen selittävimmät piirteet automaattisesti. Kuudennessa artikkelissa sovelletut klusterointimenetelmät parantavat referenssipisteiden valintaa minimaaliselle oppimiskoneelle (minimal learning machine) regressiotehtävissä erityisesti silloin, kun malli rakennetaan mahdollisimman pienellä referenssipisteiden määrällä.

REFERENCES

[1] D. Achlioptas, "Database-friendly random projections," in *Proceedings of the Twentieth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*. ACM, 2001, pp. 274–281.

[2] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *Proceedings of the 8th International Conference on Database Theory*, 2001, pp. 420–434.

[3] S. Agrawal and J. Agrawal, "Survey on anomaly detection using data mining techniques," *Procedia Computer Science*, vol. 60, pp. 708–713, 2015.

[4] C. Alzate and J. A. Suykens, "Multiway spectral clustering with out-of-sample extensions through weighted kernel pca," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 2, pp. 335–347, 2010.

[5] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[6] S. Äyrämö, *Knowledge Mining Using Robust Clustering*. University of Jyväskylä, 2006, vol. 63 of Jyväskylä Studies in Computing.

[7] S. Äyrämö and T. Kärkkäinen, "Introduction to partitioning-based clustering methods with a robust example," *Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence 1/2006*, 2006.

[8] O. Bachem, M. Lucic, and A. Krause, "Distributed and provably good seedings for k-means in constant rounds," in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 292–300.

[9] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proc. VLDB Endow.*, vol. 5, no. 7, pp. 622–633, 2012.

[10] P. Berkhin, "A survey of clustering data mining techniques," in *Grouping Multidimensional Data*. Springer, 2006, pp. 25–71.

[11] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, "When is 'nearest neighbor' meaningful?" in *Proceedings of the 7th International Conference on Database Theory*, 1999, pp. 217–235.

[12] J. C. Bezdek, R. Ehrlich, and W. Full, "FCM: The fuzzy c-means clustering algorithm," *Computers & Geosciences*, vol. 10, no. 2–3, pp. 191–203, 1984.

[13] E. Bingham and H. Mannila, "Random projection in dimensionality reduction: applications to image and text data," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2001, pp. 245–250.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] L. Bottou and Y. Bengio, "Convergence properties of the k-means algorithms," in *Advances in Neural Information Processing Systems*, 1995, pp. 585–592.

[16] C. Boutsidis, A. Zouzias, M. W. Mahoney, and P. Drineas, "Randomized dimensionality reduction for k-means clustering," *IEEE Transactions on Information Theory*, vol. 61, no. 2, pp. 1045–1062, 2015.

[17] C. Boutsidis, A. Zouzias, and P. Drineas, "Random projections for k-means clustering," in *Advances in Neural Information Processing Systems*, 2010, pp. 298–306.

[18] C. Bouveyron and C. Brunet-Saumard, "Model-based clustering of high-dimensional data: A review," *Computational Statistics & Data Analysis*, vol. 71, pp. 52–78, 2014.

[19] P. S. Bradley and U. M. Fayyad, "Refining initial points for k-means clustering." in *ICML*, vol. 98, 1998, pp. 91–99.

[20] A. Broder, L. Garcia-Pueyo, V. Josifovski, S. Vassilvitskii, and S. Venkatesan, "Scalable k-means by ranked retrieval," in *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. ACM, 2014, pp. 233–242.

[21] T. Caliński and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics-Theory and Methods*, vol. 3, no. 1, pp. 1–27, 1974.

[22] Â. Cardoso and A. Wichert, "Iterative random projections for high-dimensional data clustering," *Pattern Recognition Letters*, vol. 33, no. 13, pp. 1749–1755, 2012.

[23] J. Y. Chan and A. P. Leung, "Efficient k-means++ with random projection," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 94–100.

[24] M. Chen, S. Mao, and Y. Liu, "Big data: A survey," *Mobile Networks and Applications*, vol. 19, no. 2, pp. 171–209, 2014.

[25] R. Chitta, *Kernel-Based Clustering of Big Data*. Michigan State University, 2015.

[26] M. Cochez and H. Mou, “Twister tries: Approximate hierarchical agglomerative clustering for average distance in linear time,” in *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*. ACM, 2015, pp. 505–517.

[27] A. Crotty, A. Galakatos, K. Dursun, T. Kraska, U. Çetintemel, and S. B. Zdonik, “Tupleware:’big’ data, big analytics, small clusters.” in *CIDR*, 2015.

[28] A. Crotty, A. Galakatos, and T. Kraska, “Tupleware: Distributed machine learning on small clusters.” *IEEE Data Engineering Bulletin*, vol. 37, no. 3, pp. 63–76, 2014.

[29] D. L. Davies and D. W. Bouldin, “A cluster separation measure,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-1, no. 2, pp. 224–227, 1979.

[30] A. De Mauro, M. Greco, and M. Grimaldi, “A formal definition of big data based on its essential features,” *Library Review*, vol. 65, no. 3, pp. 122–135, 2016.

[31] A. H. de Souza Junior, F. Corona, G. A. Barreto, Y. Miche, and A. Lendasse, “Minimal learning machine: A novel supervised distance-based approach for regression and classification,” *Neurocomputing*, vol. 164, pp. 34–44, 2015.

[32] A. H. de Souza Junior, F. Corona, Y. Miche, A. Lendasse, G. A. Barreto, and O. Simula, “Minimal learning machine: A new distance-based method for supervised learning,” in *International Work-Conference on Artificial Neural Networks*. Springer, 2013, pp. 408–416.

[33] Y. Demchenko, P. Grosso, C. de Laat, and P. Membrey, “Addressing big data issues in scientific data infrastructure,” in *Proceedings of the 2013 International Conference on Collaboration Technologies and Systems*, 2013, pp. 48–55.

[34] I. S. Dhillon and D. S. Modha, “A data-clustering algorithm on distributed memory multiprocessors,” in *Large-Scale Parallel Data Mining, Lecture Notes in Artificial Intelligence*, vol. 1759. Springer-Verlag, 2000, pp. 245–260.

[35] M. L. D. Dias, L. S. de Souza, A. R. da Rocha Neto, and A. H. de Souza Junior, “Opposite neighborhood: a new method to select reference points of minimal learning machines,” in *Proceedings of the 26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2018*, 2018, pp. 201–206.

[36] J. Drake and G. Hamerly, “Accelerated k-means with adaptive distance bounds,” in *5th NIPS Workshop on Optimization for Machine Learning*, 2012, pp. 42–53.

[37] D. R. Edla and P. K. Jana, “A prototype-based modified DBSCAN for gene clustering,” *Procedia Technology*, vol. 6, pp. 485–492, 2012.

[38] C. Elkan, “Using the triangle inequality to accelerate k-means,” in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 147–153.

[39] M. Emre Celebi, H. A. Kingravi, and P. A. Vela, “A comparative study of efficient initialization methods for the k-means clustering algorithm,” *Expert Systems with Applications*, 2012.

[40] M. Emre Celebi and H. A. Kingravi, “Deterministic initialization of the k-means algorithm using hierarchical clustering,” *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 26, no. 07, 2012.

[41] A. A. Esmin, R. A. Coelho, and S. Matwin, “A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data,” *Artificial Intelligence Review*, vol. 44, no. 1, pp. 23–45, 2015.

[42] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *KDD’96 Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

[43] A. Fahad, N. Alshatri, Z. Tari, A. Alamri, I. Khalil, A. Y. Zomaya, S. Foufou, and A. Bouras, “A survey of clustering algorithms for big data: Taxonomy and empirical analysis,” *IEEE Transactions on Emerging Topics in Computing*, vol. 2, no. 3, pp. 267–279, 2014.

[44] R. Farivar, D. Rebolledo, E. Chan, and R. H. Campbell, “A parallel implementation of k-means clustering on GPUs.” in *PDPTA*, 2008, pp. 340–345.

[45] T. Fawcett, “An introduction to ROC analysis,” *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[46] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI Magazine*, vol. 17, no. 3, p. 37, 1996.

[47] X. Z. Fern and C. E. Brodley, “Random projection for high dimensional data clustering: A cluster ensemble approach,” in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 2003, pp. 186–193.

[48] J. A. Florêncio, M. L. Dias, A. R. da Rocha Neto, and A. H. de Souza Júnior, “A fuzzy c-means-based approach for selecting reference points in minimal learning machines,” in *North American Fuzzy Information Processing Society Annual Conference*. Springer, 2018, pp. 398–407.

[49] E. W. Forgy, “Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications,” *Biometrics*, vol. 21, pp. 768–769, 1965.

[50] P. Fränti and S. Sieranoja, “K-means properties on six clustering benchmark datasets,” *Applied Intelligence*, pp. 1–17, 2018.

[51] M. Goldstein and S. Uchida, “A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data,” *Plos One*, vol. 11, no. 4, p. e0152173, 2016.

[52] Y. Gong, M. Pawlowski, F. Yang, L. Brandy, L. Bourdev, and R. Fergus, “Web scale photo hash clustering on a single machine,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 19–27.

[53] T. F. Gonzalez, “Clustering to minimize the maximum intercluster distance,” *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[54] G. Hamerly, “Making k-means even faster,” in *Proceedings of the 2010 SIAM International Conference on Data Mining*. SIAM, 2010, pp. 130–140.

[55] G. Hamerly and J. Drake, “Accelerating Lloyd’s algorithm for k-means clustering,” in *Partitional Clustering Algorithms*. Springer, 2015, pp. 41–78.

[56] J. Han, M. Kamber, and A. K. H. Tung, “Spatial clustering methods in data mining: A survey,” in *Geographic Data Mining and Knowledge Discovery, Research Monographs in GIS*, H. J. Miller and J. Han, Eds. Taylor and Francis, 2001.

[57] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*. The MIT Press, 2001.

[58] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

[59] J. M. Haut, M. Paoletti, J. Plaza, and A. Plaza, “Cloud implementation of the k-means algorithm for hyperspectral image analysis,” *The Journal of Supercomputing*, vol. 73, no. 1, pp. 514–529, 2017.

[60] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, “Extreme learning machine: A new learning scheme of feedforward neural networks,” in *2004 International Joint Conference on Neural Networks (IJCNN)*, vol. 2. IEEE, 2004, pp. 985–990.

[61] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[62] A. K. Jain and R. C. Dubes, “Algorithms for clustering data,” 1988.

[63] S. Jauhiainen and T. Kärkkäinen, “A simple cluster validation index with maximal coverage,” in *Proceedings of the 25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2017*, 2017, pp. 293–298.

[64] W. B. Johnson and J. Lindenstrauss, “Extensions of lipschitz mappings into a hilbert space,” *Contemporary Mathematics*, vol. 26, no. 189–206, p. 1, 1984.

[65] I. T. Jolliffe, *Principal Component Analysis*, 2nd ed. Springer, 2002.

[66] S. Kantabutra and A. L. Couch, "Parallel k-means clustering algorithm on NOWs," *NECTEC Technical Journal*, vol. 1, no. 6, pp. 243–247, 2000.

[67] T. Kärkkäinen, "On cross-validation for MLP model evaluation," in *Structural, Syntactic, and Statistical Pattern Recognition*, ser. Lecture Notes in Computer Science (8621). Springer-Verlag, 2014, pp. 291–300.

[68] T. Kärkkäinen and S. Äyrämö, "On computation of spatial median for robust data mining," *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN, Munich*, 2005.

[69] T. Kärkkäinen, "Extreme minimal learning machine – ridge regression with distance-based basis," *Neurocomputing*, 2018, to appear.

[70] T. Kärkkäinen and S. Äyrämö, "Robust clustering methods for incomplete and erroneous data," *WIT Transactions on Information and Communication Technologies*, vol. 33, 2004.

[71] T. Kärkkäinen and E. Heikkola, "Robust formulations for training multilayer perceptrons," *Neural Computation*, vol. 16, no. 4, pp. 837–862, 2004.

[72] I. Katsavounidis, C.-C. J. Kuo, and Z. Zhang, "A new initialization technique for generalized Lloyd iteration," *IEEE Signal Processing Letters*, vol. 1, no. 10, pp. 144–146, 1994.

[73] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *Proceedings of the 14th international joint conference on Artificial intelligence*, 1995, pp. 1137–1145.

[74] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[75] T. Korenius, J. Laurikkala, and M. Juhola, "On principal component analysis, cosine and euclidean measures in information retrieval," *Information Sciences*, vol. 177, no. 22, pp. 4893–4905, 2007.

[76] H.-P. Kriegel, E. Schubert, and A. Zimek, "The (black) art of runtime evaluation: Are we comparing algorithms or implementations?" *Knowledge and Information Systems*, vol. 52, no. 2, pp. 341–378, 2017.

[77] D. Laney, "3D data management: Controlling data volume, velocity, and variety," Tech. Rep., 2001.

[78] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[79] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006, pp. 287–296.

[80] T. Liu, C. Rosenberg, and H. A. Rowley, "Clustering billions of images with large scale nearest neighbor search," in *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*. IEEE, 2007, p. 28.

[81] S. Lloyd, "Least squares quantization in PCM," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[82] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14. Oakland, CA, USA., 1967, pp. 281–297.

[83] L. B. Marinho, J. S. Almeida, J. W. M. Souza, V. H. C. Albuquerque, and P. P. Rebouças Filho, "A novel mobile robot localization approach based on topological maps using classification with reject option in omnidirectional images," *Expert Systems with Applications*, vol. 72, pp. 1–17, 2017.

[84] L. B. Marinho, A. H. de Souza Junior, and P. P. Rebouças Filho, "A new approach to human activity recognition using machine learning techniques," in *International Conference on Intelligent Systems Design and Applications*. Springer, 2016, pp. 529–538.

[85] D. P. P. Mesquita, J. P. P. Gomes, and A. H. Souza Junior, "Ensemble of efficient minimal learning machines for classification and regression," *Neural Processing Letters*, pp. 1–16, 2017.

[86] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera, "Study on the impact of partition-induced dataset shift on k-fold cross-validation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1304–1312, 2012.

[87] F. Murtagh and P. Contreras, "Random projection towards the baire metric for high dimensional clustering," in *International Symposium on Statistical Learning and Data Sciences*. Springer, 2015, pp. 424–431.

[88] J. Nayak, B. Naik, and H. Behera, "Fuzzy c-means (FCM) clustering algorithm: a decade review from 2000 to 2014," in *Computational Intelligence in Data Mining – Volume 2*. Springer, 2015, pp. 133–149.

[89] A. Y. Ng, M. I. Jordan, and Y. Weiss, "On spectral clustering: Analysis and an algorithm," in *Advances in Neural Information Processing Systems*, 2002, pp. 849–856.

[90] C. Otto, D. Wang, and A. K. Jain, "Clustering millions of faces by identity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 2, pp. 289–303, 2018.

[91] S. H. Park, I. D. Yun, and S. U. Lee, "Color image segmentation based on 3-D clustering: morphological approach fn1," *Pattern Recognition*, vol. 31, no. 8, pp. 1061–1076, 1998.

[92] J. M. Pena, J. A. Lozano, and P. Larranaga, "An empirical comparison of four initialization methods for the k-means algorithm," *Pattern Recognition Letters*, vol. 20, no. 10, pp. 1027–1040, 1999.

[93] J. Qin, W. Fu, H. Gao, and W. X. Zheng, "Distributed k-means algorithm and fuzzy c-means algorithm for sensor networks based on multiagent consensus theory," *IEEE Transactions on Cybernetics*, vol. 47, no. 3, pp. 772–783, 2017.

[94] S. Rana, S. Jasola, and R. Kumar, "A review on particle swarm optimization algorithms and their applications to data clustering," *Artificial Intelligence Review*, vol. 35, no. 3, pp. 211–222, 2011.

[95] S. Ray and R. H. Turi, "Determination of number of clusters in k-means clustering and application in colour image segmentation," in *Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques*. Calcutta, India, 1999, pp. 137–143.

[96] P. P. Rebouças Filho, S. A. Peixoto, R. V. M. da Nóbrega, D. J. Hemanth, A. G. Medeiros, A. K. Sangaiah, and V. H. C. de Albuquerque, "Automatic histologically-closer classification of skin lesions," *Computerized Medical Imaging and Graphics*, 2018.

[97] C. K. Reddy and B. Vinzamuri, "A survey of partitional and hierarchical clustering algorithms," in *Data Clustering Algorithms and Applications*. Chapman and Hall/CRC, 2013, pp. 87–110.

[98] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz, "Internal versus external cluster validation indexes," *International Journal of Computers and Communications*, vol. 5, no. 1, pp. 27–34, 2011.

[99] M. Saarela, *Automatic Knowledge Discovery from Sparse and Large-Scale Educational Data Case Finland*. University of Jyväskylä, 2017, vol. 262 of Jyväskylä Studies in Computing.

[100] M. Saarela and T. Kärkkäinen, "Do country stereotypes exist in educational data? A clustering approach for large, sparse, and weighted data." in *Proceedings of the 8th International Conference on Educational Data Mining (EDM 2015)*, 2015, pp. 156–163.

[101] S. S. Sandhu, B. Tripathy, and S. Jagga, "Kmst+: A k-means++-based minimum spanning tree algorithm," in *Smart Innovations in Communication and Computational Sciences*. Springer, 2018, pp. 113–127.

[102] J. Schneider and M. Vlachos, "Fast parameterless density-based clustering via random projections," in *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. ACM, 2013, pp. 861–866.

[103] M. J. Schneider and S. Gupta, "Forecasting sales of new and existing products using consumer reviews: A random projections approach," *International Journal of Forecasting*, vol. 32, no. 2, pp. 243–256, 2016.

[104] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the 19th International Conference on World Wide Web*. ACM, 2010, pp. 1177–1178.

[105] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[106] G. Sharma and J. Martin, "MATLAB: A language for parallel computing," *International Journal of Parallel Programming*, vol. 37, no. 1, pp. 3–36, 2009.

[107] A. S. Shirkhorshidi, S. Aghabozorgi, T. Y. Wah, and T. Herawan, "Big data clustering: a review," in *Proceedings of the International Conference on Computational Science and Its Applications*. Springer, 2014, pp. 707–720.

[108] M. Steinbach, G. Karypis, V. Kumar *et al.*, "A comparison of document clustering techniques," in *KDD Workshop on Text Mining*, vol. 400, no. 1. Boston, 2000, pp. 525–526.

[109] D. Steinley, "Local optima in k-means clustering: what you don't know may hurt you." *Psychological Methods*, vol. 8, no. 3, p. 294, 2003.

[110] A. Strehl and J. Ghosh, "Cluster ensembles – a knowledge reuse framework for combining multiple partitions," *Journal of Machine Learning Research*, vol. 3, pp. 583–617, 2002.

[111] S. Tasoulis, L. Cheng, N. Välimäki, N. J. Croucher, S. R. Harris, W. P. Hanage, T. Roos, and J. Corander, "Random projection based clustering for population genomics," in *Proceedings of the 2014 IEEE International Conference on Big Data (Big Data)*. IEEE, 2014, pp. 675–682.

[112] L. van der Maaten, E. Postma, and J. van den Herik, "Dimensionality reduction: a comparative review," *Journal of Machine Learning Research*, vol. 10, pp. 66–71, 2009.

[113] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Proceedings of the 8th International Conference on Artificial Neural Networks: Computational Intelligence and Bioinspired Systems*, ser. IWANN'05. Springer-Verlag, 2005, pp. 758–770.

[114] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data - a survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[115] P. Wartiainen and T. Kärkkäinen, "Hierarchical, prototype-based clustering of multiple time series with missing values," in *Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning – ESANN 2015*, 2015, pp. 95–100.

[116] D. Wettschereck, D. W. Aha, and T. Mohri, "A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms," *Artificial Intelligence Review*, vol. 11, no. 1-5, pp. 273–314, 1997.

[117] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip *et al.*, "Top 10 algorithms in data mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1–37, 2008.

[118] R. Xu and D. Wunsch, "Survey of clustering algorithms," *IEEE Transactions on Neural Networks*, vol. 16, no. 3, pp. 645–678, 2005.

[119] Y. Xu, W. Qu, Z. Li, G. Min, K. Li, and Z. Liu, "Efficient k -means++ approximation with mapreduce," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 12, pp. 3135–3144, 2014.

[120] X. Zeng and T. R. Martinez, "Distribution-balanced stratified cross-validation for accuracy estimation," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 12, no. 1, pp. 1–12, 2000.

[121] J. Zhang, G. Wu, X. Hu, S. Li, and S. Hao, "A parallel k-means clustering algorithm with MPI," in *2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*. IEEE, 2011, pp. 60–64.

[122] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *Proceedings of the ACM SIGMOD Conference on Management of Data*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[123] Q. Zhao and P. Fränti, "Wb-index: A sum-of-squares based index for cluster validity," *Data & Knowledge Engineering*, vol. 92, pp. 77–89, 2014.

[124] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Proceedings of the 1st International Conference on Cloud Computing*, ser. CloudCom '09, 2009, pp. 674–679.

[125] C. Zhong, M. Malinen, D. Miao, and P. Fränti, "A fast minimum spanning tree algorithm based on k-means," *Information Sciences*, vol. 295, pp. 1–17, 2015.

ORIGINAL PAPERS

PI

INITIALIZATION OF BIG DATA CLUSTERING USING DISTRIBUTIONALLY BALANCED FOLDING

by

Joonas Hämäläinen and Tommi Kärkkäinen 2016

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks,
Computational Intelligence and Machine Learning

Reproduced with kind permission of ESANN.

ESANN 2016 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 27-29 April 2016, i6doc.com publ., ISBN 978-287587027-8. Available from http://www.i6doc.com/en/.

Initialization of Big Data Clustering using Distributionally Balanced Folding

Joonas Hämäläinen and Tommi Kärkkäinen

Department of Mathematical Information Technology
P.O. Box 35, 40014 University of Jyväskylä - Finland

Abstract. Use of distributionally balanced folding to speed up the initialization phase of *K-means++* clustering method, targeting for big data applications, is proposed and tested. The approach is first described and then experimented, by focusing on the effects of the sampling method when the number of folds created is varied. In the tests, quality of the final clustering results were assessed and scalability of a distributed implementation was demonstrated. The experiments support the viability of the proposed approach.

1 Introduction

Iterative relocation clustering algorithms are known to be sensitive to the initial placement of the prototypes. Actually the twofold aim of clustering, to divide data into groups where observations within a group are more similar to each other than observations in other groups [1], is approached in the well-known algorithms, most prominently in the classical *K-means++* [2, 1], by using the two main steps: *i*) initial location of K separate prototypes, *ii*) local refinement (search) of the initial prototypes to get the final solution. Due to variations of step *i*), it is known that this kind of algorithms do not guarantee unique clustering result or convergence to the global minimum of the clustering error (e.g., [1, 3, 4]).

The final clustering result can be improved by using some other than the random strategy for the initialization [5]. Chen et al. [6] argue that in the high-dimensional space data are inherently sparse. This is due to the well-known *curse of dimensionality* [7]. For example, the random samples tend to concentrate on the corners of a hypercube and the distance between each pair of observations becomes almost the same for a wide variety of data distributions. Chen et al. [6] conclude that, for small data sets, the method by Bradley and Fayyad [8], where the original data set is first splitted into smaller subsets which themselves are clustered, yielded to the best clustering results. In general, one agrees that the initial prototypes should be as far from each other as possible (without being outliers) [9, 1]. Lately, the K-means++ algorithm [10], where the random initialization is based on a density function favoring distinct prototypes, has become the most popular variant to initialize the K-means-type of algorithms.

Sampling is the basic approach to reduce the number of observations in statistics. It has a natural role in big data applications to cope with data volume, although one of the characteristics of big data [11] is precisely the lack of stable distribution, especially with high veracity and velocity. But the way sampling is done matters, and this issue has mostly been considered in relation to the well-known cross-validation, CV (see [12, 13] and articles therein). More precisely, the *Distribution Optimally Balanced Stratified CV, DOB-SCV*, has been evaluated as an appropriate sampling method for large pool of datasets to ensure good approximation of data density in the distinct folds created.

An approach where folding is applied with the K-means++ address the special nature of a high-dimensional data by separating the observations more clearly. This is combined with the good initialization strategies as mentioned above: splitting of data and the distinct initial locations. According to our knowledge, such techniques and in particular the DOB-SCV algorithm, has not been previously suggested or tested in the big data clustering context. This is the main goal of the paper, whose structure is as follows: In Section 2, we describe the clustering approach and in Section 3 provide results from computational experiments. The paper is shortly concluded in Section 4.

2 The Method

Let $\mathbf{X}$ be the given set of observations. In the initialization of the K-means++ algorithm [10], one prototype is first selected at random from $\mathbf{X}$. Then the rest $K-1$ prototypes are selected from $\mathbf{X}$ with probability $d(\mathbf{x})^2 / \sum_{\mathbf{x}\in\mathbf{X}} d(\mathbf{x})^2$, where $d(\mathbf{x})$ is the smallest distance to a prototype that was already selected. Hence, with high probability, one obtains a set of clearly distinct initial prototypes.

The DOB-SCV, as proposed in [12] (see [13] for the actual algorithm), was targeted to create folds (disjoint subsets of data) for the cross-validation. Stratification means that the folds are created classwise, or approximating the whole data distribution if no labelling is available.

The basic strategy to create k folds is, interestingly, opposite to clustering: select a random observation from class j, add it to the first fold and then add its $k-1$ nearest class neighbors to different folds without replacement. This process is then repeated until all the observations within class j are assigned to folds. These steps are applied to all classes. As a result, in addition to the classical stratification of approximating the class sizes, also class densities are preserved in the folds. Our approach here is to use this approach to speed up the initial search of distant prototypes in the K-means++ algorithm. From preliminary tests that we have made with the available K-means implementations on the popular Hadoop platform, we have noticed that the nondistributed initialization, especially for the K-means++ implementations, takes most of the computing time.

The proposed method is formalized in Algorithm 1. We let DOB-SCV to form k distjoint folds $\{\mathbf{X}_i\}_{i=1}^{k}$ from the data $\mathbf{X}$. The initial selection of distinct prototypes in K-means++ is then done in the folds where the best initial solution by means of the overall clustering (least-squares) error is selected to initialize the actual search (cf. [8]). Steps 1-2 in Algorithm 1 can be easily parallelized by distributing folds to workers and using Single Program Multiple Data (SPMD) model so that each worker processes its local fold. Communication between workers can be done with Message Passing Interface (MPI). Same folds and workers can also be applied in the local refinement step of the K-means algorithm in parallel using SPMD model and MPI, see [14].

3 Experimental results

Our first goal of the experiments is to compare how the initialization method affects to the clustering results in comparison to the whole data K-means++ initialization. Secondly, we test the scalability of the proposed approach when the number of folds is varied, by using SPMD implementation of Algorithm 1. As reference, we used the implementation [15] for K-means++. Matlab Distributed Computing Toolbox SPMD and Message passing functions were used for the parallel implementation. All the tests were performed in Matlab 8.3.0 (R2014a) environment. For the scalability tests, we used the Taito supercluster at IT Center for Science in Finland. Cluster resources were utilized via Matlab Distributed Computing Server (MDCS). The tests were run in the parallel partition with Sandy Bridge nodes having two eight-core Intel E5-2670 2.6 GHz processors with 256 GB RAM (16 GB per core).

Quality experiments were run with 5 data sets: S-sets [16] and USPS database [17]. S-sets consist of four real-valued 2-dimensional synthetic data sets generated from 15 gaussian distributions with known centers. Each S-set consists of 5,000 vectors. The overlappings of clusters increase from S_1 to S_4. The USPS is a well-known benchmark of handwritten digit dataset with 9,298 16×16 images and labels provided. This set consists of 10 classes. The number of clusters K was fixed for S-sets to 15 and for USPS dataset to 10. For all datasets, the variables were min-max scaled into $[-1, 1]$.

We ran Algorithm 1 with varying the number of the folds k between $2 - 50$ for the even numbers. For each k, K-means++ run was repeated k times and the minimum *sum-of-squared-errors*, SSE, was used to select as the final, reference clustering result. Tests were repeated 10 times for the both methods with DOB-SCV refolding in Algorithm 1. Comparison of the clustering results for K-means++ and Algorithm 1 was performed by analysing SSE and the pairwise prototype distances. For the latter measure, we calculated the sum of the smallest pairwise Euclidean distances, in such a way that each prototype was linked to the corresponding prototype only once. These were then averaged

Algorithm 1 K-means++ with folding initialization

```
Input: Folds $\{\mathbf{X}_i\}_{i=1}^{k}$ from DOB-SCV, number of clusters $K$.
Output: Set of prototypes $\mathbf{C} = \{\mathbf{c}_j\}_{j=1}^{K}$.
1: For each fold $\mathbf{X}_i$ do K-means++ clustering initialization.
2: For each set of prototypes $\mathbf{C}_i$ from a fold, calculate clustering error for the
   whole data.
3: Select the set of prototypes $\mathbf{C}_m$ with the smallest clustering error.
4: Do the K-means search starting from $\mathbf{C}_m$.
```

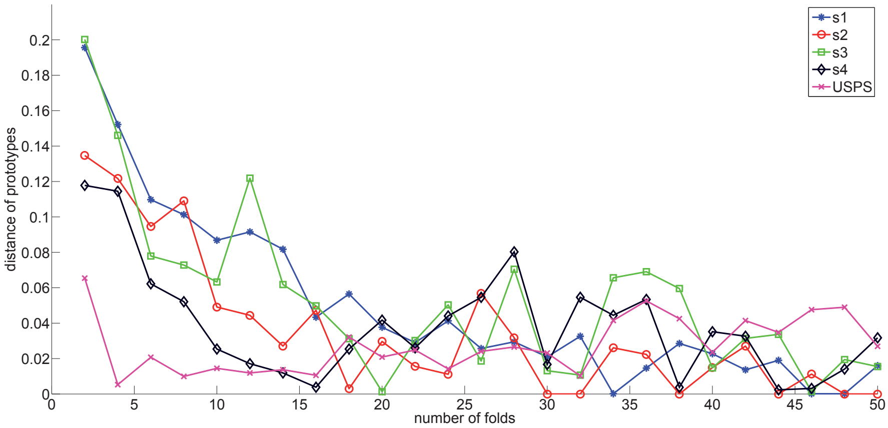


Fig. 1: Normalized prototype distances between Algorithm 1 and K-means++.

over 10 testruns for each k and normalized by dividing the result with the sum of the Euclidean norms of ground truth prototypes. For the USPS dataset, the ground truth prototypes were calculated as the class means.

In Fig. 1, the behavior of the normalized prototype distances are depicted. In general, all the results are very close to each other. We also observe that the prototypes become more similar between the methods when k increases. This might be due to the increase of the accuracy for both methods, more precisely, the number of i) the initial prototypes for Algorithm 1 and ii) the final prototypes for K-means++.

In Fig. 2, the SSE difference is the average SSE for the K-means++ which is subtracted from the average SSE for Algorithm 1. The SSE differences were again normalized, this time with the SSE for the ground truth prototypes. Al-

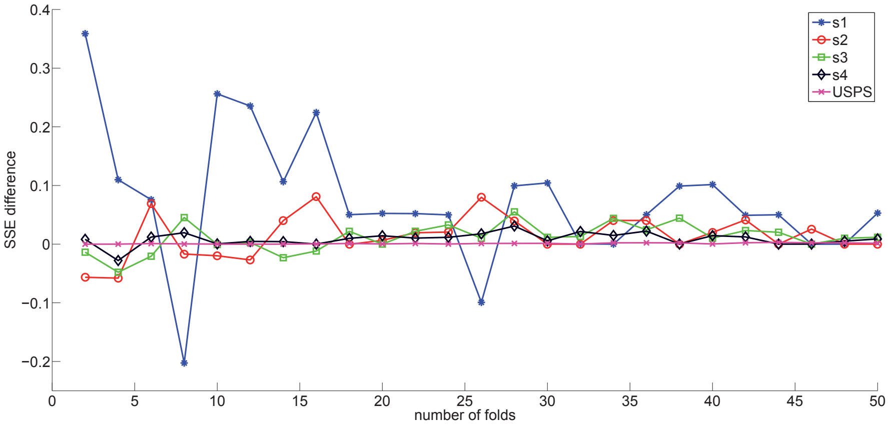
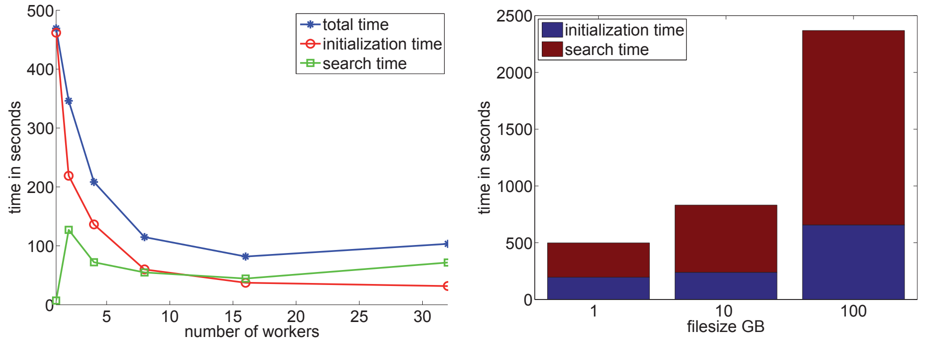


Fig. 2: Normalized SSE difference between Algorithm 1 and K-means++.



(a)

(b)

Fig. 3: Initialization and search phases wall times for parallellized Algorithm 1.

gorithm 1 occasionally gives smaller errors than the repeated, full K-means++, especially for the smaller values of k. A strong variation of the SSE difference for the dataset S_1 is most likely a consequence of higher probability to get stuck in a local minimum. For USPS, the SSE difference is close to zero for all the values of k, which indicates that the accuracy of Algorithm 1 is improved when the volume of the problem is increased. This is desirable in big data applications.

For the scalability tests with the implementation as described above, the MNIST database [18] was used. MNIST is another classical benchmark with handwritten digits that consists of 70,000 28×28 images with labels for 10 classes. In the first experiment, the number of workers were varied as 1, 2, 4, 8, 16 and 32. We also increased the data volume to demonstrate the behaviour: each data fold was copied 3 times for both observation and variable direction. Hence, the total data size was approximately 210,000×2,352. The wall clock time in Algorithm 1 for the initialization (Steps 1-3) and for the search phase (Step 4) was measured, again repeating the folding 10 times and averaging the wall clock times. Note that total time for the original (full data) K-means++ is given with one worker in Algorithm 1. As we can see from Fig. 3a, initialization time was reduced rapidly up to 8 workers until the time in communication starts to dominate, especially for the search phase.

In the second experiment, the number of workers was fixed to 100 and data size was varied approximately as, again using MNIST for copying, 210,000×2,352 (1 GB), 630,000×7,056 (10 GB) and 2,030,000×22,736 (100 GB). Algorithm 1 was ran once for each data and wall times were measured similarly as in the first experiment. Results are shown in Fig. 3b, which demonstrates that proposed method, indeed, scales for big data.

4 Conclusions

We proposed and tested distributionally optimal folding as an initialization method for the K-means++ clustering algorithm. The proposed initialization method can be easily parallellized to speedup the initialization phase. Based on the experiments, SSE can be occasionally even smaller for the proposed method compared to the k times repeated full K-means++. Overall, especially for larger values of k, the method provides very similar results compared to the full data approach. In our future work, targeting at big data applications, integration of dimension reduction to the initialization and the search phase is to be studied to reduce the data volume in iterative relocation algorithms even further.

References

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

[2] S. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[3] M. Emre Celebi, H. A. Kingravi, and P. A. Vela. A comparative study of efficient initialization methods for the k-means clustering algorithm. *Expert Systems with Applications*, 2012.

[4] M. Saarela and T. Kärkkäinen. Analyzing student performance using sparse data of core bachelor courses. *Journal of Educational Data Mining*, 7(1):3–32, 2015.

[5] L. Bai, J. Liang, and C. Dang. An initialization method to simultaneously find initial cluster centers and the number of clusters for clustering categorical data. *Knowledge-Based Systems*, 24(6):785–795, 2011.

[6] L. Chen, L. Chen, Q. Jiang, B. Wang, and L. Shi. An initialization method for clustering high-dimensional data. In *Database Technology and Applications, 2009 First International Workshop on*, pages 444–447. IEEE, 2009.

[7] M. Verleysen and D. François. The Curse of Dimensionality in Data Mining. *Analysis*, 3512:758 – 770, 2005.

[8] P. S. Bradley and U. M. Fayyad. Refining initial points for k-means clustering. In *ICML*, volume 98, pages 91–99, 1998.

[9] S. S. Khan and A. Ahmad. Cluster center initialization algorithm for k-modes clustering. *Expert Systems with Applications*, 2013.

[10] D. Arthur and S. Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[11] B. Hammer, H. He, and T. Martinetz. Learning and modeling big data. *22th European Symposium on Artificial Neural Networks (ESANN2014)*, (April):23–25, 2014.

[12] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera. Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, 23(8):1304–1312, 2012.

[13] T. Kärkkäinen. On cross-validation for MLP model evaluation. In *Structural, Syntactic, and Statistical Pattern Recognition*, Lecture Notes in Computer Science (8621), pages 291–300. Springer-Verlag, 2014.

[14] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. *LargeScale Parallel Data Mining*, 1759(802):245–260, 1999.

[15] L. Sorber. k-means++ - File Exchange - MATLAB Central, 2010.

[16] P. Fränti and O. Virmajoki. Iterative shrinking method for clustering problems. *Pattern Recognition*, 39(5):761–775, 2006.

[17] J. J. Hull. A database for handwritten text recognition research. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(5):550–554, May 1994.

[18] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2323, 1998.

PII

SCALABLE INITIALIZATION METHODS FOR CLUSTERING LARGE DATASETS

by

Joonas Hämäläinen, Tommi Kärkkäinen and Tuomo Rossi 2018

Pattern Recognition Letters (in revision)

Pattern Recognition Letters
journal homepage: www.elsevier.com

Scalable initialization methods for clustering large datasets

Joonas Hämäläinen[a,**], Tommi Kärkkäinen[a], Tuomo Rossi[a]

[a] *University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35, FI-40014 University of Jyvaskyla, Finland*

ABSTRACT

In this work, two novel initialization methods for K-means clustering are proposed. Both proposals are based on applying a divide-and-conquer approach for the K-means|| type of an initialization strategy. The second proposal also utilizes multiple lower-dimensional subspaces produced by the random projection method for the initialization. The proposed methods are scalable and can be run in parallel, which make them suitable for initializing large-scale problems. In the experiments, comparison of the proposed methods to the K-means++ and K-means|| methods is conducted using several large-scale datasets. The experiments show that the proposed methods compare favorably to the state-of-the-art.

1. Introduction

Clustering is one of the core techniques in pattern recognition. Its purpose is to form groups from data in a way that the observations within one group, the cluster, are similar to each other and dissimilar to observations in other groups.

Prototype-based clustering algorithms, such as the popular K-means (Lloyd, 1982), are known to be sensitive to initialization (Emre Celebi et al., 2012), i.e., the selection of initial prototypes. A proper set of initial prototypes can improve the clustering result and decrease the number of iterations needed for the convergence of an algorithm (Hämäläinen et al., 2017). The initialization of K-means was remarkably improved by the work of Arthur and Vassilvitskii (2007), where they proposed the K-means++ method. There, the initial prototypes are determined by favoring distinct prototypes, which in high probability are not similar to the already selected ones.

A drawback of K-means++ is that the initialization phase requires K inherently sequential passes over the data, since the selection of a new initial prototype depends on the previously selected prototypes. Bahmani et al. (2012) proposed a parallel initialization method called K-means||. The K-means|| speeds up initialization by sampling each point independently and by updating sampling probabilities less frequently. Independent sampling of the points enables parallelization of the initialization, thus providing a speedup over K-means++. However, for example MapReduce based implementation of K-means|| needs multiple MapReduce jobs for the initialization. The MapReduce K-means++ method by Xu et al. (2014) tries to address this issue, as it uses one MapReduce job to select K initial prototypes, which speeds up the initialization compared to K-means||. Suggestions of parallelizing the search phase of K-means have been given in several papers (see, e.g., (Dhillon and Modha, 1999; Zhao et al., 2009)). On a single machine, distance pruning approaches can be utilized to speed up K-means without affecting the clustering results (Elkan, 2003; Hamerly, 2010).

Dimension reduction has had an important role in making clustering algorithms more efficient. Over the years, various dimension reduction methods have been applied to decrease the dimension of data in order to speed up clustering algorithms (Boutsidis et al., 2010; Fern and Brodley, 2003; Alzate and Suykens, 2010; Napoleon and Pavalakodi, 2011). The key idea for improved efficiency is to solve an approximate solution to the clustering problem in a lower dimensional space. Dimension reduction methods are usually divided into two categories: feature selection methods and feature extraction methods (Liu and Motoda, 1998). Feature selection methods aim to select a subset of the most relevant variables from the original variables. Correspondingly, feature extraction methods aim to transform the original dataset into a lower dimensional space while trying to preserve the characteristics of the original data.

A particular dimensional reduction approach for processing large datasets is the random projection (RP) method (Achlioptas, 2003). Projecting data from the original space to a lower

**Corresponding author
e-mail: `joonas.k.hamalainen@jyu.fi` (Joonas Hämäläinen)

dimensional space while preserving the distances is the main characteristic of the RP method. This makes RP very appealing in clustering, whose core concept is similarity. Moreover, classical dimension reduction methods such as the principal component analysis (PCA) (Jolliffe, 2002) become expensive to compute for high-dimensional spaces whereas RP remains computationally efficient (Bingham and Mannila, 2001).

Fern and Brodley (2003) proposed an ensemble clustering method based on RP. They showed empirically that aggregation of clustering results from multiple lower dimensional spaces produced by RP leads to better clustering results compared to a single clustering in lower dimensional space produced by PCA or RP. Other combinations of K-means and RP have been studied in several papers (Cohen et al., 2014; Boutsidis et al., 2010, 2015; Cardoso and Wichert, 2012). RP for K-means++ was analyzed in Chan and Leung (2017).

In general, the typical K-means clustering procedure is to use non-deterministic initialization, such as K-means++, followed by the Lloyd's iterations with multiple restarts. Prototypes corresponding to the smallest sum-of-squared clustering error are selected as the final clustering result. In Hämäläinen and Kärkkäinen (2016), such a multistart strategy was carried out during the initialization phase, thus reducing the need to repeat the whole clustering algorithm. More precisely, they proposed and tested a parallel method based on K-means++ clustering of subsets produced by the distribution optimally balanced stratified cross-validation (DOB-SCV) algorithm (Moreno-Torres et al., 2012). Here, such an approach is developed further with the help of K-means|| and RP. More precisely, we run K-means|| method in the low-dimensional subsets, which are created by RP. In contrast to the previous work (Hämäläinen and Kärkkäinen, 2016), the new methods also restrict the number of Lloyd's iterations in the subsets. Hence, the proposed initialization method reduces the volume of data with sampling, subsampling, and dimensional reduction, and then solves the K-means clustering problem in a coarse fashion. Moreover, from the perspective of parallel computing, using a parallelizable clustering method in the subset clustering allows fixing the number of subsets and treating each subset locally in parallel, hence improving the scalability. To this end, the purpose of this article is to propose two new algorithms for clustering initialization and compare them experimentally with K-means++ and K-means|| using several large-scale datasets.

2. Background

In this section, we introduce the basic composition of the existing algorithms.

2.1. K-means clustering problem and the basic algorithms

Let $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$ be a dataset such that $\mathbf{x}_i \in \mathbb{R}^M \; \forall 1 \leq i \leq N$, and let $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$ be a set of prototypes, where each prototype also belongs to $\mathbb{R}^M$. The goal of the K-means clustering algorithm is to find a partition of $\mathbf{X}$ into K disjoint subsets, by minimizing the sum-of-squared error (SSE) defined as

$$\text{SSE}(\mathbf{C}) = \sum_{\mathbf{x}\in\mathbf{X}} \min_{\mathbf{c}\in\mathbf{C}} \|\mathbf{c} - \mathbf{x}\|^2. \tag{1}$$

Algorithm 1: K-means||

```
Input: Dataset $\mathbf{X}$, #clusters $K$, and over-sampling factor $l$.
Output: Set of prototypes $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$.
$\mathbf{C} \leftarrow$ select point $\mathbf{c}_1$ uniformly random from $\mathbf{X}$.
$\psi \leftarrow$ compute $SSE(\mathbf{C})$.
for $O(\log(\psi))$ times do
    $\mathbf{C}' \leftarrow$ sample each point $\mathbf{x} \in \mathbf{X}$ independently with
        probability $l \cdot d(\mathbf{x})^2/SSE(\mathbf{C})$.
    $\mathbf{C} \leftarrow \mathbf{C} \cup \mathbf{C}'$
For each $\mathbf{x}$ in $\mathbf{C}$ attach a weight defined as the number of
    points in $\mathbf{X}$ closer to $\mathbf{x}$ than any other point in $\mathbf{C}$.
Do a weighted clustering of $\mathbf{C}$ into $K$ clusters.
```

An approximate solution to the minimization problem with (1) is typically computed by utilizing the Lloyd's K-means algorithm (Lloyd, 1982). Its popularity is based on simplicity and scalability. Notice that because of the min-operator, the cost function in (1) is mathematically nonsmooth, i.e., nondifferentiable (Kärkkäinen and Heikkola, 2004). However, it is easy to show that after the initialization, the K-means type of iterative relocation algorithm converges in finite many steps, because the value of SSE is decreased in each iteration (Hämäläinen et al., 2017).

Prototype-based clustering algorithms, like K-means, are initialized before the prototype relocation (search) phase. The classical initialization algorithm, readily proposed in MacQueen (1967), is to randomly generate the initial set of prototypes. A slight refinement of this strategy is to select, instead of random points (from appropriate value ranges), random indices and use the corresponding observations in data as initialization (Forgy, 1965). Because of this choice, there cannot be empty clusters in the first iteration. Bradley and Fayyad (1998) proposed an initialization method where J randomly selected subsets of the data are first clustered with K-means. Next, it forms a superset of the $J \times K$ prototypes obtained from the subset clustering. Finally, the initial prototypes are achieved as the result of K-means clustering of the superset.

Arthur and Vassilvitskii (2007) introduced the K-means++ algorithm, which improves the initialization of K-means clustering. The algorithm selects first prototype at random, and then the remaining $K - 1$ prototypes are sampled using probabilities based on the squared distances to the already selected set, thus favoring distant prototypes. The generalized form of such an algorithm with different l_p-distance functions and the corresponding cluster location estimates was depicted in Hämäläinen et al. (2017).

The parallelized K-means++ method, called K-means||, was proposed by Bahmani et al. (2012) (see Algorithm 1). In the algorithm, one samples points from $\mathbf{X}$ in a slightly different fashion compared to K-means++. More precisely, the sampling probabilities are multiplied with the over-sampling factor l and the sampling is done independently for each data point. The initial SSE for the first sampled point ψ determines the number of sampling iterations. K-means|| runs $O(log(\psi))$ sampling iterations. For each iteration, the expected number of points is l. Hence, after $O(log(\psi))$ iterations, the expected number of points

```
Algorithm 2: SK-means||
Input: Subsets $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_S\}$ , #clusters $K$, and #Lloyd's
    iterations $T_{init}$.
Output: Set of prototypes $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$.
1: $\mathbf{C}_i \leftarrow$ for each subset $\mathbf{X}_i$ run K-means||.
2: $\mathbf{C}_i \leftarrow$ for each subset $\mathbf{X}_i$ run $T_{init}$ Lloyd's iterations
    initialized with $\mathbf{C}_i$.
3: Compute local SSE for each $\mathbf{C}_i$ in $\mathbf{X}_i$.
4: $\mathbf{C} \leftarrow$ select prototypes corresponding to smallest local
    SSE.
```

added to $\mathbf{C}$ is $O(l\,log(\psi))$. Finally, weights representing the accumulation of data around the sampled points are set and the result of the weighted clustering then provides the K initial prototypes. K-means++ can be used to cluster the weighted data (see Algorithm 1 in Bachem et al. (2017)). Selecting $r = 5$ instead of $O(log(\psi))$ rounds and setting the over-sampling factor to $2K$ were demonstrated to be sufficient in Bahmani et al. (2012). Recently, Bachem et al. (2017) proved theoretically that small r instead of $O(log(\psi))$ iterations is sufficient in K-means||. A modifcation of K-means|| for initializing robust clustering was described and tested in (Hämäläinen et al., 2018).

2.2. *Random projection*

The background for RP (Achlioptas, 2003) comes from the Johnson-Lindenstrauss lemma (Johnson and Lindenstrauss, 1984). The lemma states that points in a high dimensional space can be projected to a lower dimension space while approximately preserving the distances of the points, when the projection is done with a matrix whose elements are randomly generated. Hence, for an $N \times M$ dataset $\mathbf{X}$, let $\mathbf{R} \in M \times P$ be a random matrix. Then, the random projected data matrix $\widetilde{\mathbf{X}}$ is given by $\widetilde{\mathbf{X}} = \frac{1}{\sqrt{P}}\mathbf{X}\mathbf{R}$. The random matrix $\mathbf{R}$ consists of independent random elements (r_{ij}) which can be drawn from one of the following probability distributions (Achlioptas, 2003): $r_{ij} = +1$ with probability $1/2$, or -1 with probability $1/2$; or $r_{ij} = +1$ with probability $1/6$, 0 with probability $2/3$, or -1 with probability $1/6$.

3. Reduced K-means|| type initialization

Next we introduce the novel initialization algorithms for K-means.

3.1. *SK-means||*

The first new initialization method for K-means clustering, Subset K-means|| (SK-means||), is described in Algorithm 2. The method is based on S randomly sampled non-disjoint subsets $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_S\}$ from $\mathbf{X}$ of approximately equal size, such as $\mathbf{X} = \cup_{i=1}^{S}\mathbf{X}_i$. First, K-means|| is applied in each subset, which gives the corresponding set of initial prototypes $\mathbf{C}_i$. Next, each initial prototype set $\mathbf{C}_i$ in $\mathbf{X}_i$ is refined with T_{init} Lloyd's iterations. T_{init} is assumed to be significantly smaller than the number of Lloyd's iterations needed for convergence. Then, SSE is computed locally for each $\mathbf{C}_i$ in $\mathbf{X}_i$. Differently from the earlier work (Hämäläinen and Kärkkäinen, 2016), this locally computed SSE is now used as the selection criteria for the initial prototypes instead of the global SSE. Computation of SSE for $\mathbf{X}_i$ in Step 3 is obviously much faster than to compute it for the whole $\mathbf{X}$. However, a drawback is that if the subsets are too small to characterize the whole data, the selection of the initial prototypes might fail. Therefore, S should be selected such that the subsets are sufficiently large.

```
Algorithm 3: SRPK-means||
Input: Subsets $\{\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_S\}$ , #clusters $K$, #Lloyd's
    iterations $T_{init}$, and random projection dimension $P$.
Output: Set of prototypes $\mathbf{C} = \{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_K\}$.
1: $\mathbf{R}_i \leftarrow$ for each subset $\mathbf{X}_i$ generate $M \times P$ random matrix.
2: $\widetilde{\mathbf{X}}_i \leftarrow$ for each subset $\mathbf{X}_i$ compute $\frac{1}{\sqrt{P}}\mathbf{X}_i\mathbf{R}_i$
3: $\widetilde{\mathbf{C}}_i \leftarrow$ for each $\widetilde{\mathbf{X}}_i$ run K-means||.
4: $\mathbf{I}_i \leftarrow$ for each $\widetilde{\mathbf{X}}_i$ run $T_{init}$ Lloyd's iterations initialized with
    $\widetilde{\mathbf{C}}_i$.
5: For each partitioning $\mathbf{I}_i$ compute prototypes $\mathbf{C}_i$ in original
    space $\mathbf{X}_i$.
6: Compute local SSE for each $\mathbf{C}_i$ in $\mathbf{X}_i$.
7: $\mathbf{C} \leftarrow$ select prototypes corresponding to smallest local
    SSE.
```

The convergence rate of K-means is fast and the most significant improvements in the clustering error are achieved during the first few iterations (Bottou and Bengio, 1995; Broder et al., 2014). Therefore, for the initialization purposes, T_{init} can restricted, e.g., to 5 iterations. Moreover, since the number of Lloyd's iterations needed for convergence might vary significantly (e.g., Hämäläinen et al. (2017)), a restriction on the number of Lloyd's iterations helps in synchronization, when a parallel implementation of the SK-means|| method is used.

The computational complexity of K-means|| is of the order $\mathcal{O}(rlNM)$, where r is the number of initialization rounds. Therefore, SK-means|| also has the complexity of the order $\mathcal{O}(rlNM)$ in Step 1. In addition, SK-means|| runs T_{init} Lloyd's iterations with the complexity of $\mathcal{O}(T_{init}KNM)$, and computes local SSE with the complexity of $\mathcal{O}(KNM)$. Hence, the total complexity of SK-means|| is of the order $\mathcal{O}(rlNM + T_{init}KNM)$.

3.2. *SRPK-means||*

The second novel proposal, Subset Random Projection K-means|| (SRPK-means||), adds RPs to SK-means||. Since SK-means|| mainly uses time in computing distances in Steps 1 and 2, it is reasonable to speed up the distance computation with RP. The RP based method is presented in Algorithm 3. Generally, SRPK-means|| computes a set of candidate initial prototypes in a lower dimensional space and then evaluates these in the original space. Similarly to Algorithm 2, the best set of prototypes based on the local SSE are selected.

The proposal first computes a unique random matrix for each subset $\mathbf{X}_i$. Then, the P dimensional random projected subset $\widetilde{\mathbf{X}}_i$ is computed in the each subset $\mathbf{X}_i$. Steps 3–4 are otherwise the same as the Steps 1–2 in Algorithm 2, but these steps are applied for the lower dimensional subsets $\{\widetilde{\mathbf{X}}_1, \widetilde{\mathbf{X}}_2, ..., \widetilde{\mathbf{X}}_S\}$. Next, the labels $\mathbf{I}_i$ for partitioning each subset are used to compute

$\mathbf{C}_i$ in the original space $\mathbf{X}_i$. Finally, the local SSEs are computed and the best set of prototypes are returned as the initial prototypes. Note that the last two steps in Algorithm 3 are the same as Steps 3–4 in Algorithm 2. SRPK-means|| computes projected data matrices, which require a complexity of $\mathcal{O}(PNM)$ (naive multiplication) (Boutsidis et al., 2010). Execution of K-means|| in the lower dimensional space requires $\mathcal{O}(rlNP)$, and T_{init} Lloyd's iterations requires $\mathcal{O}(T_{init}KNP)$ operations. Step 6 requires $\mathcal{O}(KNM)$ operations, since it computes the local SSEs in the original space, so that the total computational complexity of the SRPK-means|| method is $\mathcal{O}(PNM + rlNP + T_{init}KNP + KNM)$. Typically, applications of RP are based on the assumption $P << M$. Thus, when the dimension of data M is increased, the contribution of the second and the third term of the total computational complexity start to diminish. Moreover, when both M and K are large compared to P, the last term dominates the overall computational complexity. Therefore, in terms of running time, SRPK-means|| is especially suited for clustering large-scale data with very high dimensionality into a large number of clusters.

Fern and Brodley (2003) noted that clustering with RP produces highly unstable and diverse clustering results. However, this can be exploited in clustering to find different candidate structures of data, which then can be combined into a single result (Fern and Brodley, 2003). The proposed initialization method in this paper uses a similar idea as it tries to find structures from multiple lower dimensional spaces that minimize the local SSE. In addition, selecting a result that gives the smallest local SSE excludes the bad structures, which could be caused by inappropriate $\mathbf{R}_i$ or $\mathbf{C}_i$.

4. Parallel implementation of proposed algorithms

Bahmani et al. (2012) implemented K-means|| with the MapReduce programming model. It can also be implemented by the Single Program Multiple Data (SPMD) programming model with message passing. Then all the steps of the parallelized Algorithms 1, 2, and 3 are executed inside an SPMD block. Next, a parallel implementation of K-means|| as depicted in Algorithm 1 is briefly described, by using Matlab Parallel Computing Toolbox (PCT), SPMD blocks, and message passing functions (see Sharma and Martin (2009) for a detailed description about PCT). First, data $\mathbf{X}$ is split into Q subsets of approximately equal size and then the subsets are distributed to Q workers. Step 1 picks a random point from a random worker and broadcasts this point to all other workers. In Step 2, each worker calculates distances and SSE for its local data. Next, points are aggregated by calling *gplus*-function, after which the aggregation distributes this sum to other workers. In Steps 4–5, each worker samples points from its local data, the next points are aggregated to $\mathbf{C}'$ by calling *gop*-function, and then $\mathbf{C}'$ is broadcasted to all workers. Again, distances and SSE are calculated similarly as in Step 2. Each worker in Step 6 assigns weights based on its local data, after which the weights are aggregated with *gop*-function. Finally, Step 7 is computed sequentially.

Similarly to the parallel K-means|| implementation, a parallel implementation of Algorithm 2 with SMPD and message passing is described next. First, each subset $\mathbf{X}_i$ from S subsets is split into J approximately equal size subsets and then these subsets are distributed to $J \times S$ workers, e.g., subset $\mathbf{X}_i$ is distributed to workers $(i-1)J+1, ..., (i-1)J+J$. In Steps 1–3, each subset of workers runs steps for subset $\mathbf{X}_i$ in parallel similarly as described in the previous paragraph. For parallel Lloyd's iterations, a similar strategy as proposed in Dhillon and Modha (1999) can be used in Step 2. Steps 1–3 require calling modified *gop*-function and *gplus*-function for the subset of workers; these functions were modified to support this requirement. Finally, prototypes corresponding to the smallest local SSE from the subset i' allocated workers $(i' - 1)J + 1, ..., (i' - 1)J + J$ are returned as the initialization.

The parallel SRPK-means|| in Algorithm 3 can be implemented in a highly similar fashion to the parallel SK-means||. More precisely, in Step 1, each worker $(i - 1)J + 1$, where $i \in \{1, 2, ..., S\}$, generates the random matrix $\mathbf{R}_i$ and broadcasts it to workers $(i-1)J+1, ..., (i-1)J+J$. In Step 2, each worker computes random projected data for its local data. Steps 3–4 are otherwise computed similarly to the parallel SK-means|| Steps 1–2, except these steps are executed for the projected subsets. In Step 5, the prototypes are computed in the original space in parallel. Finally, Steps 6–7 are the same as Steps 3–4 in Algorithm 2.

5. Empirical evaluation of proposed algorithms

In this section, empirical comparison between K-means++, K-means||, SK-means||, and SRPK-means|| is presented by using 13 real and two synthetic datasets. Parallel implementations of the proposed methods and K-means|| were applied to the seven largest datasets and serial implementations were used with the eight smallest datasets. K-means++ was used only for the eight smallest datasets. The performance of the methods was evaluated by analyzing SSE, the number of iterations needed for convergence, and the running time.

5.1. Experimental setup

Table 1: Characteristics of datasets

Dataset	N	M	K
HAR	7 352	561	6
ISO	7 797	617	26
LET	20 000	16	26
GFE	27 936	300	36
MNI	70 000	784	10
BIR	100 000	2	100
BSM	583 250	77	50*
FCT	581 012	54	7
SVH	630 420	3 072	100*
RCV	781 265	1 000	350
USC	2 458 285	68	100*
KDD	4 898 431	41	100*
M8M	8 100 000	784	265
TIN	15 860 403	384	100*
OXB	16 334 970	128	100*

Basic information about the datasets is shown in Table 1. For the serial experiments, we used the following eight datasets: Human Activity Recognition Using Smartphones[1] (HAR), ISOLET[1] (ISO), Letter Recognition[1] (LET), Grammatical Facial Expressions[1] (GFE), MNIST[2] (MNI), Birch3[3] (BIR), Buzz in Social Media[1] (BSM), and Covertype[1] (COV). For the parallel experiments, we chose seven large high-dimensional datasets: Street View House Numbers[4] (SVH), RCV1v2 collection of documents[5] (RCV), US Census Data 1990[1] (USC), KDD Cup 1999 Data[1] (KDD), MNIST8M [5] (M8M), Tiny Images[6] (TIN), and Oxford Buildings[7] (OXB). The BIR dataset (Fränti and Sieranoja, 2018) was selected to test SK-means|| for low dimensional data. With the OXB dataset, we utilized transformed dataset with 128-dimensional SIFT descriptors extracted from the original dataset. For the TIN dataset, we sampled a 20 percent subset from the Gist binary file (`tinygist80million.bin`), where 79302017 images are characterized with 384 dimensional Gist descriptors. The highest dimensional dataset was SVH, where we combined the training, testing, and validation subsets into a single dataset. We excluded the attack type feature (class label) from the KDD dataset. We used the Twitter data for the BSM dataset and the training dataset for the HAR dataset in the experiments. For the RCV dataset, we used the full industries test set (350 categories) and selected 1000 out 47236 features with the same procedure as in (De Vries and Geva, 2009). For the M8M and the RCV datasets, we used the scaled datasets given in[5], all other datasets were min-max scaled into $[-1, 1]$.

For the experiments, each dataset was randomly divided into 8 subsets, which were roughly of equal size. The experiments were run in Matlab R2014a environment. The parallel algorithms were implemented with Matlab Parallel Computing Toolbox with the SPMD blocks and message passing functions as discussed in Section 4. The parallel experiments were run in a Taito computer cluster utilizing Sandy Bridge nodes with 16 cores and 256 GB memory. A parallel pool of 32 workers was used in the experiments; therefore, 4 workers were allocated for each subset.

For all datasets we used the following settings: *i*) for K-means||: $l = 2K$ and $r = 5$; *ii*) for SK-means|| and SRPK-means||: $T_{init} = 5$ and $S = 8$; *iii*) and for SRPK-means||: $P \in \{5, 10, 20, 40\}$ and R with $r_{ij} = \pm 1$. After initialization, the Lloyd's algorithm was executed until the number of new assignments between the consecutive iterations was below or equal to the threshold. For the five largest datasets (SVH, RCV, OXB, M8M and TIN) we set this threshold to 1% of N and otherwise to zero. In the parallel experiments, runs were repeated 10 times for each setting. In the serial experiments, runs were repeated 100 times for each setting. Values for the number of clusters, K, are given in the last column of Table 1. Since the MNIST8M dataset is constructed artificially from the original MNIST dataset, we set K for MNIST8M based on the optimal value for MNIST used by Gallego et al. (2018). Otherwise, the selection is either based on the known number of classes or fixed arbitrarily (indicated with * in Table 1).

The clustering result quality between the methods was compared by using SSE. The SSE values were computed with formula (1) for the whole data. Finally, statistical comparison between the methods was performed with the nonparametric Kruskal-Wallis test (Kruskal and Wallis, 1952; Saarela et al., 2017), since in most of the cases the clustering errors were not normally distributed. The significance level was set to 0.05.

5.2. Results

In this section, the results are shown and analyzed by considering separately the accuracy (Section 5.2.1), efficiency (Section 5.2.2), and scalability (Section 5.2.3) of the algorithms.

5.2.1. Clustering accuracy

SSE values after the initialization (initial SSE) and after the Lloyd's iterations (final SSE) for K-means++ (K++), K-means|| (K||), SK-means|| (SK||), and SRPK-means|| (SRPK||) methods are summarized in Table 4, where $**$ in the first column refers to a statistically significant difference between the methods for the final SSE. For the initial SSE, we did not include any results from the statistical testing because of differences in variances. In addition, + indicates that the method has statistically significantly better SSE compared to K-means++, $*$ indicates that the method has statistically significantly better SSE compared to K-means||, $\dagger_{P'}$ indicates that the method has statistically significantly better SSE compared to SRPK-means|| when $P = P'$, and ‡ indicates that the method has statistically significantly better SSE compared to SK-means||. The coefficient under the name of the data in the first column is the data-specific multiplier which scales the SSE to the true level. Moreover, note that the assumption of equal variances underlying the Kruskal-Wallis test, for the final SSE, is satisfied only for SVH, RCV, USC, KDD, OXB, and M8M, based on the Brown-Forsythe test. The assumption of equal variances for the final SSE is not satisfied for HAR, ISO, LET, GFE, MNI, BIR, BSM, FCT, and TIN, based on the Brown-Forsythe test.

Clearly, SK-means|| and SRPK-means|| outperform K-means|| and K-means++, in terms of the initial SSE. SRPK-means|| with $P = 40$ reaches almost the same initialization SSE level as SK-means||. For the six largest datasets, SRPK-means|| with $P = 20$ always had smaller max-value of SSE after initialization than the min-value of K-means||. K-means++ has about two times larger initial SSE than SK-means|| and SRPK-means|| ($P = 40$). Overall, in terms of the final clustering error, SRPK-means|| achieved better final SSE than K-means|| and K-means++ with only few exceptions. SK-means|| gives better final SSE than the baseline methods when $M < 100$. Otherwise, the final SSE for SK-means|| is better or equal compared to the baseline methods. One can notice that the final SSE is highly similar for K-means++ and K-means||, statistical testing indicates that there is no difference. Note that in Table 4 the min-values of all methods for the final SSE are equal for small number of clusters ($K < 11$). This is probably due to the fact that the smaller number of possible partitions (Äyrämö, 2006)

[1] http://archive.ics.uci.edu/ml/index.php

[2] http://yann.lecun.com/exdb/mnist/

[3] http://cs.joensuu.fi/sipu/datasets/

[4] http://ufldl.stanford.edu/housenumbers/

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

[6] http://horatio.cs.nyu.edu/mit/tiny/data/

[7] http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/

Table 2: Running time for the initialization

data	M	K||	SK||	SRPK||	
				$p = 5$	$p = 40$
KDD	41	**5.0**	8.7	7.9	10.2
USC	68	**3.0**	5.0	3.9	5.6
OXB	128	26.0	39.1	**23.7**	27.4
TIN	384	52.1	65.8	**24.9**	28.5
M8M	784	98.5	145.4	**32.2**	37.9
RCV	1000	14.1	20.6	**4.8**	5.6
SVH	3,072	13.7	17.3	3.4	**3.3**

Table 3: #iterations

data	K++	K||	SK||	SRPK||			
				$p = 5$	$p = 10$	$p = 20$	$p = 40$
HAR**	28.5	34	**18**$^{+*\dagger_5}$	26*	21^{+*}	20^{+*}	21.5^{+*}
ISO**	32	32	**27**$^{+*\dagger_{5,10}}$	36	33	31	28^{†_5}
LET**	82	78	**65**$^{+*\dagger_5}$	75.5	72.5	-	-
GFE**	60	58.5	**50.5**$^{+}$	55.5	54	51	52
MNI	80	82	**64**	83.5	82	74.5	75.5
BIR**	97	**84**	88^{+}	-	-	-	-
BSM**	36	33.5	28^{+}	31.5	30.5	**25.5**$^{+*\dagger_5}$	31^{+}
FCT**	14.5	8	**1**$^{+*\dagger_{5-20}}$	6^{+}	6$^{+*\dagger_{5-20}}$	4$^{+*\dagger_5}$	3$^{+*\dagger_{5,10}}$
SVH**	-	32.5	28.5^{†_5}	35	32	30	**27**^{†_5}
RCV**	-	20	19	20.5	20	**17**^{*†_5}	**17**$^{*\dagger_{5,10}}$
USC	-	**81**	86	93.5	83.5	77	98.5
KDD	-	82	72	96.5	78	**69**	70
M8M**	-	31	**24**$^{*\dagger_{5-20}}$	33	31.5	30	28.5^{†_5}
TIN**	-	37.5	**33**$^{*\dagger_{5-20}}$	40	38.5	39	35.5
OXB**	-	27.5	**22**$^{*\dagger_{5-40}}$	28.5	28	27.5	28.5

implies a smaller number of local minima compared to higher values of K.

One can note from Table 4 that the proposed approaches are especially better than K-means|| for the datasets where the variability of the obtained clustering results (median absolute deviation/mad-values in Table 4) is high for K-means||. The highest mad-values for K-means|| are for the BSM and FCT datasets. This indicates that, when the obtained clustering results for K-means|| are highly different from each other, an evaluation of the initial prototypes prior to the K-means refinement is beneficial, in the way it is conducted with the proposed methods. For the proposed methods, the oversampling factor l could probably be tuned to lower values with smaller effect to the clustering quality compared to K-means||, which would speed up the initialization phase.

5.2.2. *Initialization running time and convergence*

Running time for the initialization (median of 10 runs) for the parallel experiments is shown in Table 2. Running time for the initialization taken by K-means|| is around 60% − 80% of the running time of SK-means||. SRPK-means|| runs clearly faster than SK-means|| for datasets with dimensionality more than 100, and for the four highest dimensional datasets, SRPK-means|| runs clearly faster than K-means||. Note that differences are small between $P = 5$ and $P = 40$ for SRPK-means||.

The median number of Lloyd's iterations needed for convergence after the initialization phase are summarized in Table 3, where statistically significant differences are denoted similarly as in Table 4. The assumption of equal variances was satisfied for all datasets except for FCT. In general, SK-means|| seems to a require smaller number of Lloyd's iterations than K-means++ and K-means||, which directly translates to faster running time of the K-means search. Based on the statistical testing, SRPK-means|| is better than or equal compared to the baseline methods in terms of the number of iterations. Therefore, SRPK-means|| can also speed up the search phase of the K-means clustering method. Increasing the RP dimension from 5 to 40 further improved the speed of convergence for SRPK-means||. Out of the parameter values used in the experiments, selecting $P = 40$ gives the best trade off between the running time and the clustering error for SRPK-means||. Furthermore, note that there is no statistical difference between K-means++ and K-means|| with respect to the number of Lloyd's iterations.

5.2.3. *Scalability*

We conducted scalability tests for TIN and SVH to show how running time varies as a function of #processing elements (Matlab workers) and to demonstrate the benefits of using SRPK-means|| for a very high-dimensional dataset (SVH) when K is increased. We concentrated on the running time of the initialization and the corresponding SSE, which are the main focus of the work. We performed scalability experiments in two parts: 1) Tests with TIN: #processing elements was varied from 8 to 64 and $K = 100$ was fixed; 2) Tests with SVH: The number of clusters was varied as $K \in \{100, 200, 400, 800\}$ and #processing elements was fixed to 32. Otherwise, we used the same parameter settings as in the previous experiments.

Median running time and SSE curves out of 10 runs are shown in Figure 1. Results for the experiment 1 are shown in Figure 1a. In terms of Amdahl's law, K-means|| and SK-means|| perform equally well: running time is nearly halved when #processing elements is doubled from 8 to 16 and from 16 to 32. In this perspective, performance of SRPK-means|| is slightly worse than K-means|| and SK-means||. The results for the experiment 2 are shown in Figure 1b–1c. Clearly, for very high-dimensional data, SRPK-means|| runs much faster compared to K-means|| and SK-means||. As analyzed in Section 3.2, the speedup for SRPK-means|| is increased when K is increased. A similar observation was made between K-means++ and RPK-means++ in Chan and Leung (2017). Furthermore, when $K = 800$, the speedup for SRPK-means|| with respect to K-means|| is 7–8, and with respect to SK-means|| speedup is 9–10. Moreover, according to Figure 1c, SRPK-means|| (when $P = 40$) and SK-means|| sustain their accuracy when K is increased in a frame of K-means||.

6. Conclusion

In this paper, we proposed two parallel initialization methods for large-scale K-means clustering. The methods are based on divide-and-conquer type of K-means|| approach and random projections. The proposed initialization methods are scalable and fairly easy to implement with different parallel programming models. Sequential implementations of the proposed methods could also be beneficial for clustering initialization if K-means|| is replaced with K-means++.

The experimental results for 15 datasets showed that the proposed methods improve clustering error and the speed of convergence compared to state-of-the-art performance. Experiments with SRPK-means|| method demonstrate that utilization of RP and K-means|| is beneficial for clustering large-scale high dimensional datasets. In particular, SRPK-means|| is an appealing approach as a standalone algorithm for clustering very high-dimensional large-scale datasets.

In future work, it would interesting test to a RP based local SSE selection for SRPK-means||, which uses the same RP matrix in each subset for the initial prototype selection. In this case, utilization of sparse RP variants (Li et al., 2006) or the mailman algorithm (Boutsidis et al., 2010) for the matrix multiplication could be beneficial, particularly in applications where K is close to P. Furthermore, integrating the proposed methods into the robust K-means|| (Hämäläinen et al., 2018) would be beneficial for clustering noisy data, because the final clustering error can have a large variance in these cases.

Acknowledgments

The work of TK has been supported by the Academy of Finland from the projects 311877 (Demo) and 315550 (HNP-AI).

References

Achlioptas, D., 2003. Database-friendly random projections: Johnson-lindenstrauss with binary coins. Journal of Computer and System Sciences 66, 671 – 687. Special Issue on {PODS} 2001.

Alzate, C., Suykens, J.A., 2010. Multiway spectral clustering with out-of-sample extensions through weighted kernel pca. IEEE transactions on pattern analysis and machine intelligence 32, 335–347.

Arthur, D., Vassilvitskii, S., 2007. k-means++: The advantages of careful seeding, in: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics. pp. 1027–1035.

Äyrämö, S., 2006. Knowledge Mining Using Robust Clustering. volume 63 of Jyväskylä Studies in Computing. University of Jyväskylä.

Bachem, O., Lucic, M., Krause, A., 2017. Distributed and provably good seedings for k-means in constant rounds, in: International Conference on Machine Learning, pp. 292–300.

Bahmani, B., Moseley, B., Vattani, A., Kumar, R., Vassilvitskii, S., 2012. Scalable k-means++. Proc. VLDB Endow. 5, 622–633. doi:10.14778/2180912.2180915.

Bingham, E., Mannila, H., 2001. Random projection in dimensionality reduction: Applications to image and text data, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM. pp. 245–250.

Bottou, L., Bengio, Y., 1995. Convergence properties of the k-means algorithms, in: Advances in neural information processing systems, pp. 585–592.

Boutsidis, C., Zouzias, A., Drineas, P., 2010. Random projections for k-means clustering. CoRR .

Boutsidis, C., Zouzias, A., Mahoney, M.W., Drineas, P., 2015. Randomized Dimensionality Reduction for k -Means Clustering. Information Theory, IEEE Transactions on 61, 1045–1062.

Bradley, P.S., Fayyad, U.M., 1998. Refining initial points for k-means clustering., in: ICML, pp. 91–99.

Broder, A., Garcia-Pueyo, L., Josifovski, V., Vassilvitskii, S., Venkatesan, S., 2014. Scalable k-means by ranked retrieval, in: Proceedings of the 7th ACM international conference on Web search and data mining, ACM. pp. 233–242.

Cardoso, Â., Wichert, A., 2012. Iterative random projections for high-dimensional data clustering. Pattern Recognition Letters 33, 1749 – 1755.

Chan, J.Y., Leung, A.P., 2017. Efficient k-means++ with random projection, in: Neural Networks (IJCNN), 2017 International Joint Conference on, IEEE. pp. 94–100.

Cohen, M.B., Elder, S., Musco, C., Musco, C., Persu, M., 2014. Dimensionality reduction for k-means clustering and low rank approximation. CoRR .

De Vries, C.M., Geva, S., 2009. K-tree: large scale document clustering, in: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, ACM. pp. 718–719.

Dhillon, I.S., Modha, D.S., 1999. A data-clustering algorithm on distributed memory multiprocessors. LargeScale Parallel Data Mining 1759, 245–260.

Elkan, C., 2003. Using the triangle inequality to accelerate k-means, in: Proceedings of the 20th International Conference on Machine Learning (ICML-03), pp. 147–153.

Emre Celebi, M., Kingravi, H.A., Vela, P.A., 2012. A comparative study of efficient initialization methods for the k-means clustering algorithm. Expert Systems with Applications .

Fern, X., Brodley, C., 2003. Random projection for high dimensional data clustering: A cluster ensemble approach, in: Proceedings of the 20th International Conference on Machine Learning, pp. 186–193.

Forgy, E.W., 1965. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. Biometrics 21, 768–769.

Fränti, P., Sieranoja, S., 2018. K-means properties on six clustering benchmark datasets. Applied Intelligence , 1–17.

Gallego, A.J., Calvo-Zaragoza, J., Valero-Mas, J.J., Rico-Juan, J.R., 2018. Clustering-based k-nearest neighbor classification for large-scale data with neural codes representation. Pattern Recognition 74, 531–543.

Hämäläinen, J., Jauhiainen, S., Kärkkäinen, T., 2017. Comparison of internal clustering validation indices for prototype-based clustering. Algorithms 10, 105.

Hämäläinen, J., Kärkkäinen, T., 2016. Initialization of big data clustering using distributionally balanced folding, in: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2016, pp. 587–592.

Hämäläinen, J., Kärkkäinen, T., Rossi, T., 2018. Scalable robust clustering method for large and sparse data, in: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2018, pp. 449–454.

Hamerly, G., 2010. Making k-means even faster, in: Proceedings of the 2010 SIAM international conference on data mining, SIAM. pp. 130–140.

Johnson, W.B., Lindenstrauss, J., 1984. Extensions of lipschitz mappings into a hilbert space. Contemporary mathematics 26, 1.

Jolliffe, I.T., 2002. Principal Component Analysis. 2nd ed.. Springer.

Kärkkäinen, T., Heikkola, E., 2004. Robust formulations for training multilayer perceptrons. Neural Computation 16, 837–862.

Kruskal, W.H., Wallis, W.A., 1952. Use of ranks in one-criterion variance analysis. Journal of the American statistical Association 47, 583–621.

Li, P., Hastie, T.J., Church, K.W., 2006. Very sparse random projections, in: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM. pp. 287–296.

Liu, H., Motoda, H., 1998. Feature Selection for Knowledge Discovery and Data Mining. Kluwer Academic Publishers, Norwell, MA, USA.

Lloyd, S., 1982. Least squares quantization in PCM. IEEE Transactions on Information Theory 28, 129–137. doi:10.1109/TIT.1982.1056489.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations, in: Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, Oakland, CA, USA.. pp. 281–297.

Moreno-Torres, J.G., Sáez, J.A., Herrera, F., 2012. Study on the impact of partition-induced dataset shift on k-fold cross-validation. IEEE Transactions on Neural Networks and Learning Systems 23, 1304–1312.

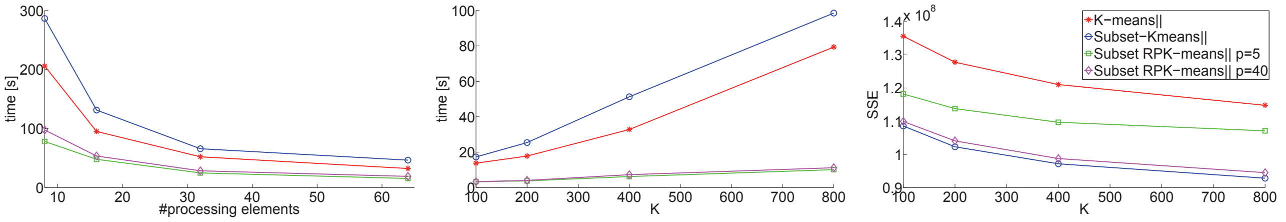
Napoleon, D., Pavalakodi, S., 2011. A new method for dimensionality reduction using k-means clustering algorithm for high dimensional data set. International Journal of Computer Applications 13, 41–46.

Saarela, M., Hämäläinen, J., Kärkkäinen, T., 2017. Feature ranking of large, robust, and weighted clustering result, in: Proceedings of 21st Pacific Asia Conference on Knowledge Discovery and Data Mining - PAKDD 2017, pp. 96–109.

Sharma, G., Martin, J., 2009. Matlab: A language for parallel computing. International Journal of Parallel Programming 37, 3–36.

Xu, Y., Qu, W., Li, Z., Min, G., Li, K., Liu, Z., 2014. Efficient k -means++ approximation with mapreduce. IEEE Transactions on Parallel and Distributed Systems 25, 3135–3144.

Zhao, W., Ma, H., He, Q., 2009. Parallel k-means clustering based on mapreduce, in: Proceedings of the 1st International Conference on Cloud Computing, pp. 674–679.



(a)

(b)

(c)

Fig. 1: Scalability with respect to #processing elements and the number of clusters K.

Table 4: Clustering accuracy using SSE

		Initialization							Final						
		K++	K||	SK||	SRPK||				K++	K||	SK||	SRPK||			
data	stats				p = 5	p = 10	p = 20	p = 40				p = 5	p = 10	p = 20	p = 40
HAR**	median	2.9267	1.7504	**1.4082**	1.4878	1.4523	1.4371	1.4222	1.3803	1.3803	**1.3707***	**1.3707**$^{+*}$	**1.3707**$^{+*}$	**1.3707***	**1.3707**$^{+*}$
10^5	mad	0.3839	0.2063	**0.0519**	0.0640	0.0554	0.0672	0.0631	0.0247	0.0269	**0.0059**	0.0109	0.0088	0.0074	0.0095
	max	5.6183	4.0399	1.9232	1.7310	**1.6624**	1.6734	1.7074	1.5442	1.5442	**1.4094**	1.4126	1.4126	**1.4094**	1.4144
	min	2.4219	1.5181	**1.3841**	1.4016	1.3860	1.3882	1.3900	**1.3707**	**1.3707**	**1.3707**	**1.3707**	**1.3707**	**1.3707**	**1.3707**
ISO**	median	10.1969	6.0737	**5.2497**	6.2368	5.8513	5.5509	5.4086	4.7846	4.7777	4.7679	4.7590^{+*}	4.7650^{+}	**4.7555**$^{+*\ddagger}$	4.7590^{+*}
10^5	mad	0.2874	0.2073	0.1475	0.1510	**0.1360**	0.1582	0.1493	0.0281	0.0285	**0.0215**	0.0301	0.0338	0.0236	0.0320
	max	11.1637	7.2736	6.0510	6.7378	6.2192	6.1989	**5.8519**	4.8955	4.9124	4.8565	4.8428	4.8506	**4.8176**	4.8601
	min	9.3376	5.6181	**4.9746**	5.8912	5.5547	5.2810	5.0959	4.7175	4.7168	4.7178	4.7109	4.7166	4.7156	**4.7106**
LET**	median	2.6293	1.6243	**1.3071**	1.6730	1.5067	-	-	1.1017	1.1003	**1.0975**$^{+*}$	1.0989	1.0986	-	-
10^4	mad	0.1548	0.1208	**0.0603**	0.0844	0.0795	-	-	0.0060	0.0052	**0.0040**	0.0071	0.0059	-	-
	max	3.8036	2.1248	**1.5182**	1.9576	1.7570	-	-	1.1187	1.1192	**1.1126**	1.1267	1.1171	-	-
	min	2.1253	1.3782	**1.1751**	1.5141	1.3597	-	-	1.0884	**1.0864**	1.0871	1.0883	1.0891	-	-
GFE**	median	3.9497	2.5022	**2.1087**	2.4164	2.2760	2.1977	2.1448	1.8621	1.8585	1.8469^{+*}	**1.8386**$^{+*}$	1.8389^{+*}	1.8411^{+*}	1.8438^{+*}
10^5	mad	0.2145	0.1072	**0.0600**	0.0716	0.0882	0.0727	0.0704	0.0183	0.0140	**0.0109**	0.0131	0.0124	0.0129	0.0140
	max	4.9085	3.0283	2.3899	2.6221	2.7836	2.4177	**2.3769**	1.9594	1.9105	1.8839	1.8845	**1.8678**	1.8846	1.9009
	min	3.5318	2.2229	**1.9762**	2.2831	2.1540	2.0387	2.0012	1.8274	1.8175	1.8187	1.8173	1.8179	**1.8165**	1.8170
MNI**	median	2.3625	1.4722	**1.1424**	1.2918	1.2601	1.2194	1.1917	1.1013	1.0980	**1.0979**$^{+}$	**1.0979**$^{+}$	**1.0979**$^{+}$	1.0980^{+}	**1.0979**$^{+}$
10^7	mad	0.1047	0.0819	**0.0246**	0.0282	0.0274	0.0308	0.0315	0.0025	0.0027	**0.0018**	0.0027	0.0020	0.0021	0.0022
	max	2.9160	1.8498	**1.2373**	1.4228	1.3197	1.2963	1.2722	1.1076	1.1111	1.1069	1.1108	**1.1045**	1.1070	1.1052
	min	2.1211	1.2972	**1.1068**	1.2464	1.2027	1.1445	1.1355	**1.0977**	**1.0977**	**1.0977**	**1.0977**	**1.0977**	**1.0977**	**1.0977**
BIR**	median	5.4011	3.0420	**2.4033**	-	-	-	-	1.8137	1.8151	**1.7739**$^{+*}$	-	-	-	-
10^2	mad	0.5161	0.3804	**0.2313**	-	-	-	-	0.0423	0.0440	**0.0231**	-	-	-	-
	max	7.3151	5.8659	**3.4308**	-	-	-	-	2.0110	2.0894	**1.8452**	-	-	-	-
	min	4.1618	2.2741	**1.9868**	-	-	-	-	1.7347	1.7188	**1.7074**	-	-	-	-
BSM**	median	2.0774	1.2901	**1.1001**	1.5480	1.2238	1.1195	1.1129	1.1430^{†_5}	1.1933	1.0775$^{+*\dagger_{5,10}}$	1.2228	1.1215^{*†_5}	1.0833$^{+*\dagger_{5,10}}$	**1.0681**$^{+*\dagger_{5,10}}$
10^5	mad	0.4395	0.1890	**0.1797**	0.3356	0.3064	0.2126	0.2255	0.0688	0.0769	**0.0419**	0.1190	0.0849	0.0561	0.0437
	max	10.3542	3.6431	6.5058	4.1810	3.6241	2.4254	**2.2779**	1.4277	1.4808	1.2475	1.5491	1.3860	1.2429	**1.1740**
	min	1.5767	1.0420	0.9558	1.1611	0.9912	**0.9443**	0.9843	0.9633	1.0344	0.9516	0.9999	**0.9160**	0.9392	0.9570
FCT**	median	4.2565	2.2745	**1.9369**	2.2606	2.1251	2.0149	1.9744	1.9665	2.0096	**1.9215**$^{+*\dagger_{5-20}}$	1.9796	1.9645*	1.9408$^{+*\dagger_5}$	1.9359$^{+*\dagger_{5,10}}$
10^6	mad	0.3461	0.1561	**0.0341**	0.0898	0.0651	0.0625	0.0435	0.0727	0.0793	**0.0299**	0.0544	0.0559	0.0536	0.0440
	max	6.0590	2.7642	**2.0613**	2.4897	2.3442	2.2716	2.0842	2.3120	2.3666	**1.9887**	2.1125	2.1264	2.1105	2.0273
	min	3.5876	1.9638	**1.8645**	2.0662	2.0013	1.8657	1.8652	**1.8644**	**1.8644**	**1.8644**	**1.8644**	**1.8644**	**1.8644**	**1.8644**
SVH	median	-	1.3559	**1.0855**	1.1820	1.1464	1.1155	1.0992	-	**1.0703**	1.0704	1.0704	1.0705	1.0706	1.0708
10^8	mad	-	0.0636	**0.0014**	0.0185	0.0149	0.0075	0.0042	-	**0.0003**	0.0004	0.0005	0.0005	**0.0003**	**0.0003**
	max	-	1.5968	**1.0889**	1.2279	1.1765	1.1290	1.1083	-	**1.0711**	1.0720	**1.0711**	1.0717	1.0714	1.0712
	min	-	1.3027	**1.0836**	1.1639	1.1246	1.1082	1.0922	-	**1.0696**	1.0698	1.0698	1.0700	1.0703	1.0703
RCV**	median	-	2.5506	**2.1405**	2.5897	2.4864	2.3575	2.2313	-	2.0876	2.0913	2.0780	2.0757*‡	**2.0702**^{*‡†_5}	2.0715^{*‡†_5}
10^5	mad	-	0.0233	**0.0022**	0.0138	0.0041	0.0045	0.0040	-	0.0027	0.0018	0.0019	**0.0014**	0.0026	0.0022
	max	-	2.5886	**2.1427**	2.5979	2.4945	2.3652	2.2363	-	2.0922	2.0951	2.0823	2.0778	2.0767	**2.0755**
	min	-	2.4849	**2.1345**	2.5647	2.4830	2.3523	2.2228	-	2.0812	2.0863	2.0764	2.0737	2.0688	**2.0674**
USC**	median	-	1.7014	**1.1936**	1.5530	1.3218	1.2284	1.1989	-	1.1903	1.1779	1.1736*	**1.1688***	1.1718*	1.1709*
10^7	mad	-	0.0394	**0.0036**	0.0338	0.0217	0.0079	0.0037	-	0.0070	0.0057	0.0072	0.0091	0.0091	**0.0049**
	max	-	1.7671	**1.2016**	1.6178	1.3542	1.2415	1.2038	-	1.2020	1.1908	**1.1803**	1.1841	1.1869	1.1829
	min	-	1.6110	**1.1877**	1.4957	1.2799	1.2191	1.1934	-	1.1781	1.1712	1.1595	**1.1566**	1.1602	1.1667
KDD**	median	-	30.5335	2.5466	3.1781	2.6651	2.5817	**2.5162**	-	2.5218	2.4726	2.4853	2.4582*	2.4755	**2.4529***
10^5	mad	-	7.3666	**0.0337**	0.1019	0.0483	0.0562	0.0440	-	0.0372	0.0419	0.0698	0.0413	0.0464	**0.0316**
	max	-	58.0452	**2.5962**	3.3024	2.7083	2.6275	2.6367	-	2.6288	2.5432	2.6241	**2.5223**	2.5640	2.5406
	min	-	22.2667	2.4803	2.9935	2.5447	**2.4483**	2.4878	-	2.4614	2.4084	2.3961	**2.3927**	2.4032	2.4330
M8M**	median	-	2.6631	**2.2390**	2.9313	2.6415	2.4185	2.3153	-	2.2159	2.2154	**2.2139***	**2.2139***	2.2141*	**2.2139***
10^8	mad	-	0.0164	**0.0009**	0.0286	0.0150	0.0071	0.0041	-	0.0012	0.0008	0.0010	0.0009	**0.0005**	0.0009
	max	-	2.6959	**2.2412**	2.9835	2.6690	2.4299	2.3176	-	2.2203	2.2165	2.2162	2.2155	**2.2145**	2.2157
	min	-	2.6391	**2.2376**	2.8782	2.6192	2.4091	2.3043	-	2.2143	2.2131	2.2128	**2.2126**	2.2131	2.2132
TIN	median	-	10.7568	**8.8782**	9.7335	9.4194	9.2042	9.0595	-	8.8060	8.8065	**8.8058**	8.8075	8.8073	8.8073
10^7	mad	-	0.1498	**0.0057**	0.1054	0.0879	0.0338	0.0139	-	**0.0012**	0.0014	0.0016	0.0018	0.0016	0.0030
	max	-	11.1649	**8.8812**	9.9343	9.5627	9.2543	9.0841	-	8.8091	**8.8077**	8.8091	8.8106	8.8093	8.8108
	min	-	10.4721	**8.8623**	9.5824	9.3339	9.1431	9.0474	-	8.8031	8.8029	8.8042	8.8047	8.8044	**8.8024**
OXB**	median	-	1.7678	**1.5375**	1.6432	1.6249	1.6014	1.5829	-	1.5267	1.5268	**1.5254***‡	1.5256*‡	1.5255*‡	1.5255*‡
10^8	mad	-	0.0181	**0.0004**	0.0055	0.0058	0.0028	0.0017	-	0.0006	**0.0002**	0.0003	0.0005	0.0005	0.0004
	max	-	1.8224	**1.5384**	1.6575	1.6315	1.6082	1.5850	-	1.5286	1.5271	**1.5258**	1.5266	1.5262	1.5261
	min	-	1.7506	**1.5367**	1.6393	1.6162	1.5995	1.5797	-	1.5258	1.5259	1.5251	**1.5250**	**1.5250**	**1.5250**

PIII

SCALABLE ROBUST CLUSTERING METHOD FOR LARGE AND SPARSE DATA

by

Joonas Hämäläinen, Tommi Kärkkäinen and Tuomo Rossi 2018

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning

ESANN 2018 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 25-27 April 2018, i6doc.com publ., ISBN 978-287587047-6. Available from http://www.i6doc.com/en/.

Scalable robust clustering method for large and sparse data

Joonas Hämäläinen, Tommi Kärkkäinen* and Tuomo Rossi

University of Jyvaskyla, Faculty of Information Technology,
P.O. Box 35, FI-40014 University of Jyvaskyla, Finland

Abstract. Datasets for unsupervised clustering can be large and sparse, with significant portion of missing values. We present here a scalable version of a robust clustering method with the available data strategy. More precisely, a general algorithm is described and the accuracy and scalability of a distributed implementation of the algorithm is tested. The obtained results allow us to conclude the viability of the proposed approach.

1 Introduction

Clustering is one of the core techniques in unsupervised learning. Based on a similarity measure (e.g., Euclidean distance), its purpose is to partition a given data into groups, clusters, where members belonging to one cluster are similar to each other and dissimilar to other clusters. Classically, clustering is divided into two main categories, partitional and hierarchical, although a large variety of different approaches have been suggested [1, 2].

Since the real-world clustering problems are becoming larger and larger, applying sequential clustering algorithms to these problems becomes impractical. Over the years, a lot of research related to the parallellizing of the well-known K-means algorithm with various parallel computation models has been carried out [3, 4, 5]. K-means|| [6] is parallelizable version of the K-means++ [7]. Contrary to K-means++, imposed by the inherently sequential nature, K-means|| is scalalable and it can be easily implemented in parallel with multiple parallel programming models. As shown by [6], proper initialization of a parallel algorithm plays an important role both in accuracy and scalability.

K-spatialmedians is prototype-based clustering method which applies available data strategy and spatial median as cluster prototype [8]. The available data strategy refers to an approach, where all distance computations are projected to the available values. This ensures that no assumptions on the unknown distribution of the missing values (MVs) is being made during clustering. Robustness and accuracy of the approach for tens of percents of MVs was extensively tested in [9]. However, differently to the use of the mean as in K-means, one needs to apply an iterative method to compute the cluster prototype. Hence, scalability of the parallel implementation is not self-evident. Therefore, the purpose in this article is twofold: i) to compare clustering results between K-means and K-spatialmedians, ii) to consider scalability of a parallel implementation of K-spatialmedians

*The work of TK has been supported by the Academy of Finland from the projects 311877 (Demo) and 315550 (HNP-AI)

2 Parallel K-spatialmedians||

Let $\mathbf{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ denote a dataset in M dimensional space and let $\mathbf{P} = \{\mathbf{p}_1, ..., \mathbf{p}_N\}$ be a $N \times M$ projection matrix where

$$(\mathbf{p}_i)_j = \begin{cases} 1, \text{if } (\mathbf{x}_i)_j \text{ exists}, \\ 0, \text{otherwise}. \end{cases} \tag{1}$$

The clustering error function that is, after an initialization, locally minimized by the K-spatialmedians algorithm reads as [10, 11]

$$\mathcal{J}(\{\mathbf{m}_k\}_{k=1}^K) = \sum_{i=1}^N \min_{k=1,\dots,K} \| \operatorname{Diag}(\mathbf{p}_i)\,(\mathbf{x}_i - \mathbf{m}_k)\|_2, \tag{2}$$

where $\operatorname{Diag}(\mathbf{p}_i)$ creates a diagonal matrix using a vector $\mathbf{p}_i$. The result of the minimization is the set of prototypes $\{\mathbf{m}_k\}_{k=1}^K$, with the cluster memberships $\mathbf{C}_k = \{i : \| \operatorname{Diag}(\mathbf{p}_i)\,(\mathbf{x}_i - \mathbf{m}_k)\|_2 \leq \| \operatorname{Diag}(\mathbf{p}_i)\,(\mathbf{x}_i - \mathbf{m}_{k'})\|_2 \text{ for } 1 \leq k \neq k' \leq K\}$. Multiplication with $\mathbf{p}_i$ in (2) realizes the projection of the distance computation to only the available values of individual observations. As the definition (2) suggests, the iterative relocation of cluster prototypes simply means that one needs to solve the minimization problem iterative in each cluster. For this purpose, successive over-relaxation (SOR) of the well-known Weiszfeld algorithm for a candidate solution can be used [8, 9].

Let us assume that the data is partitioned into Q disjoint subsets: $\mathbf{X} = \{\mathbf{X}_1, ..., \mathbf{X}_Q\}$ such as $\mathbf{X} = \cup_{i=1}^Q \mathbf{X}_i$. Then the cluster memberships are spread to data partitions such as $\mathbf{C}_k = \{\mathbf{C}_{k1}, ..., \mathbf{C}_{kQ}\}$. Moreover, we denote $\mathbf{C}_{kq} = \mathbf{C}_k \cap \{i : \mathbf{x}_i \in \mathbf{X}_q\}$, where $q = 1, ..., Q$. Hence, in the SOR algorithm from the current step t into $t+1$, the candidate prototype $\mathbf{v}_k$ (see [9], p. 138) for the kth cluster can be solved with

$$\begin{aligned} \mathbf{v}_k &= (\sum_{i \in \mathbf{C}_k} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i))^{-1} \sum_{i \in \mathbf{C}_k} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)\mathbf{x}_i \\ &= (\sum_{q=1}^Q \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i))^{-1} \sum_{q=1}^Q \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)\mathbf{x}_i, \end{aligned}$$

where $\alpha_i^t = 1/\sqrt{\| \operatorname{Diag}(\mathbf{p}_i)(\mathbf{u}_k^t - \mathbf{x}_i)\|_2^2 + \epsilon}$, where ϵ is a small positive constant. If we define $\mathbf{A}_{qk}^t = \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)$ and $\mathbf{b}_{qk}^t = \sum_{i \in \mathbf{C}_{kq}} \alpha_i^t \operatorname{Diag}(\mathbf{p}_i)\mathbf{x}_i$, we get

$$\mathbf{v}_k = (\sum_{q=1}^Q \mathbf{A}_{qk}^t)^{-1} \sum_{q=1}^Q \mathbf{b}_{qk}^t. \tag{3}$$

Finally, the prototype $\mathbf{u}_k$ is updated as follows

$$\mathbf{u}_k^{t+1} = \mathbf{u}_k^t + \omega(\mathbf{v}_k - \mathbf{u}_k^t), \tag{4}$$

Algorithm 1: K-spatialmedians||

```
Input: Data partitions $\mathbf{X} = \{\mathbf{X}_1, ..., \mathbf{X}_Q\}$, projection matrix partitions $\mathbf{P} = \{\mathbf{P}_1, ..., \mathbf{P}_Q\}$,
    the number of clusters $K$, the maximum number of SOR iterations $t_{max}$, the
    threshold for convergence of SOR $\epsilon_{tol}$.
Output: Final prototypes $\{\mathbf{m}_k\}_{k=1}^K$.
 1: Initialize $\{\mathbf{m}_k\}_{k=1}^K$ with parallel K-spatialmedians$||^0$ for the complete rows in $\mathbf{X}$.
    (master and slaves)
 2: Broadcast $\{\mathbf{m}_k\}_{k=1}^K$ to all $Q$ slave processes. (master)
 3: Assign local cluster memberships $\mathbf{C}_{kq}$ for $k = 1, ..., K$. (slaves)
 4: Set $t = 0$ and $\mathbf{u}_k^t = \mathbf{m}_k$ for $k = 1, ..., K$. (master)
 5: Compute $\mathbf{A}_{qk}^t$ and $\mathbf{b}_{qk}^t$ for $k = 1, ..., K$. (slaves)
 6: Compute the global sums $\sum_{q=1}^Q \mathbf{A}_{qk}^t$ and $\sum_{q=1}^Q \mathbf{b}_{qk}^t$ by parallel reduction for the
    master process for $k = 1, ..., K$. (slaves)
 7: Compute $\mathbf{v}_k$ with Eq. 3 for $k = 1, ..., K$. (master)
 8: Compute $\mathbf{u}_k^{t+1}$ with Eq. 4 for $k = 1, ..., K$. (master)
 9: Set $t = t + 1$ and if $t < t_{max}$ and $\underset{k=1,...,K}{\text{median}} \|\mathbf{u}_k^t - \mathbf{u}_k^{t-1}\|_\infty > \epsilon_{tol}$, then repeat steps
    5-8. (master)
10: Set $\mathbf{m}_k = \mathbf{u}_k^t$ for $k = 1, ..., K$. (master)
11: Repeat steps 2-10 until convergence.
```

where $\omega \in [0, 2]$ determines the stepsize along the direction of $(\mathbf{v}_k - \mathbf{u}_k^t)$. For the consecutive SOR iterations t and $t+1$, the stopping criterion for the kth cluster is defined as $\|\mathbf{u}_k^{t+1} - \mathbf{u}_k^t\|_\infty \leq \epsilon_{tol}$.

The proposed parallel method K-spatialmedians|| is described in Algorithm 1. The distribution is based on single program multiple data (SPMD) model. The approach assumes that $\mathbf{X}$ and $\mathbf{P}$ are approximately equally distributed to Q processing elements. The proposed method first applies modified K-means|| for the initialization (referred as K-spatialmedians$||^0$). The first modification to K-means|| is that we use the Euclidean distance instead of the squared Euclidean distance during the whole initialization procedure and we apply K-spatialmedians instead of K-means to cluster the sampled points with weights. The second modification deals with the MV handling, where, because we need to have complete prototypes after the initialization, K-spatialmedians$||^0$ is run only for the complete observations in $\mathbf{X}$. In the steps 4-9, the spatial medians are computed in parallel based on the SOR algorithm. The serial version of the SOR algorithm is depicted in [9]. Note that the parallellized SOR algorithm differs from the serial one in the stopping criterion. In the parallel version, the number of SOR iterations required for convergence is the same for each cluster, since the stopping criterion is based on the median of $\{\|\mathbf{u}_k^{t+1} - \mathbf{u}_k^t\|_\infty\}_{k=1}^K$.

3 Experiments and results

The accuracy of K-spatialmedians|| was compared with K-means|| for a synthetic dataset. The scalability properties of the parallel K-spatialmedians|| implementation were experimented with a large real dataset.

3.1 Experimental setup

All experiments were performed in the MATLAB R2017b environment. The scalability experiments were performed using a cluster equipped with eight Intel Xeon CPU E7-8837 with each having 128 GB memory and 8 cores. We implemented the parallel K-spatialmedians‖ with SPMD paradigm by utilizing MATLAB's parallel computing toolbox (PCT).

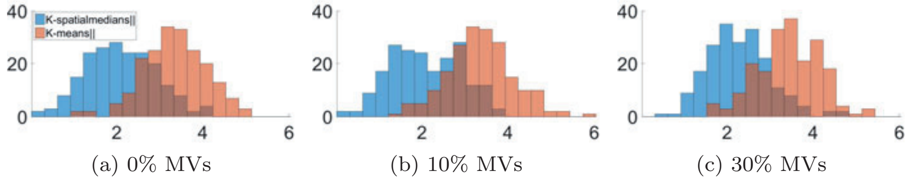
We realized the accuracy experiments with a synthetic S2[1] dataset. S2 is a two-dimensional dataset with 5000 observations. In order to assess robustness of K-spatialmedians‖, we disturbed original S2 with outliers and missing values. First, we replaced 250 observations with uniformly random observations, where both values were generated from two times larger range than the original S2. Then, we generated the MVs by randomly selecting elements from data and replacing them with MVs. Moreover, we ensured that we did not replace an observation's both elements with MVs.

For the scalability experiments we selected the Oxford buildings (OXB) dataset[2]. The experiments were run with additional dataset, which consists of 16,334,970 SIFT descriptors extracted from the original dataset with dimensionality 128. Moreover, this dataset was modified by replacing 10 percent of randomly chosen elements with MVs attached to $N/2$ randomly selected observations. The scalability related to the speedup was examined with a random 20 percent sample of OXB dataset with MVs. The scalability of with respect to the data size was tested with varying a random sample size from 20 to 100 percents.

All datasets were min-max scaled to the range $[-1, 1]$. For K-means‖, we run the initialization for the full rows of $\mathbf{X}$ and in the K-means search phase we used the available data strategy. For K-means‖ and K-spatialmedians‖^0, we set $l = 2 * K$ and $r = 5$, based on the experiments in [6]. For K-spatialmedians‖, we set $\epsilon_{tol} = 10^{-3}$, $\omega = 1.5$, and $t_{max} = 100$. For S2 we set the number of clusters to $K = 15$. For S2, the clustering iterations were ran until there were no new cluster assignments with respect to the previous iteration, and we repeated these runs 200 times. Since K-means‖ and K-spatialmedians‖ aim to minimize different cost functions, to get fair comparison, all the reported clustering errors were computed with respect to the ground truth prototypes. For each prototype, we computed Euclidean distance to the closest ground truth prototype and summed these distances. Furthermore, each of the ground truth prototypes and the prototypes archieved from the experiments contributes to the clustering error only once. In the experiments related to the speedup and the data size we set $K = 10$. To analyze the scalability with respect to the number of clusters, we varied K between 10 and 160, and this was conducted with 32 processing elements (MATLAB workers). We varied the number of processing elements between 1 and 32 to test the speedup. In the scalability experiments, clustering was performed once with 20 iterations for each setting.

[1] `http://cs.uef.fi/sipu/datasets/`

[2] `http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/`



(a) 0% MVs (b) 10% MVs (c) 30% MVs

Fig. 1: Error distributions for S2 with varying level of MVs.

3.2 Clustering quality

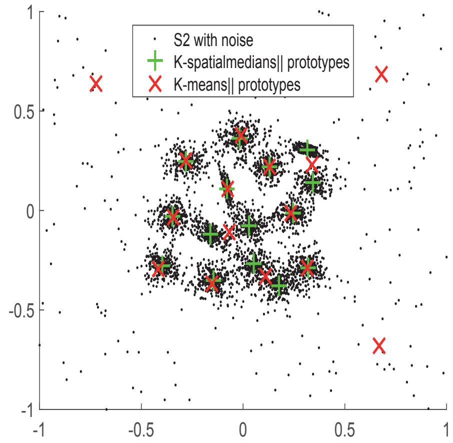
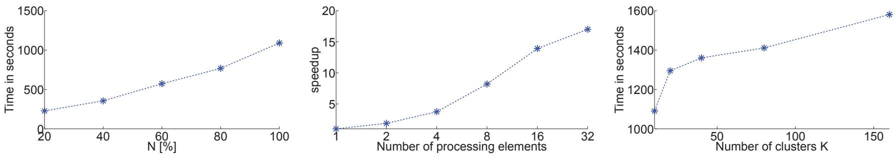


Fig. 2: The best final prototypes out of 200 runs for S2 with 10% MVs.

K-spatialmedians|| and K-means|| error distributions for S2 with 0%, 10% and 30% MVs are shown in Figure 1. Clearly, K-spatialmedians|| finds better clustering results than K-means||. Based on visual inspection of the best resulting prototypes (selection based on Eq. 2), K-spatialmedians|| is able to find an optimal clustering result for 0% and 10% MVs. For 30% MVs, K-spatialmedians|| misplaces one prototype incorrectly. Similarly, based on visual inspection of the best resulting prorotypes (selection based on SSE with the available data strategy), K-means|| misplaces six prototypes for 0%, 10% and 30% MVs. These best resulting prototypes for S2 with 10% MVs are shown in Figure 2, where they are plotted in a frame of the original S2 data points with noise.

3.3 Scalability

The scalability results for K-spatialmedians|| are shown in Figure 3. The execution time increases linearly with respect to the data size, similarly as for the original K-spatialmedians. Moreover, we observed that time taken by the initialization is negligible with respect to total running time (about 1% of the total running time). As a function of the number of processing elements, the parallel implementation scales well. Speedup is nearly linear from 1 to 16 processing elements. As a function of the number of clusters, the execution time increases linearly after $K = 20$. The nonlinear behaviour in the beging of the curve is due to a moderate increase of SOR iterations. The total number of SOR iterations for $K = 10$ is 106, for $K = 20$ 130, and for $K = 40$ 127. Finally, we also assessed Gustafson's law with 32 processing elements, and we observed 60% of the theoretical speedup.



(a) Data size (b) Speedup (c) Number of clusters

Fig. 3: Scalability of K-spatialmedians|| for OXB dataset with 10% MVs.

4 Conclusions

In this paper, we proposed K-spatialmedians||, which is a parallel version of K-spatialmedians for large and sparse data. Moreover, K-spatialmedians|| utilizes an initialization strategy based on K-means||. Based on the experiments on the synthetic dataset with noise and missing values, K-spatialmedians|| outperforms K-means|| in terms of clustering quality. Based on the experiments, the proposed algorithm scales well with respect to the size of data, the speedup and the number of clusters. In the future work, we plan to study the proposal in more detail in terms of the initialization.

References

[1] Charu C Aggarwal and Chandan K Reddy. *Data clustering: algorithms and applications.* CRC press, 2013.

[2] Vahan Petrosyan and Alexandre Proutiere. Viral initialization for spectral clustering. In *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2017*, pages 293–298, 2017.

[3] I. S. Dhillon and D. S. Modha. A data-clustering algorithm on distributed memory multiprocessors. *LargeScale Parallel Data Mining*, 1759(802):245–260, 1999.

[4] Weizhong Zhao, Huifang Ma, and Qing He. Parallel k-means clustering based on mapreduce. In *Proceedings of the 1st International Conference on Cloud Computing*, CloudCom '09, pages 674–679, 2009.

[5] Reza Farivar, Daniel Rebolledo, Ellick Chan, and Roy H Campbell. A parallel implementation of k-means clustering on GPUs. In *Pdpta*, volume 13, pages 212–312, 2008.

[6] Bahman Bahmani, Benjamin Moseley, Andrea Vattani, Ravi Kumar, and Sergei Vassilvitskii. Scalable k-means++. *Proceedings of the VLDB Endowment*, 5(7):622–633, 2012.

[7] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[8] T Kärkkäinen and S Äyrämö. On computation of spatial median for robust data mining. *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems, EUROGEN, Munich*, 2005.

[9] Sami Äyrämö. *Knowledge mining using robust clustering.* University of Jyväskylä, 2006.

[10] Sami Äyrämö and Tommi Kärkkäinen. Introduction to partitioning-based clustering methods with a robust example. *Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence 1/2006*, 2006.

[11] Joonas Hämäläinen, Susanne Jauhiainen, and Tommi Kärkkäinen. Comparison of internal clustering validation indices for prototype-based clustering. *Algorithms*, 10(3):105, 2017.

PIV

COMPARISON OF INTERNAL CLUSTERING VALIDATION INDICES FOR PROTOTYPE-BASED CLUSTERING

by

Joonas Hämäläinen, Susanne Jauhiainen and Tommi Kärkkäinen 2017

Algorithms, 10(3):105

Reproduced with kind permission of MDPI.

algorithms

Article

Comparison of Internal Clustering Validation Indices for Prototype-Based Clustering

Joonas Hämäläinen *,†, Susanne Jauhiainen † and Tommi Kärkkäinen †

Faculty of Information Technology, University of Jyvaskyla, P.O. Box 35, FI-40014 Jyvaskyla, Finland; susanne.m.jauhiainen@student.jyu.fi (S.J.); tommi.karkkainen@jyu.fi (T.K.)

* Correspondence: joonas.k.hamalainen@jyu.fi

† These authors contributed equally to this work.

Received: 13 July 2017; Accepted: 1 September 2017; Published: 6 September 2017

Abstract: Clustering is an unsupervised machine learning and pattern recognition method. In general, in addition to revealing hidden groups of similar observations and clusters, their number needs to be determined. Internal clustering validation indices estimate this number without any external information. The purpose of this article is to evaluate, empirically, characteristics of a representative set of internal clustering validation indices with many datasets. The prototype-based clustering framework includes multiple, classical and robust, statistical estimates of cluster location so that the overall setting of the paper is novel. General observations on the quality of validation indices and on the behavior of different variants of clustering algorithms will be given.

Keywords: prototype-based clustering; clustering validation index; robust statistics

1. Introduction

Clustering aims to partition a given dataset (a set of observations) into groups (clusters) that are separated from other groups in a twofold manner: observations within a cluster are similar to each other and dissimilar to observations in other clusters [1]. Diverse sets of clustering approaches have been developed over the years, e.g., density-based, probabilistic, grid-based, and spectral clustering [2]. However, the two most common groups of crisp (here, we do not consider fuzzy clustering [3]) clustering algorithms are partitional and hierarchical clustering [4]. Hierarchical clustering constructs a tree structure from data to present layers of clustering results, but because of the pairwise distance matrix requirement, the basic form of the method is not scalable to a large volume of data [5]. Moreover, many clustering algorithms, including hierarchical clustering, can produce clusters of arbitrary shapes in the data space, which might be difficult to interpret for knowledge discovery [6].

The two aims of clustering for K groups in data are approached in the partitional algorithms, most prominently in the classical K-means [4,7], by using two main phases: initial generation of K prototypes and local refinement of the initial prototypes. The initial prototypes should be separated from each other [4,8]. Lately, the K-means++ algorithm [9], where the random initialization is based on a density function favoring distinct prototypes, has become the most popular variant to initialize the K-means-type of an algorithm. Because the prototype refinement acts locally, we need a globalization strategy to explore the search space. This can be accomplished with repeated restarts through initial prototype regeneration [10] or by using evolutionary approaches with a population of different candidate solutions [11].

One can utilize different error (score) functions in partitional clustering algorithms [12]. Mean is the statistical estimate of the cluster prototype in K-means and the clustering error is measured with the least-squares residual. This implies the assumption of spherically symmetric, normally distributed data with Gaussian noise. These conditions are relaxed when the cluster prototype is replaced, e.g., with a robust location estimate [13–15]. The two simplest robust estimates of location are median and spatial median, whose underlying spherically symmetric distributions are uniform and Laplace distributions, respectively. If the type of data is discrete, for instance, an integer variable with uniform quantization error [16], then the Gaussian assumption is not valid. Median, given by the middle value of the ordered univariate sample (unique only for odd numbers of points [17]), can, like the mean, be estimated from the marginal distribution being inherently univariate. The spatial median, on the other hand, is truly a multivariate, orthogonally equivariant location estimate [18]. These location estimates and their intrinsic properties are illustrated and more thoroughly discussed in [17,19]. The median and spatial median have many attractive statistical properties, especially since their so-called breakdown point is 0.5, i.e., they can handle up to 50% of contaminated and erroneous data.

In a typical unsupervised scenario, one does not possess any prior knowledge of the number of clusters K. Finding the best possible representation of data with K groups is difficult because the number of all possible groupings is the sum of Stirling numbers of the second kind [19]. Defining validation measures for clustering results has been, therefore, a challenging problem that different approaches have tried to overcome [20–25]. The quality of a clustering result can be measured with a Clustering Validation Index (CVI). The aim of a CVI is to estimate the most appropriate K based on the compactness and separation of the clusters. Validation indices can be divided into three categories [26]: internal, external, and relative. An external validation index uses prior knowledge, an internal index is based on information from the data only, and in a relative CVI, multiple clustering results are compared. A comprehensive review of clustering validation techniques up to 2001 was provided in [27]. There exists also alternative approaches for determining the number of clusters, e.g., by measuring the stability of the clustering method [28] or using multiobjective evolutionary approaches [11,29].

In this paper, we continue the previous work reported in [30] by focusing on a comparison of the seven best internal CVIs, as identified in [30] and augmented by [22]. The earlier comparisons, typically reported when suggesting novel CVIs, only include K-means as the partitional clustering algorithm [22,30–36]. Here, this treatment is generalized by using multiple statistical estimates as a cluster prototype and to define the clustering error, under the currently most common initialization strategy as proposed in [9] (which is also generalized). Note that prototype-based clustering can also be conducted with an incremental fashion [37–39]. However, here were restrict ourselves on the batch versions of the algorithms, which can be guaranteed to converge in a finite number of iterations (see Section 2.1). The definitions of the considered validation indices are also extended and empirically compared with K-means, K-medians, and K-spatialmedians (using spatial median as a prototype estimate) clustering results for a large pool of benchmark datasets. According to our knowledge, there exists no previous work that compares CVIs with multiple different distance metrics. Our aim is to sample the main characteristics of the indices considered and to identify what indices most reliably refer to ground truth values of the benchmark datasets. Note that by their construction, all CVIs considered here can also be used to suggest the number of clusters in hierarchical clustering.

The structure of the article is as follows. After this introductory section, we describe generalized prototype-based clustering, discuss its convergence, and also present the generalized versions of cluster initialization and indices in Section 2. Our experimental setup is described in Section 3, and the results are given and discussed in Section 4. Finally, conclusions are drawn in Section 5.

2. Methods

In this section, we introduce and analyze all the necessary formulations for clustering and cluster validation indices.

2.1. General Prototype-Based Clustering and Its Convergence

As described above, prototype-based partitional clustering algorithms comprise two main phases. First, they start with an initial partition of the data, and second, the quality of this partition is improved by a local search algorithm during the search phase. The initial partition can be obtained based on many different principles [4,8], but a common strategy is to use distinct prototypes [9]. Most typically, the globalization of the whole algorithm is based on random initialization with several regenerations [10]. Then, the best solution with the smallest clustering error is chosen as the final result. The iterative relocation algorithm skeleton for prototype-based partitional clustering is presented in Algorithm 1 [12,16].

```
Algorithm 1: Prototype-based partitional clustering algorithm.
    Input: Dataset and the number of clusters $K$.
    Output: Partition of dataset into $K$ disjoint groups.
    Select $K$ points as the initial prototypes;
    repeat
        1. Assign individual observation to the closest prototype;
        2. Recompute the prototype with the assigned observations;
    until the partition does not change;
```

As stated in [17] (see also [19]), the different location estimates for a cluster prototype arise from different l_p-norms to the q-th power as the distance measure and the corresponding clustering error function; mean refers to $\|\cdot\|_2^2$ ($p = q = 2$), median is characterized by $\|\cdot\|_1^1$ ($p = q = 1$), and the spatial median is given by $\|\cdot\|_2^1$ ($p = 2, q = 1$). Hence, generally the repetition of Steps 1 and 2 from the search phase of Algorithm 1 locally minimize the following clustering error criterion:

$$\mathcal{J}(\{\mathbf{b}_k\}) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathbf{C}_k} \|\mathbf{x}_i - \mathbf{b}_k\|_p^q. \tag{1}$$

Here $\{\mathbf{b}_k\}$, $k = 1, \ldots, K$ denote the prototype vectors to be determined and $\{\mathbf{x}_i\}_{i=1}^N$, $\mathbf{x}_i \in \mathbb{R}^n$ refers to the given set of n-dimensional observations. The interpretation of (1) is that each observation $\mathbf{x}_i$ is assigned to cluster k with the closest prototype on the l_p-norm:

$$\mathbf{C}_k = \{1 \le k \le K \mid \|\mathbf{x}_i - \mathbf{b}_k\|_p \le \|\mathbf{x}_i - \mathbf{b}_{k'}\|_p \quad \forall k \neq k'\}.$$

Hence, as noted in [40], another more compact way of formalizing the clustering error criterion reads as

$$\mathcal{J}(\{\mathbf{b}_k\}) = \sum_{i=1}^{N} \min_{k=1,\ldots,K} \|\mathbf{x}_i - \mathbf{b}_k\|_p^q, \tag{2}$$

which more clearly shows the *nonsmoothness* of the clustering problem, because the min-operator is not classically differentiable (see [17] and references therein). This observation gives rise to a different set of clustering algorithms that are based on nonsmooth optimization solvers [41].

However, despite the nonsmoothness of the error function, it can be shown that the search phase of Algorithm 1 decreases the clustering error, ensuring local convergence of the algorithm in finite many steps. We formalize this in the next proposition. The proof here is a slight modification and simplification of the more general treatment in [19], Theorem 5.3.1, along the lines of the convergence analyses in different problem domains, as given in [42–44].

Proposition 1. *The repeated Steps 1 and 2 of Algorithm 1 decrease the clustering error function (2). This guarantees convergence of the algorithm in finite many steps.*

Proof. Let us denote by superscript t the current iterates of the prototypes $\{\mathbf{b}_k^t\}$ with the initial candidates for $t = 0$. If assignments to clusters and to the closest cluster prototypes do not change, we are done, so let us assume that the repeated step 1 in Algorithm 1 has identified at least one $1 \leq j \leq N$ such that, for $\mathbf{x}_j \in \mathbf{C}_k^t$, there exists a better prototype candidate:

$$\|\mathbf{x}_j - \mathbf{b}_k^t\|_p > \|\mathbf{x}_j - \mathbf{b}_{k'}^t\|_p \quad \text{for some } \; k' \neq k. \tag{3}$$

Then, a direct computation, using monotonicity of the function $\| \cdot \|^q$ for $q = \{1, 2\}$ and reflecting the change in the assignments, gives

$$\begin{aligned}
\mathcal{J}(\{\mathbf{b}_k^t\}) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathbf{C}_k^t} \|\mathbf{x}_i - \mathbf{b}_k^t\|_p^q &= \sum_{k=1}^{K} \left(\sum_{\substack{\mathbf{x}_i \in \mathbf{C}_k^t \\ i \neq j}} \|\mathbf{x}_i - \mathbf{b}_k^t\|_p^q + \|\mathbf{x}_j - \mathbf{b}_k^t\|_p^q \right) \\
&> \sum_{k=1}^{K} \left(\sum_{\substack{\mathbf{x}_i \in \mathbf{C}_k^t \\ i \neq j}} \|\mathbf{x}_i - \mathbf{b}_k^t\|_p^q + \|\mathbf{x}_j - \mathbf{b}_{k'}^t\|_p^q \right) = \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathbf{C}_k^{t+1}} \|\mathbf{x}_i - \mathbf{b}_k^t\|_p^q \quad (4) \\
&\geq \sum_{k=1}^{K} \sum_{\mathbf{x}_i \in \mathbf{C}_k^{t+1}} \|\mathbf{x}_i - \mathbf{b}_k^{t+1}\|_p^q = \mathcal{J}(\{\mathbf{b}_k^{t+1}\}).
\end{aligned}$$

Here, the last inequality follows from the repeated Step 2 of Algorithm 1 and from the optimization-based definitions of mean/median/spatial median as minimizers of the l_p^q-norm [17] over a dataset:

$$\sum_{\mathbf{x}_i \in \mathbf{C}_k^{t+1}} \|\mathbf{x}_i - \mathbf{b}_k^{t+1}\|_p^q = \min_{\mathbf{b} \in \mathbb{R}^n} \sum_{\mathbf{x}_i \in \mathbf{C}_k^{t+1}} \|\mathbf{x}_i - \mathbf{b}\|_p^q \leq \sum_{\mathbf{x}_i \in \mathbf{C}_k^{t+1}} \|\mathbf{x}_i - \mathbf{b}_k^t\|_p^q \quad \text{for all } \; k. \tag{5}$$

Because (4) and (5) are valid for any reallocated index j satisfying (3), we conclude that the clustering error strictly decreases when a reallocation for a set of observations occurs in Algorithm 1.

To this end, because there exists only a finite number of possible sets $\mathbf{C}_k^t$, we must have $\mathbf{C}_k^{t+1} = \mathbf{C}_k^t$ after a finite number of steps $t \mapsto t+1$. This ends the proof. □

The K-means++ initialization method utilizes squared Euclidean distance-based probabilities to sample initial prototypes from the data points. In Algorithm 2, this initialization strategy is generalized for varying l_p-norms to the q-th power. Note that Algorithm 2 is the same as the K-means++ initialization algorithm when $p = q = 2$. In order to be successful, one needs to assume in Algorithm 2 that the dataset $\{\mathbf{x}_i\}_{i=1}^N$ has at least K distinct data points, which is a natural and reasonable assumption. In Step 2, the probability for each point $\mathbf{x}_i$ to be selected as the next initial prototype is proportional to the distance to the closest already selected prototype divided by the value of the clustering error Function (2) for the already selected prototypes. Clearly, in each iteration, the most distant points (with respect to the previously selected initial prototypes) have the highest probability of being selected.

Algorithm 2: General K-means++-type initialization.

```
Input: Dataset $\{\mathbf{x}_i\}_{i=1}^N$ and the number of clusters $K$.
Output: Initial prototypes $\{\mathbf{b}_k\}_{k=1}^K$.
1. Select $\mathbf{b}_1 = \mathbf{x}_i$ uniformly randomly, $i = 1, \dots, N$;
for $k = 2, k = k+1, k \leq K$ do
    2. Select $\mathbf{b}_k = \mathbf{x}_i$ with probability $\frac{\min_{j=1,\dots,k-1} \|\mathbf{x}_i - \mathbf{b}_j\|_p^q}{\mathcal{J}(\{\mathbf{b}_j\}_{j=1}^{k-1})}$, $i = 1, \dots, N$;
end
```

2.2. Cluster Validation Indices

From now on, for a given number of clusters K, we denote, by $\{\mathbf{c}_k\}$ and $\{\mathbf{C}_k\}$, $k = 1, \ldots, K$, the best prototypes and divisions obtained after a fixed number of repeated applications of Algorithm 1. When $K = 1$, we denote the prototype, i.e., the mean, median, or spatial median, of the whole data with m. Moreover, we let $\mathcal{J}_K$ denote the corresponding clustering error over the whole data and $\mathcal{J}_K^k = \sum_{\mathbf{x}_i \in \mathbf{C}_k} \|\mathbf{x}_i - \mathbf{c}_k\|_p^q$ to refer to the corresponding final within-cluster errors.

Cluster validation considers the quality of the result of a clustering algorithm, attempting to find the partition that best fits the nature of the data. The number of clusters, given as a parameter for many clustering algorithms (such as the ones presented in Section 2.1), should be decided based on the natural structure of the data. Like the best clustering solution, the number of clusters is also not always clear and many 'right' answers can exist (see, e.g., [19], Figure 5). The number can also depend on the resolution, i.e., whether the within- and between- cluster separabilities are considered globally or locally. Here, we focus on the CVIs based on the internal criteria.

The validation indices measure how well the general goal of clustering—high similarity within clusters and high separability between clusters—is achieved. These are considered with measures of within-cluster (*Intra*) and between-cluster (*Inter*) separability, for which lower and higher values are better, respectively. Normally, a division between *Intra* and *Inter* is made and the optimal value is at the minimum or maximum, based on the order of the division.

In Table 1, the best internal validation indices (as determined for the K-means-type of clustering in [30], augmented with [22]) are introduced in a general fashion for the l_p^q-norm setting. All except one of the indices have been modified in such a way that the optimal number of clusters can be found at the minimal value. Only the Wemmert–Gançarski index, which has a unique pattern of the general formula, is given in the original form, where the maximum value indicates the optimal number of clusters. In Table 1, if the formula that combines Intra and Inter depends on the generated clusters, then this is indicated with the corresponding parameters.

Table 1. Internal cluster validation indices.

Name	Notation	Intra	Inter	Formula
KCE [30]	KCE	$K \times \mathcal{J}_K$		*Intra*
WB-index [22]	WB	$K \times \mathcal{J}_K$	$\sum_{k=1}^{K} n_k \|\mathbf{c}_k - m\|_p^q$	$\frac{Intra}{Inter}$
Calinski–Harabasz [24]	CH	$(K-1) \times \mathcal{J}_K$	$(N-K) \times \sum_{k=1}^{K} n_k \|\mathbf{c}_k - m\|_p^q$	$\frac{Intra}{Inter}$
Davies–Bouldin [23]	DB	$\frac{1}{n_k}\mathcal{J}_K^k + \frac{1}{n_{k'}}\mathcal{J}_K^{k'}$	$\|\mathbf{c}_k - \mathbf{c}_{k'}\|_p^q$	$\frac{1}{K}\sum_{k=1}^{K} \max_{k \neq k'} \frac{Intra(k,k')}{Inter(k,k')}$
Pakhira, Bandyopadhyay, and Maulik [45]	PBM	$K \times \mathcal{J}_K$	$\max_{k \neq k'}(\|\mathbf{c}_k - \mathbf{c}_{k'}\|_p^q) \times \mathcal{J}_1$	$\left(\frac{Intra}{Inter}\right)^2$
Ray–Turi [25]	RT	$\frac{1}{N} \times \mathcal{J}_K$	$\min_{k \neq k'} \|\mathbf{c}_k - \mathbf{c}_{k'}\|_p^q$	$\frac{Intra}{Inter}$
Wemmert– Gançarski ([46])	WG	$\|\mathbf{x}_i - \mathbf{c}_k\|_p^q$	$\min_{k \neq k'} \|\mathbf{x}_i - \mathbf{c}_{k'}\|_p^q$	$\frac{1}{N}\sum_{k=1}^{K} max\left(0,\, n_k - \sum_{i \in I_k} \frac{Intra(i)}{Inter(i)}\right)$

As can be seen from the formulas of the indices, there are a lot of similarities in how different indices measure the within- and between-cluster separability. For example, the clustering error straightforwardly measures the similarity within clusters and, therefore, almost all of the indices include it in their measure of *Intra*. In case of between-cluster separability, it is common to measure, for instance, the distance between cluster prototypes or between cluster prototypes and the whole data prototype. The rationale behind the index structures and their more detailed descriptions can be found in the original articles.

2.3. On Computational Complexity

The computational complexity of the prototype-based clustering is $\mathcal{O}(RTKNn)$, where T is the number of iterations needed for convergence and R is the chosen number of repetitions of joint Algorithms 1 and 2. As the clustering itself is quite time-consuming, especially with large datasets, it is only natural to also consider the complexity of the indices to avoid excessively complex computations. Here, the KCE index has an advantage in not requiring any extra calculation after the clustering solution has been obtained. For the indices that go through the prototypes once, here the WB and CH that measure the prototype distances to the whole data prototype, the complexity is $\mathcal{O}(Kn)$. Indices that measure the distances between all the prototypes, such as DB, PBM, and RT, have complexity $\mathcal{O}(K^2n)$.

In our tests, WG is the index with the highest complexity, $\mathcal{O}(KNn)$, going through the whole data, comparing points and the prototypes. A commonly used and generally well performing index, Silhouette (see, e.g., [30,33]), goes through the whole data twice when calculating its values and therefore its complexity is $\mathcal{O}(N^2n)$. With large datasets of at least hundreds of thousands of observations, this might be even more complex than the clustering task itself, with the chosen values of R and K and the observed value of T (see Figure A1) in Section 4. If computationally more involved indices would be used, also application of more complex clustering algorithms should be considered. Therefore, Silhouette was excluded from our tests.

2.4. About Earlier Validation Index Comparisons

There has been a lot of research on cluster validation, including comparisons of different validation indices and clustering algorithms. Often when a new index is proposed, the work also includes a set of comparisons that conclude that the new index is the best one. In [34], eight common CVIs were compared and with 5% additional noise, different densities, and skewed distributions, most indices were able to find the correct number of clusters. However, only three of them were able to recognize close subclusters. In their tests, S_Dbw was the only CVI that suggested the correct number of clusters for all datasets. In our previous tests in [30], the S_Dbw also recognized the close subclusters in Sim5D2, but it did not perform that well in general.

Often no single CVI has a clear advantage in every context, but each is best suited to a certain kind of data. This was also the conclusion in [33], where 30 different indices with 720 synthetic and 20 real datasets were compared. However, a group of about 10 indices were found to be the most recommendable, including Silhouette, Davies–Bouldin * and Calinski–Harabasz at the top. Also, in the earlier extensive comparison [47], where 30 indices were compared, the authors suggested that if different datasets were used for testing, the order of the indices would change but the best ones—including Calinski–Harabasz, Duda–Hart, and the C-index—would still perform well.

3. Experimental Setup

Next, we test the indices in Table 1 with different distance measures, i.e., with different prototype-based clustering algorithms. The index values are calculated with the distance corresponding to the clustering method used, i.e., city-block with the K-medians, squared Euclidean with the K-means, and Euclidean with the K-spatialmedians. All datasets were scaled to the range of $[-1, 1]$. All the tests were run on MATLAB (R2014a), where a reference implementation on both the validation indices and the general clustering Algorithm 1 with the initialization given in Algorithm 2 were prepared. The impact of the necessary amount of repetitions of Algorithms 1 and 2 was tested with multiple datasets (*S*-sets, *Dim*-sets, *A1*, *Unbalance*), comparing the clustering error and the cluster assignments. With 100 repetitions, the minimum clustering error and the corresponding cluster assignments were stabilized and an appropriate clustering result was found. This result with the minimum clustering error was used for computing the CVI value.

To test the indices, we used the basic benchmark datasets described in detail in [48], with the two other synthetic datasets http://users.jyu.fi/~jookriha/CVI/Data/ as given in [30] (see Figure 1).

These benchmark sets are synthetic datasets, suggested for use when testing any algorithm dealing with clustering spherical or Gaussian data. Here, we restrict ourselves to the benchmarks with at most 20 clusters, because the interpretation and knowledge discovery from the clustering results with a large number of prototypes might become tedious [6,49]. Therefore, the number of clusters was also tested with $K = 2 - 25$.

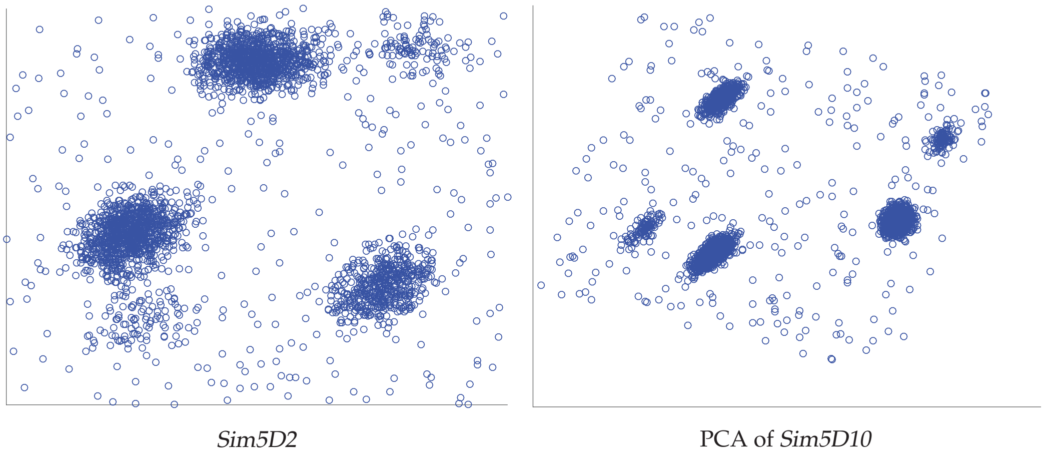

Sim5D2 PCA of *Sim5D10*

Figure 1. Scatter plots of *Sim5* datasets.

In addition, we use six real datasets that include *Steel Plates*, *Ionosphere*, and *Satimage (Train)* from the UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets.html *Iris* and *Arrhythmia* from MATLAB's sample datasets, https://se.mathworks.com/help/stats/_bq9uxn4.html, and the *USPS* dataset. https://www.otexts.org/1577 A summary of all these datasets can be seen in Table 2. For these real datasets, even if there are class labels provided, we do not compare the clustering results with the class information because of the completely unsupervised scenario with the internal validation indices. Moreover, the classes need not correspond to the clusters determined by the prototype-based clustering algorithm, since the probability density functions of the classes are not necessarily spherically symmetric. For these datasets, we therefore only study and compare the stability of the suggestions on the numbers of clusters for different indices.

Table 2. Description of datasets.

Data	Size	Dimensions	Clusters	Description
S	5000	2	15	Varying overlap
G2	2048	1–1024	2	Varying overlap and dimensionality
DIM	1024	32–1024	16	Varying dimensionality
A	3000–7500	2	20–50	Varying number of clusters
Unbalance	6500	2	8	Both dense and sparse clusters
Birch	100,000	2	1–100	Varying structure
Sim5	2970	2–10	5	Small subclusters close to bigger ones
Data	**Size**	**Dimensions**	**Classes**	**Description**
Iris	150	4	3	Three species of iris
Arrhythmia	452	279	13	Different types of cardiac arrhythmia
Steel Plates	1941	27	7	Steel plates faults
Ionosphere	351	34	2	Radar returns from the ionosphere
USPS	9298	256	10	Numeric data from scanned handwritten digits
Satimage (Train)	6435	36	6	Satellite images

Finally, our datasets include the following: *A1*, with 20 spherical clusters and some overlap; the *S*-sets, including four datasets with 15 Gaussian clusters, and cluster overlap increasing gradually from *S1* to *S4*; *Dim*-sets, including six datasets with 16 well-separated clusters in high-dimensional space and dimensions varying from 32 to 1024; a subset of *Birch2*, including 19 datasets with 2–20 clusters with their centroids on a sine curve; *Unbalance*, with eight clusters in two separate groups, the one having three dense and small clusters and the other five more sparse and bigger clusters; and a subset of *G2*, with 20 datasets—the lowest and highest overlap in ten different dimensions from 1 to 1024. Finally, together with the six real datasets, we had altogether 62 datasets in our tests.

4. Results

In this section, we provide the results of the clustering validation index tests with K-medians, K-means, and K-spatialmedians clustering. Results for the synthetic datasets are combined in Table A1, where each cell includes the results for all three methods with 'cb' referring to the city-block distance ($p = q = 1$), 'se' to the squared Euclidean distance ($p = q = 2$), and 'ec' to the Euclidean distance ($p = 2,\ q = 1$). In addition, the convergence properties of the clustering algorithms are compared for varying K values.

4.1. CVIs for Synthetic Datasets

For the *Dim*-sets, results were equal (and correct) in all dimensions from 32 to 1024 and for the *G2*-sets results were equal (and correct) from dimension eight upwards. Therefore, these have been excluded from Table A1. Correct suggestions for the number of clusters are marked in bold.

The most challenging synthetic datasets seem to be *Unbalance*, *Sim5D2*, and *Sim5D10*. Only a few indices were able to recognize the correct number and no single index managed to solve both the *Unbalance* and the *Sim* sets.

As concluded in the previous studies (see Section 2.4), different indices seem to work better with different kinds of datasets. However, there are also a lot of differences in the general performances of the tested CVIs. The overall success rates for the CVIs, i.e., for how many of the datasets (%) it gave correct suggestions, can be seen in Table 3, listed separately for each distance measure.

Table 3. Right suggestions for the 56 synthetic datasets (number of right suggestions/number of datasets).

Index	City-Block	Squared Euclidean	Euclidean
KCE	85.7%	87.5%	85.7%
WB	87.5%	80.4%	87.5%
CH	58.9%	85.7%	57.1%
DB	60.7%	60.7%	62.5%
PBM	87.5%	64.3%	89.3%
RT	60.7%	58.9%	64.3%
WG	94.6%	96.4%	92.9%

In conclusion, the WG index outperforms all the other indices in all three distance measures and clustering approaches. WB and KCE also perform very well in general. For some indices, the performances vary between different distances; for example, CH works very well with the squared Euclidean distance, while PBM clearly works better with city-block and Euclidean distances. As a whole, the recommendation for the use of indices is as follows: for the original K-means, WG, KCE, and CH are the three best indices, and for the robust variants with K-medians and K-spatialmedians, WG, PBM, and WB have the highest success rate.

4.2. CVIs for Real Datasets

As mentioned, here we only observe and compare the stability of the clustering results. The results are combined in Table 4. With real datasets, a typical behavior of the internal validation indices is the suggestion of only a small number of clusters. This is especially true for KCE and CH, even if we know that there are observations from multiple classes, with the class boundaries having unknown forms and shapes. Different from the other indices, the results of DB seem to deviate a lot, with high variability over the datasets. The same can happen for RT, with squared Euclidean distance.

Table 4. Internal CVIs results for real datasets.

cb, se, ec	KCE	WB	CH	DB	PBM	RT	WG
Iris	2, 3, 2	2, 3, 2	2, 3, 2	2, 2, 2	3, 22, 3	2, 2, 2	2, 2, 2
Arrhythmia	2, 2, 2	2, 5, 2	2, 2, 2	25, 24, 17	2, 14, 2	2, 25, 3	2, 25, 25
Steel	2, 2, 2	3, 5, 2	2, 2, 2	7, 3, 7	3, 7, 2	2, 2, 3	3, 2, 3
Ionosphere	2, 2, 2	2, 2, 2	2, 2, 2	11, 23, 2	3, 20, 3	4, 4, 4	4, 2, 2
USPS	2, 2, 2	2, 4, 2	2, 2, 2	2, 18, 11	2, 4, 4	2, 12, 7	2, 2, 7
Satimage (Train)	2, 3, 2	3, 6, 2	2, 3, 2	3, 3, 3	3, 6, 3	3, 3, 3	3, 3, 3

Based on the observed behavior, the most stable, and therefore the recommended indices, seem to be WB, PBM, and WG. However, when the data is of higher dimension without a Gaussian structure, the curse-of-dimensionality [50] might be the reason for the basic suggestions of a low number of clusters. Therefore, to obtain more fine-tuned clustering results and validation index evaluations, it might be necessary to use the prototype-based clustering in a hierarchical manner, as suggested in [16,51]. For example, many indices suggested three clusters for the *Sim5* datasets, but after a further partitioning of the data, new index values could be calculated for those three clusters separately, and the correct division into five clusters altogether could be revealed.

4.3. Convergence

For each repetition, the number of iterations needed for convergence, T, was saved. Median values of T for synthetic datasets were: 19 for K-medians, 19 for K-means, and 21 for K-spatialmedians. K-spatialmedians requires slightly more iterations than K-means and K-medians. In practice, the effect of the total running time between $T = 19$ and $T = 21$ is negligible. For real datasets, median values of T are again similar: 15 for K-medians, 17 for K-means, and 16 for K-spatialmedians. K-means performs slightly worse than the robust K-medians and K-spatialmedians for real datasets. It is known that K-means is sensitive to noise, which could explain these results. Overall, based on the median values of T, there seems to be no practical difference between the convergence of K-medians, K-means, and K-spatialmedians.

We plotted and analyzed the median of T as a function of K for each dataset separately in order to compare the convergence characteristics of K-means, K-medians, and K-spatialmedians. The most relevant plots are shown in Figure A1. Even though the median values of T are close to each other in general, there are some interesting differences with multiple datasets.

From Figure A1, we can observe that the robust location estimates, median and spatial median, clearly converge faster than the mean for the *S4* dataset, which has highly overlapping clusters. For the *USPS* dataset, K-medians seems to converge slightly faster than K-means. In the figure, plots for datasets *b2-sub-5*, *b2-sub-10*, *b2-sub-15*, and *b2-sub-20* show that K-means converges faster than K-medians and K-spatialmedians when K is smaller than the number of clusters in the dataset. This might be because the mean location estimate is more sensitive to movement towards the center of multiple combined clusters than the robust location estimates since outlying points within a cluster affect it more than the robust location estimates. The plots of datasets *G2-2-10* and *G2-64-10* demonstrate how the curse-of-dimensionality greatly affects K-medians when compared to K-means and K-spatialmedians. K-medians converge faster than K-means and K-spatialmedians in the 2-dimensional case; however, from the 64-dimensional case onward the reverse is observed.

5. Conclusions

Tests for a representative set of previously qualified internal clustering validation indices with many datasets, for the most common prototype-based clustering framework with multiple statistical estimates of cluster location, were reported here. This study further confirmed the conclusions in the previous index comparisons that no single CVI dominates in each context, and some indices are better suited to different kinds of data. Therefore, it is recommended to utilize multiple indices when doing cluster analysis. However, in our tests with the synthetic datasets, the WG index outperformed other indices in all distance measures used, also showing stable behavior with real datasets. It found the correct number of clusters for all datasets, except the *Sim5* with different sized clusters and overlap. Due to this high performance, the WG index would be worth studying more in the future. In addition, PBM was an index with a high and stable performance, along with WB for robust clustering and KCE and CH for K-means. The correct number of clusters in the *Sim5* datas was only found with WB and KCE for K-means and PBM with robust estimates. Based on these tests, DB and RT are not recommended. In general, the indices seem to perform worse with higher cluster overlap and some of them (CH, DB, RT) fail more often when the number of clusters grows.

Here, we also extended previous index comparisons using K-means clustering, with robust K-medians and K-spatialmedians clusterings. Just like the indices, the different distance measures also seem to work differently with different datasets and, in addition, some indices seem to work better with different distances. For instance, for the synthetic datasets, the PBM index clearly works better with city-block and Euclidean distances. Therefore, the usage of robust K-medians and K-spatialmedians could bring added value when trying to find the optimal number of clusters present in the data.

In addition, the convergence properties of the clustering algorithms were compared. Based on the experiments, the number of iterations needed for convergence varies for different datasets between the methods, e.g., K-means requires more iterations than the robust clustering methods for noisy datasets, while K-medians is most affected by an increase in dimensionality. Moreover, K-means++-type initialization strategy seems to work well with city-block and Euclidean distance.

Moreover, as many indices often suggest quite low numbers of clusters, due to clusters being seemingly close to each other, when observed globally in a high-dimensional space, the hierarchical application of a partitional prototype-based clustering algorithm is recommended, in order to improve recognition of possible clusters at different resolution levels.

Acknowledgments: The authors would like to thank the editor and reviewers for their insightful suggestions, which improved the paper.

Author Contributions: Tommi Kärkkäinen conceived and designed this research; Joonas Hämäläinen and Susanne Jauhiainen performed the experiments; Joonas Hämäläinen, Susanne Jauhiainen, and Tommi Kärkkäinen wrote the paper.

Conflicts of Interest: The authors declare no conflict of interest.

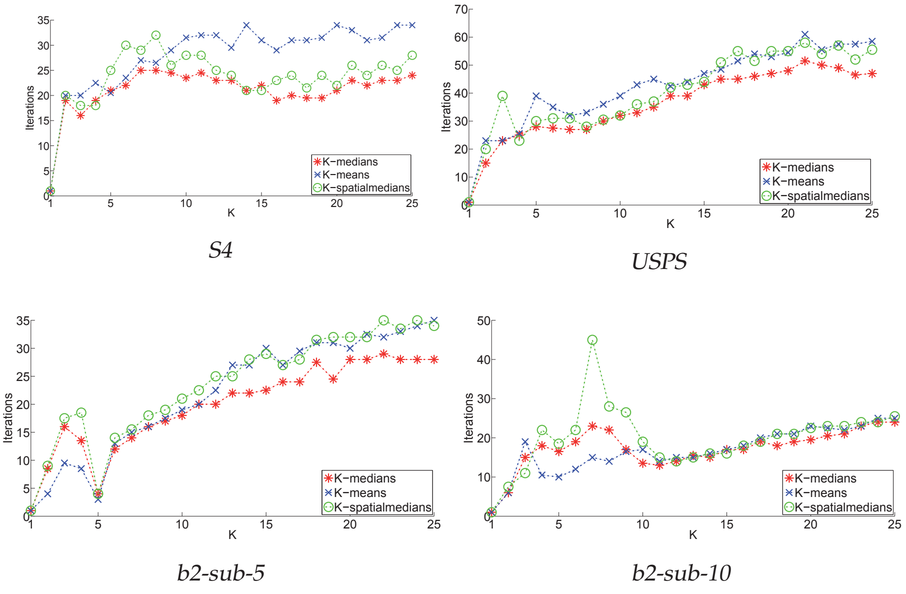
Appendix A

Table A1. Internal CVIs results for synthetic datasets.

cb, se, ec	**KCE**	**WB**	**CH**	**DB**	**PBM**	**RT**	**WG**
Unbalance	5, 20, 5	5, 20, 5	5, 20, 5	**8**, **8**, **8**	5, 25, 5	**8**, 2, **8**	**8**, **8**, **8**
a1	2, **20**, 2	2, **20**, 2	2, **20**, 2	**20**, 19, 16	6, 24, 6	2, 18, 17	**20**, **20**, **20**
Sim5D10	3, **5**, 3	3, **5**, 3	3, 3, 3	3, 3, 3	**5**, 10, **5**	3, 3, 3	3, 3, 3
Sim5D2	3, **5**, 3	3, **5**, 3	3, 3, 3	3, 3, 3	**5**, 16, **5**	3, 3, 3	3, 3, 3
S1	2, **15**, 2	**15**, **15**, **15**	2, **15**, 2	**15**, **15**, **15**	**15**, **15**, **15**	**15**, **15**, **15**	**15**, **15**, **15**
S2	2, **15**, 2	2, **15**, 3	2, **15**, 2	**15**, **15**, **15**	**15**, **15**, **15**	**15**, **15**, **15**	**15**, **15**, **15**
S3	2,**15**, 2	2, **15**, 2	2, **15**, 2	7, 13, **15**	4, **15**, 4	4, 4, **15**	**15**, **15**, **15**
S4	2, **15**, 2	2, **15**, 3	2, **15**, 2	17, 17, **15**	5, 23, 5	17, 13, **15**	16, **15**, 16
DIM032	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
DIM1024	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**	**16**, **16**, **16**

Table A1. *Cont.*

cb, se, ec	KCE	WB	CH	DB	PBM	RT	WG
b2-sub-2	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2**, 20, **2**	**2, 2, 2**	**2, 2, 2**
b2-sub-3	**3, 3, 3**	**3, 3, 3**	**3, 3, 3**	**3, 3, 3**	**3, 3, 3**	**3, 3, 3**	**3, 3, 3**
b2-sub-4	**4, 4, 4**	**4, 4, 4**	**4, 4, 4**	**4, 4, 4**	**4, 4, 4**	**4, 4, 4**	**4, 4, 4**
b2-sub-5	**5, 5, 5**	**5, 5, 5**	**5, 5**, 2	**5, 5, 5**	**5, 5, 5**	**5, 5, 5**	**5, 5, 5**
b2-sub-6	**6, 6, 6**	**6, 6, 6**	2, **6** ,2	5, **6**, 5	**6, 6, 6**	5, **6, 6**	**6, 6, 6**
b2-sub-7	**7, 7, 7**	**7, 7, 7**	2, **7**, 2	5, 6, 6	**7**, 14, **7**	2, 2, 2	**7, 7, 7**
b2-sub-8	**8, 8, 8**	**8, 8, 8**	2, **8**, 2	6, 6, 7	**8** ,17, **8**	2, 2, 2	**8, 8, 8**
b2-sub-9	**9**, 19, **9**	**9**, 19, **9**	2, **9**, 2	6, 7, 8	**9**, 21, **9**	2, 7, 7	**9, 9, 9**
b2-sub-10	**10**, 21, **10**	**10**, 21, **10**	2, 21, 2	8, 8, 9	**10**, 25, **10**	8, 8, 8	**10, 10, 10**
b2-sub-11	**11**, 23, **11**	**11**, 23, **11**	2, 23, 3	9, 9, 10	**11**, 24, **11**	9, 9, 8	**11, 11, 11**
b2-sub-12	**12**, 25, **12**	**12**, 25, **12**	2, 25, 3	10, 10, 11	**12**, 25, **12**	10, 10, 9	**12, 12, 12**
b2-sub-13	**13, 13, 13**	**13**, 24, **13**	2, **13**, 2	11, 11, 12	**13**, 24, **13**	11, 11, 11	**13, 13, 13**
b2-sub-14	**14, 14, 14**	**14, 14, 14**	2, **14**, 2	12, 12, 13	**14, 14, 14**	12, 12, 12	**14, 14, 14**
b2-sub-15	**15, 15, 15**	**15, 15, 15**	2, **15**, 2	13, 13, 14	**15, 15, 15**	13, 13, 13	**15, 15, 15**
b2-sub-16	**16, 16, 16**	**16, 16, 16**	2, **16**, 2	14, 14, 15	**16, 16, 16**	14, 14, 14	**16, 16, 16**
b2-sub-17	**17, 17, 17**	**17, 17, 17**	2, **17**, 2	15, 15, 16	**17, 17, 17**	15, 2, 2	**17, 17, 17**
b2-sub-18	**18, 18, 18**	**18, 18, 18**	2, **18**, 2	15, 15, 16	**18, 18, 18**	2, 2, 2	**18, 18, 18**
b2-sub-19	**19, 19, 19**	**19, 19, 19**	2, **19**, 2	16, 16, 17	**19, 19, 19**	2, 2, 2	**19, 19, 19**
b2-sub-20	**20, 20, 20**	**20, 20, 20**	2, **20**, 2	2, 2, 2	**20**, 21, **20**	2, 2, 2	**20, 20**, 2
G2-1-10	**2**, 25, **2**	**2**, 25, **2**	**2**, 25, **2**	**2, 2, 2**	25, 25, 25	**2, 2, 2**	**2, 2, 2**
G2-1-100	**2**, 25, **2**	**2**, 25, **2**	**2**, 25, **2**	22, 25, 22	21, 25, 21	3, 3, 3	**2, 2, 2**
G2-2-10	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2**, 20, **2**	**2, 2, 2**	**2, 2, 2**
G2-2-100	**2, 2, 2**	**2**, 22, **2**	**2, 2, 2**	21, 19, 25	8, 23, **2**	10, 7, 7	**2, 2, 2**
G2-4-10	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**
G2-4-100	**2, 2, 2**	**2**, 3, **2**	**2, 2, 2**	**2**, 17, 23	**2**, 6, **2**	**2**, 16, 16	**2, 2, 2**
G2-8-10	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**
G2-8-100	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
G2-1024-10	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**
G2-1024-100	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**	**2, 2, 2**



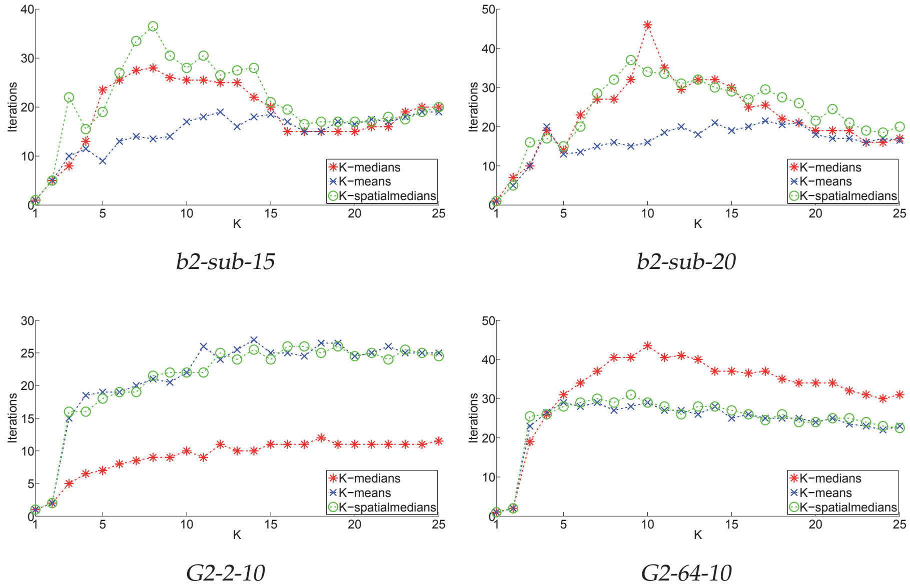
S4

USPS

b2-sub-5

b2-sub-10

Figure A1. *Cont.*



b2-sub-15

b2-sub-20

G2-2-10

G2-64-10

Figure A1. Median of the number of iterations needed for convergence with varying *K*.

References

1. Jain, A.K.; Murty, M.N.; Flynn, P.J. Data clustering: A review. *ACM Comput. Surv.* **1999**, *31*, 264–323.
2. Aggarwal, C.C.; Reddy, C.K. *Data Clustering: Algorithms and Applications*; CRC Press: New York, NY, USA, 2013.
3. Xie, X.L.; Beni, G. A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.* **1991**, *13*, 841–847.
4. Jain, A.K. Data clustering: 50 years beyond K-means. *Pattern Recognit. Lett.* **2010**, *31*, 651–666.
5. Zaki, M.J.; Meira, W., Jr. *Data Mining and Analysis: Fundamental Concepts and Algorithms*; Cambridge University Press: New York, NY, USA, 2014.
6. Saarela, M.; Hämäläinen, J.; Kärkkäinen, T. Feature Ranking of Large, Robust, and Weighted Clustering Result. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Jeju, Korea, 23–26 May 2017; pp. 96–109.
7. Lloyd, S. Least squares quantization in PCM. *IEEE Trans. Inf. Theory* **1982**, *28*, 129–137.
8. Khan, S.S.; Ahmad, A. Cluster center initialization algorithm for K-modes clustering. *Expert Syst. Appl.* **2013**, *40*, 7444–7456.
9. Arthur, D.; Vassilvitskii, S. K-means++: The advantages of careful seeding. In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, LA, USA, 7–9 January 2007; pp. 1027–1035.
10. Xu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678.
11. Hruschka, E.R.; Campello, R.J.; Freitas, A.A.; de Carvalho, A.C.P.L.F. A survey of evolutionary algorithms for clustering. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2009**, *39*, 133–155.
12. Han, J.; Kamber, M.; Tung, A. Spatial Clustering Methods in Data Mining: A Survey. In *Geographic Data Mining and Knowledge Discovery*; Miller, H., Han, J., Eds.; CRC Press: Boca Raton, FL, USA, 2001.
13. Huber, P.J. *Robust Statistics*; John Wiley & Sons Inc.: New York, NY, USA, 1981.
14. Rousseeuw, P.J.; Leroy, A.M. *Robust Regression and Outlier Detection*; John Wiley & Sons Inc.: New York, NY, USA, 1987; p. 329.
15. Hettmansperger, T.P.; McKean, J.W. *Robust Nonparametric Statistical Methods*; Edward Arnold: London, UK, 1998; p. 467.
16. Saarela, M.; Kärkkäinen, T. Analysing Student Performance using Sparse Data of Core Bachelor Courses. *J. Educ. Data Min.* **2015**, *7*, 3–32.

17. Kärkkäinen, T.; Heikkola, E. Robust Formulations for Training Multilayer Perceptrons. *Neural Comput.* **2004**, *16*, 837–862.
18. Croux, C.; Dehon, C.; Yadine, A. The *k*-step spatial sign covariance matrix. *Adv. Data Anal. Classif.* **2010**, *4*, 137–150.
19. Äyrämö, S. Knowledge Mining Using Robust Clustering. Ph.D. Thesis, Jyväskylä Studies in Computing 63, University of Jyväskylä, Jyväskylä, Finland, 2006.
20. Shannon, C.E. A mathematical theory of communication. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2001**, *5*, 3–55.
21. Strehl, A.; Ghosh, J. Cluster ensembles—A knowledge reuse framework for combining multiple partitions. *J. Mach. Learn. Res.* **2002**, *3*, 583–617.
22. Zhao, Q.; Fränti, P. WB-index: A sum-of-squares based index for cluster validity. *Data Knowl. Eng.* **2014**, *92*, 77–89.
23. Davies, D.L.; Bouldin, D.W. A cluster separation measure. *IEEE Trans. Pattern Anal. Mach. Intell.* **1979**, *PAMI-1*, 224–227.
24. Caliński, T.; Harabasz, J. A dendrite method for cluster analysis. *Commun. Stat. Theory Methods* **1974**, *3*, 1–27.
25. Ray, S.; Turi, R.H. Determination of number of clusters in k-means clustering and application in colour image segmentation. In Proceedings of the 4th International Conference on Advances in Pattern Recognition and Digital Techniques, Calcutta, India, 27–29 December 1999; pp. 137–143.
26. Rendón, E.; Abundez, I.; Arizmendi, A.; Quiroz, E.M. Internal versus external cluster validation indexes. *Int. J. Comput. Commun.* **2011**, *5*, 27–34.
27. Halkidi, M.; Batistakis, Y.; Vazirgiannis, M. On clustering validation techniques. *J. Intell. Inf. Syst.* **2001**, *17*, 107–145.
28. Kuncheva, L.I.; Vetrov, D.P. Evaluation of stability of k-means cluster ensembles with respect to random initialization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2006**, *28*, 1798–1808.
29. Handl, J.; Knowles, J. An evolutionary approach to multiobjective clustering. *IEEE Trans. Evolut. Comput.* **2007**, *11*, 56–76.
30. Jauhiainen, S.; Kärkkäinen, T. A Simple Cluster Validation Index with Maximal Coverage. In Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESAINN 2017), Bruges, Belgium, 26–28 April 2017; pp. 293–298.
31. Kim, M.; Ramakrishna, R. New indices for cluster validity assessment. *Pattern Recognit. Lett.* **2005**, *26*, 2353–2363.
32. Maulik, U.; Bandyopadhyay, S. Performance evaluation of some clustering algorithms and validity indices. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 1650–1654.
33. Arbelaitz, O.; Gurrutxaga, I.; Muguerza, J.; Pérez, J.M.; Perona, I. An extensive comparative study of cluster validity indices. *Pattern Recognit.* **2013**, *46*, 243–256.
34. Liu, Y.; Li, Z.; Xiong, H.; Gao, X.; Wu, J. Understanding of internal clustering validation measures. In Proceedings of the 2010 IEEE 10th International Conference on.Data Mining (ICDM), Sydney, Australia, 13–17 December 2010; pp. 911–916.
35. Agrawal, K.; Garg, S.; Patel, P. Performance measures for densed and arbitrary shaped clusters. *Int. J. Comput. Sci. Commun.* **2015**, *6*, 338–350.
36. Halkidi, M.; Vazirgiannis, M. Clustering validity assessment: Finding the optimal partitioning of a data set. In Proceedings of the IEEE International Conference on Data Mining (ICDM 2001), San Jose, CA, USA, 29 November–2 December 2001; pp. 187–194.
37. Lughofer, E. A dynamic split-and-merge approach for evolving cluster models. *Evol. Syst.* **2012**, *3*, 135–151.
38. Lughofer, E.; Sayed-Mouchaweh, M. Autonomous data stream clustering implementing split-and-merge concepts—Towards a plug-and-play approach. *Inf. Sci.* **2015**, *304*, 54–79.
39. Ordonez, C. Clustering binary data streams with K-means. In Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. San Diego, CA, USA, 13 June 2003; pp. 12–19.
40. Bagirov, A.M.; Yearwood, J. A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems. *Eur. J. Oper. Res.* **2006**, *170*, 578–596.
41. Karmitsa, N.; Bagirov, A.; Taheri, S. *MSSC Clustering of Large Data using the Limited Memory Bundle Method*; Discussion Paper; University of Turku: Turku, Finland, 2016.

42. Kärkkäinen, T.; Majava, K. Nonmonotone and monotone active-set methods for image restoration, Part 1: Convergence analysis. *J. Optim. Theory Appl.* **2000**, *106*, 61–80.
43. Kärkkäinen, T.; Kunisch, K.; Tarvainen, P. Augmented Lagrangian Active Set Methods for Obstacle Problems. *J. Optim. Theory Appl.* **2003**, *119*, 499–533.
44. Kärkkäinen, T.; Kunisch, K.; Majava, K. Denoising of smooth images using L^1-fitting. *Computing* **2005**, *74*, 353–376.
45. Pakhira, M.K.; Bandyopadhyay, S.; Maulik, U. Validity index for crisp and fuzzy clusters. *Pattern Recognit.* **2004**, *37*, 487–501.
46. Desgraupes, B. “ClusterCrit: Clustering Indices”. R Package Version 1.2.3., 2013. Available online: https://cran.r-project.org/web/packages/clusterCrit/ (accessed on 6 September 2017).
47. Milligan, G.W.; Cooper, M.C. An examination of procedures for determining the number of clusters in a data set. *Psychometrika* **1985**, *50*, 159–179.
48. Fränti, P.; Sieranoja, S. K-means properties on six clustering benchmark datasets. *Algorithms* **2017**, submitted.
49. Saarela, M.; Kärkkäinen, T. Do country stereotypes exist in educational data? A clustering approach for large, sparse, and weighted data. In Proceedings of the 8th International Conference on Educational Data Mining (EDM 2015), Madrid, Spain, 26–29 June 2015; pp. 156–163.
50. Verleysen, M.; François, D. The Curse of Dimensionality in Data Mining and Time Series Prediction. In Proceedings of the International Work-Conference on Artificial Neural Networks (IWANN), Cadiz, Spain, 14–16 June 2005; Volume 5, pp. 758–770.
51. Wartiainen, P.; Kärkkäinen, T. Hierarchical, prototype-based clustering of multiple time series with missing values. In Proceedings of the 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2015), Bruges, Belgium, 22–24 April 2015; pp. 95–100.

PV

FEATURE RANKING OF LARGE, ROBUST, AND WEIGHTED CLUSTERING RESULT

by

Mirka Saarela, Joonas Hämäläinen and Tommi Kärkkäinen 2017

PAKDD 2017 proceedings, Advances in Knowledge Discovery and Data Mining:
21st Pacific-Asia Conference

Feature Ranking of Large, Robust, and Weighted Clustering Result

Mirka Saarela, Joonas Hämäläinen, and Tommi Kärkkäinen
mirka.saarela,joonas.k.hamalainen,tommi.karkkainen@jyu.fi

Department of Mathematical Information Technology, P.O. Box 35 (Agora), FI-40014
University of Jyväskylä, Finland

Abstract. A clustering result needs to be interpreted and evaluated for knowledge discovery. When clustered data represents a sample from a population with known sample-to-population alignment weights, both the clustering and the evaluation techniques need to take this into account. The purpose of this article is to advance the automatic knowledge discovery from a robust clustering result on the population level. For this purpose, we derive a novel ranking method by generalizing the computation of the Kruskal-Wallis H test statistic from sample to population level with two different approaches. Application of these enlargements to both the input variables used in clustering and to metadata provides automatic determination of variable ranking that can be used to explain and distinguish the groups of population. The ranking method is illustrated with an open data and then, applied to advance the educational knowledge discovery from large scale international student assessment data, whose robust clustering into disjoint groups on three different levels of abstraction was performed in [19].

Keywords: Population analysis; Kruskal-Wallis test; Robust Clustering; Educational Knowledge Discovery

1 Introduction

Various large-scale educational assessments, like the Programme for International Student Assessment (PISA), regularly collect large amount of data characterizing worldwide student populations to assess and compare arrangements and policies between different educational systems [16]. Although data originating from these assessments are of high quality and publicly available, there is surprisingly little research activity on the secondary analysis. This is due to the technical complexities within the different representations and transformations of data and the lack of methods that allow advanced analysis of these large datasets [18]. One example of the complication of analyzing PISA datasets are the weights. Through complex sampling designs only certain students of the studied population are selected for the assessment and weights are used to indicate the number of students in the population that a sampled student represents. This means that these weights must be taken into account in all steps of the knowledge discovery to analyze the population instead of the collected sample (e.g., [20, 14]).

The purpose of this paper is to advance the educational knowledge discovery from a robust, weighted clustering result. There exists various clustering methods and approaches, like e.g. density-based, probabilistic, grid-based, and spectral clustering [2], together with their comparisons and evaluations (e.g., [6]). Although hierarchical methods allow summarization and exploration of a given dataset through the visual dendrogram, the basic form of the technique is not scalable to large number of observations because of the pairwise distance matrix requirement [25]. Moreover, it is not clear how to take into account the weights in hierarchical clustering as presented, e.g., in PISA datasets. On the other hand, in [3] a robust (cf. [24]) prototype-based clustering algorithm was developed that can handle large datasets with high and unknown sparsity patterns (i.e., tens of percents of missing values). This paper continues the efforts of [19], where the weighted enlargement of the above-mentioned algorithm was applied to create prototypes for the PISA 2012 dataset on three different levels of abstraction, with different numbers of clusters of the student population. The dynamic numbers of clusters were based on the use of multiple cluster indices (e.g., [13]) suggesting the number of clusters, again taking into account the weights (see [19] for details).

One main advantage of crisp, prototype-based clustering result is the guarantee of globally separable subsets of data. The data division is completely determined by the disjoint labels, typically integers from 1 to K for K clusters, encoding the clustering result. This means that, in order to make an interpretation of the result, one can consider and compare data distributions of both the actual variables used in clustering as well as relevant metadata. Note that the use of a hierarchical clustering method with locally greedy aggregation could produce clusters of arbitrary shape in the data space, which could then be difficult or even impossible to interpret because of the overlapping variable distributions.

The results in [19] were obtained with a robust clustering method with (available data) spatial median as the cluster prototype, which is characterized by the Laplace density distribution. A feature selection approach for the robust EM-algorithm with Laplace mixture models was suggested in [5]. There the feature selection, similarly to the construction of classifiers [11], referred to ranking the given input features to select the most important ones for the clustering result. Here, our purpose is, similarly to the techniques proposed in [23, 4], to assess the importance of variables with a given labeling. For this purpose, we apply the same method as in [5] where it was suggested that the feature ranking can be realized by Kruskal-Wallis (KW) statistical test. More precisely, the estimate of importance of a random variable with clustering provided labeling is supplied by the H statistics of the KW test [15], without need to compute the p-values and perform the actual statistical testing. To omit the hypothesis testing relaxes both the requirements of the KW test concerning the equal variances [15] and selection of appropriate distribution for the test statistics [21]. Moreover, because KW is a univariate method, it is easy to restrict the computation of the test statistic to the available values of a variable. This means utilizability with an arbitrary sparsity pattern.

Hence, one needs to generalize the KW H into the population level by using the weights. This is a difficult problem in statistics because of the reliance of KW on data ranking. After an extensive search for relevant literature and knowledge we were able to identify one related work generalizing KW [1], but not solving the problem at hand.

The only article that was identified as fully relevant was [22], which suggested a very natural generalization of KW for *integer weights*: create univariate data to compute the KW test statistic, where each observation is copied as many times as the integer weight suggests. Clearly, we then precisely test the target population and not the sample. The purpose of this paper is to propose an approximate extension of this approach to real-valued weights, by utilizing the classical bootstrapping [8], and to compare this to an analytically derived novel heuristic formula. Both of these approaches are tested and evaluated with two different existing clustering results from [19], when ranking both actual input variables and selected set of metadata variables.

2 On PISA data

The collected data of each PISA assessment, which since 2000 is conducted every three years, can be downloaded from the website[1] of the Organisation of Economical and Cultural Development (OECD). To select a reliable sample of the population, which in PISA are all 15-year-old students within the participating countries, the OECD applies a two-stage sampling design: First, schools attended by 15-year-old students are assigned to mutually exclusive groups based on explicit strata and schools from these groups are selected with probabilities proportional to their size. Then, students within those school are selected randomly with equal probability. The weight w_i assigned to each participating student i consists of the school base weight, the within-school base weight, and five adjustment factors, especially the one which compensates the non-participation of a sampled student [17]. Students that are sampled for the PISA test are asked to show their proficiencies in a cognitive test and answer a background questionnaire, which gathers information about demographics, activities, and attitudes of the students.

Table 1 details all PISA 2012 variables used in this study. The left-hand side of the table shows all the variables that in [19] were clustered on a population-level. The *ESCS* combines all information of the PISA background questionnaire that relate to the students' economic, social and cultural situation. The next five variables on the left-hand side of Table 1 are generally associated with the students' success in the PISA cognitive test, and the remaining nine variables relate directly to the students' mathematics performance, which was the main assessment area in PISA 2012. All of these 15 variables are so-called PISA scale indices that summarize many of the original questions in the students' background questionnaires by employing the Rasch model [17]. Since only a subset of all test item are allocated to each student (this is called rotated design), around one third of the values for these 15 variables are missing.

On the right-hand side of Table 1, the meta-variables to be used in this study are listed. The first eight variables of general interest are all PISA scale indices that were computed to summarize the information obtained from the ICT questionnaire, which assessed the students' computing availability and familiarity as well as their attitudes towards computers. The next and last set of variables in Table 1 are the plausible values (PVs) for each assessment domain (mathematics, reading, and science). PISA does not provide individual test performance scores. Instead, to reliably assess the proficiencies

[1] https://www.oecd.org/pisa/pisaproducts/

of populations, five PVs for each assessment domain are estimated with Bayesian statistics and reported for each student. Note that we have allocated only one line in the table per assessment domain for the three sets of PVs but there are five single PVs vectors per assessment domain, i.e., 15 PVs altogether, that are used in the analysis.

Table 1. PISA variables used in this study with the original variables (i.e., the data that was used for clustering) on the left-hand side and metadata (i.e., additional PISA variables used to explain the clustering result) on the right-hand side.

PISA data used for clustering		PISA metadata	
variable	**ID**	**variable**	**ID**
economic, social and cultural status	ESCS	ICT availability at home	ICTHOME
sense of belonging	BELONG	ICT availability at school	ICTSCH
attitude towards school: learning outcome	ATSCHL	ICT entertainment use	ENTUSE
attitude towards school: learning activities	ATTLNACT	ICT use at home for school-related tasks	HOMSCH
perseverance	PERSEV	use of ICT at school	USESCH
openness to problem solving	OPENPS	use of ICT in math lessons	USEMATH
self-responsibility for failing in math	FAILMAT	positive attitudes towards computers	ICTATTPOS
interest in mathematics	INTMAT	positive attitudes towards computers	ICTATTPOS
instrumental motivation to learn math	INSTMOT	plausible values 1-5 in mathematics	PVMATH
self-efficacy in mathematics	MATHEFF	plausible values 1-5 in reading	PVREADING
anxiety towards mathematics	ANXMAT	plausible values 1-5 in science	PVSCIENCE
self-concept in math	SCMAT		
behaviour in math	MATBEH		
intentions to use math	MATINTFC		
subjective norms in math	SUBNORM		

The PVs are random draws from the Bayesian posterior distribution of a student's ability. In PISA, the prior distribution is a population model that is estimated with a latent regression model. This latent regression computes the average proficiencies of examinee subgroups given evidence about the distribution and associations of collateral variables in the data. In PISA 2012, these collateral variables included to the latent regression model were all available student-level information besides their performance in the cognitive test [17, page 157]. That means, in particular, that also all variables listed in Table 1 except the 15 PVs themselves have been used to estimate the PVs, and therefore, the PVs cannot be seen totally independent of them. The likelihood of the success in test is a Rasch model, where the probability of success is a logistic function of the latent ability and some parameters (e.g. difficulties) of the test items. The obtained posterior distribution of a student's ability is specific for each student, since each student has different values of background variables and test results.

To sum up, student proficiencies in PISA are not directly observed. The PVs are estimates for group performance and only a selection of likely proficiencies for students that attained each score. Moreover, for the study at hand, it is important to note that all background information (i.e., all data that were clustered and all metadata except the PVs themselves) have been used in the latent regression model which contributes to the posterior distribution from which the PVs are drawn from.

3 Methods and formulations

Let $\{x_i\}_{i=1}^N$ be a given, multidimensional dataset, where N observations $x_i \in \mathbb{R}^n$ are given. Assume further that a given set of positive, real-valued weights $\{w_i\}_{i=1}^N$ is also given. Moreover, assume that there is a set of missing values in $\{x_i\}$ with unknown sparsity pattern. To identify this pattern, define the projection vectors $p_i, i = 1, \ldots, N$, that capture the existing variable values:

$$(p_i)_j = \begin{cases} 1, \text{if } (x_i)_j \text{ exists}, \\ 0, \text{otherwise}. \end{cases} \tag{1}$$

3.1 Robust, prototype-based clustering method for weighted sparse data

Let us briefly recapitulate the clustering method and the overall approach that was used hierarchically in [19], to produce three levels of disjoint clusters of PISA 2012 population with 2, 8, and 53 clusters, respectively.

The spatial median clustering algorithm, `k-SpatMeds`, proceeds similarly to any prototype-based method: first, an initial set of *complete* (i.e., no missing values) prototypes is created and second, these are refined by iteratively linking observations to the closest prototype whose value is then recomputed. The algorithm stops when there are no more changes in the linking. Mathematically, the score function that is locally minimized via the search procedure reads as follows:

$$\mathcal{J}_w = \sum_{j=1}^{K} \sum_{i=1}^{n_j} w_i \|\mathrm{Diag}\{p_i\}(x_i - c_j)\|_2. \tag{2}$$

Here, Diag transforms a vector into a diagonal matrix. The latter sum is computed over the subset of data attached to the jth cluster. One observes from (2) that to take into account the first-order alignment of the sample data with the corresponding population is straightforward. Moreover, projection of the Euclidean distance between the observation and the prototype to available values creates an implicit (secondary) weighting that favors more complete observations over the sparser ones in cluster creation. Algorithmically, one still needs to check that the iterative refinement of the prototypes does not introduce missing values to them, because the resulting set of cluster prototypes $\{c_i\}_{i=1}^K$ should be complete to allow proper interpretation. The robustness of this algorithm as thoroughly described and tested in [3], refers to the tolerance of both missing values and noisy data. To this end, one can apply the `k-SpatMeds` algorithm hierarchically to refine a set of disjoint clusters further.

3.2 Construction of test statistic for Kruskal-Wallis with weights

Next we describe two different approaches to estimate the test statistic H of the KW rank-test with real-valued weights. Because the KW test is univariate, we can restrict ourselves to univariate random variable.

Integer approximation with bootstrapping Let $\{x_i, l_i\}_{i=1}^N$ be the pairs of a univariate observation $x_i \in \mathbb{R}$ and its cluster-indicating label $l_i \in \mathbb{N}$, where $1 \leq l_i \leq K$ for K denoting the number of clusters/groups. Let $n_k = |C_k| = \{i \in \mathbb{N} \,|\, l_i = k\}$ determine the size of cluster C_k. The original formula for the KW H is given by [15]

$$H = \frac{12}{N(N+1)} \sum_{k=1}^{K} \frac{s_k^2}{n_k} - 3(N+1), \tag{3}$$

where r_i denotes the *rank* of observation x_i in global sorting and $s_k = \sum_{i \in C_k} r_i$ the sum of ranks in cluster C_k. When there are equal values (ties) in data, one can compute the mean rank of equal observations and share this value among the ties.

As described, $w_i \in \mathbb{R}$ measures the amount of population that the ith observation represents. If all w_i's are integers, then in [22] it was proposed how to modify the basic KW test: rank a derived dataset representing the whole population, where each (available) observation is copied as many times as the weight suggests. This approach is referred from now on as *Integerweighted-KW, IW-KW*. Note that when such an enlarged data are ranked we end up with multiple ties whose mean ranks are then shared. In the following, we describe a novel approach how to approximate this integer-weighted KW using a bootstrapping technique.

Let w denote an arbitrary, real-valued weight. The proposed technique is, firstly, based on approximating w up to an accuracy of the first decimal place. This can be simply done as follows: determine the two integers $w_l = \lfloor w \rfloor$ and $w_h = \lceil w \rceil$ that provide lower and upper bound of w as integers. Let then $d = [10 * (w - w_l)]$ be the rounded integer that encapsulates the decimal place 1 of w. Vector v of ten integers, which is created by repeating w_l $10-d$ times and w_h d times, provides an integer-approximating set of real-valued w in such a way that the mean of v is exactly the same as w up to the first decimal. For instance, for $w = 8.647$, $w_l = 8, w_h = 9$, and $d = 6$. And, for $v = \left[8\ 8\ 8\ 8\ 9\ 9\ 9\ 9\ 9\ 9\right]$, we have $mean\{v\} = 8.6$. Similarly, in order to create an integer-approximation of w being accurate to the second decimal place, it is enough to just redefine $d = [100 * (w - w_l)]$. Proceeding with the example just given, the integer vector of size 100 with 65 nines and 35 eights would yield to $mean\{v\} = 8.65$. For the general procedure, the result of the just proposed integer approximation of all weights is stored in the matrix $\mathbf{W} \in \mathbb{N}^{N \times D}$, where D is 10 when approximating the first decimal place and 100 for the second decimal place, correspondingly.

Next we suggest to use the classical bootstrapping [8] to create a set of KW test statistics based on the IW-KW and W. Hence, we create a random sample of indices $\{1, \ldots, N\}$ with replacement, and for the resulting unique set of indices $\tilde{I}$, for the available values of $\{x_i\}_{i \in \tilde{I}}$, we apply IW-KW. When this is repeated D times for all the integer columns of W, we obtain D different samples of the bootstrap estimate of the KW H. To this end, similarly as with the derivation of W, we then simply take the mean of the D-vector to produce the final approximation of H for the real-valued weights.

Analytic formula Let $\bar{r}$ denote the global mean rank (equal to $\frac{1+N}{2}$) and $\bar{r}_k$ the mean rank of the observations in cluster C_k. An equivalent form of the original formula (3) for the KW test statistic H, as given in [9], reads as

$$H = (N-1)\frac{\sum_{k=1}^{K} n_k(\bar{r}_k - \bar{r})^2}{\sum_{i=1}^{N}(r_i - \bar{r})^2}. \tag{4}$$

From this form, it is easy to derive an interpretation of the KW test statistic. With clusterwise $\bar{r}_k$ and global $\bar{r}$ mean ranks, the dividend presents sum of clusterwise variances multiplied by the size of the cluster whereas the divisor computes the global variance of ranks. Hence, when the weights represent the number of samples in the population, it is straightforward to derive an analogous formula to (4) in the population level. Hence, let $\bar{r}_w = \frac{\sum_{i=1}^{N} w_i r_i}{\sum_{i=1}^{N} w_i}$ be the weighted average rank and $(\bar{r}_w)_k$ the weighted average rank of cluster C_k. Then, we define

$$H_w = \frac{\sum_{k=1}^{K}(\sum_{i \in C_k} w_i)((\bar{r}_w)_k - \bar{r}_w)^2}{\sum_{i=1}^{N} w_i(r_i - \bar{r}_w)^2}. \tag{5}$$

Note that we have omitted the multiplier $(N-1)$ from (4), which would be generalized into $(\sum_i w_i - 1)$ to represent the whole population. With PISA 2012 weights, which align the half a million students sample to the 24 million population, this means we do not include multiplication of H_w by over 24 million. Because the final ranking of variables, as suggested in [5], is based on sorting the H values of the variables in descending order, this omission does not change the result.

4 Evaluation

Implementation We computed the KW rank-test H test statistics for real-value weighted data with two approaches, as described in Section 3. The bootstrapping with the IW-KW was tested with two different Ws. We will refer to the bootstrapping based method as Bootstrap KW. Further, Bootstrap KW with D = 10 refers to the one decimal place approximation of real-valued weights. Similarly, the two decimal place approximation is referred as Bootstrap KW with D = 100. In addition, the KW test statistics were computed directly from formula (5). In the following, this is shortly referred as Analytic KW. The two clustering results that are used in the experiments corresponded to 8 (*Labels 1*) and 53 (*Labels 2*) clusters from [19] in the second and third levels of refinement, respectively. The first result in [19] with the two clusters is excluded here, since the KW rank-test exactly generalizes the MannWhitney U-test for the two groups.

To speed up the computations, we implemented a parallel version of Bootstrap KW with Matlab PCT, SPMD blocks and message passing functions. The tests were run in Matlab 8.5.0 environment by using a cluster of 8 nodes. Each node consists of Intel Xeon CPU E7-8837 with 8 cores and 128 GB RAM. Each worker in the distributed computations corresponds to one of the 64 cores. Since Bootstrap KW computes the KW H values independently for each variable in a loop, those loop iterations can be easily parallelized with SPMD blocks. First, each worker reads one column of variable values from the data matrix and the corresponding sparsity indicator (1). Next, each worker computes the KW H values by utilizing its local data. Finally, results are aggregated and rankings for the variables based on the H values are formed. The number of workers is equal to the number of variables in all parallel runs.

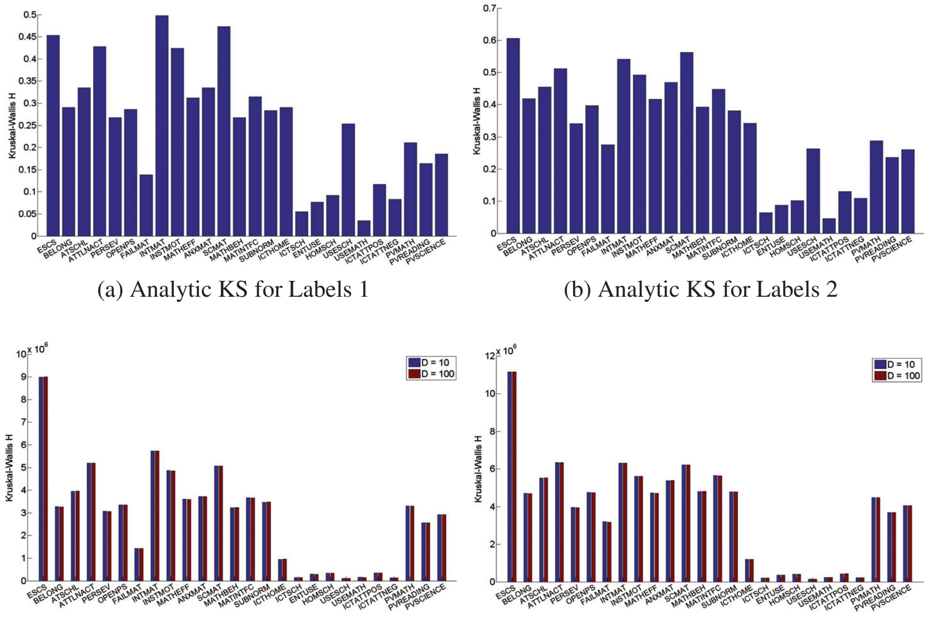
The five individual PVs for mathematics, reading, and science, as given in Table 1, were first treated as independent variables, such that five H values were computed for them. The final value of the test statistic was then taken as the mean of these according to the recommended way of analysis in [17].

Results To generally test the proposed approaches, we first used the Iris data from UCI machine-learning repository. For this, we created random integer weights in the range 5–25 and newly generated the data for each run. The KW H values for Analytic KW and Bootstrap KW $D = 100$ approaches gave the same variable ranking results in eight out of ten runs. After adding 5% zero-mean uniformly distributed noise to make weights real-values, we obtained the same ranking order for the different approaches in nine out of ten runs. Moreover, similarly as in [7], features 4 and 3 were always selected as the important ones while features 1 and 2 were always last in the list. When we used the same data for each run the ranking order was always the same.

Table 2 summarizes all ranking for the combined (originally clustered and meta) PISA data. In the table, the last column *rank of rankings* indicates for each variable the total rank, i.e. the rank of the sum of rankings of all methods on both labeling levels.

Table 2. Rankings for full (original and metadata) variables for the different analysis approaches for both PISA clustering results.

	Labels 1			Labels 2			
		Bootstrap KW			Bootstrap KW		rank of
Variable	Analytic KW	$D = 10$	$D = 100$	Analytic KW	$D = 10$	$D = 100$	rankings
ESCS	3	1	1	1	1	1	1
BELONG	11	13	13	9	13	13	12
ATSCHL	7	6	6	7	7	7	6
ATTLNACT	4	3	3	4	2	2	3
PERSEV	15	15	15	15	16	16	15
OPENPS	12	11	11	11	11	11	11
FAILMAT	20	18	18	17	18	18	19
INTMAT	1	2	2	3	3	3	2
INSTMOT	5	5	5	5	6	6	5
MATHEFF	9	9	9	10	12	12	9
ANXMAT	6	7	7	6	8	8	7
SCMAT	2	4	4	2	4	4	4
MATHBEH	14	14	14	12	9	9	13
MATINTFC	8	8	8	8	5	5	8
SUBNORM	13	10	10	13	10	10	10
ICTHOME	10	19	19	14	19	19	17
ICTSCH	25	24	24	25	25	25	25
ENTUSE	24	22	22	24	22	22	22
HOMSCH	22	21	21	23	21	21	21
USESCH	16	26	26	18	26	26	23
USEMATH	26	23	23	26	23	23	24
ICTATTPOS	21	20	20	21	20	20	20
ICTATTNEG	23	25	25	22	24	24	26
PVMATH	17	12	12	16	14	14	14
PVREADING	19	17	17	20	17	17	18
PVSCIENCE	18	16	16	19	15	15	16



(a) Analytic KS for Labels 1

(b) Analytic KS for Labels 2

(c) Bootstrap KS for Labels 1

(d) Bootstrap KS for Labels 2

Fig. 1. KW H values for two clustering results for the combined (originally clustered and meta) PISA data determined with the analytic and the two bootstrap KW approaches.

KW H values for both clustering results are shown in Figure 1. As can be seen from Table 2, variable rankings between the analytic and the bootstrapped results are highly similar with the exception that variable *USESCH* had a ranking difference 10 for Labels 1 and ranking difference 8 for Labels 2. In addition, variable *ICTHOME* had ranking difference 9 for Labels 1 and ranking difference 5 for Labels 2.

The Kendall's tau distance (see [10]) provides a way to compute distance between two ranking lists with an equal set of variables. The Kendall's tau distance is equal to the bubble sort algorithm steps to convert one list to the same order as the other one. If m is the number of elements in the list, then the maximum value for the Kendall's tau distance is $m(m-1)/2$ which is typically used to normalize this distance metric. Thus, the Kendall's tau distance is limited to an interval $[0,1]$, where value 0 refers to the identical lists and value 1 to the case where one list is the reverse of the other list. The Kendall's tau distances between the Analytic KW and Bootstrap KW with $D = 100$ were 0.1015 for Labels 1 and 0.1138 for Labels 2. This concludes that, overall, the rankings are highly similar as measured by the Kendall's tau distance.

Bootstrap KW with $D = 10$ and Bootstrap KW with $D = 100$ gave identical rankings for the variables. Experimentally, it seems that approximation of the real-valued weights using just the first decimal place ($D = 10$) is accurate enough. However, for a few variables slight differences can be noticed from the Figures 1c and 1d. We also computed speedups for the distributed Bootstrap KW. We measured running time for the first variable computations by using a serial implementation of the Bootstrap KW, and multiplied this with the total number of variables to get an estimate for the serial implementation running time. Further, we measured running time for the corresponding parallel implementation. Thus, parallel Bootstrap KW with $D = 100$ gives 34 × speedup compared to sequential code for Labels 1 and 35 × speedup for Labels 2. Correspondingly, parallel Bootstrap KW with $D = 10$ gives 28 × speedup for Labels 1 and 33 × speedup for Labels 2. In practice, this means that using the distributed version enables one to carry out the whole cluster analysis chain in realtime.

As expected, we see from Table 2 and Figure 1 that the actually clustered variables generally contribute more to the clustering result than the metadata variables. However, this first observation does not hold for all variables: The metadata PVs in mathematics were more important than the level of self-responsibility for failing in mathematics (see row *FAILMAT* in Table 2), which was clustered. Generally, the PVs are the most important variables from the metavariables. This ranking result makes sense because the clustered variables are, as explained in Section 2, part of the posterior model from which the PVs were sampled. Moreover, most of the clustered variables are directly associated with the students' mathematics proficiencies. Hence, the PVs in mathematics should be important variables when explaining the clustering result and, thus, these observations support the validity of our results.

As can be seen in Table 2, the students' *ESCS* is the most important variable determining the different clusters. This was already assumed in [19] where the most distinguishing country clusters were those that showed different stages of development. Moreover, the students' *ESCS* is the single variable in the whole PISA data, which accounts for most of the variance in performance [16]. Therefore, it is reasonable to assume that the variable that explains the mathematics proficiency the most, is also the most important when variables associated with the mathematics performance, are clustered. The students' *ESCS* takes not only the highest parental education and occupation into account but also the students' home possessions. Therefore, the *ICTHOME*, which summarizes the home possessions in the ICT area, is partly associated with the students' *ESCS* [17, page 132]. Hence, it seems reasonable that *ICTHOME* is next to the PVs one of the most important variables from the metadata (see Table 2).

To sum up, weighted enlargements with all approaches proposed in Section 3 successfully enabled ranking of input and metadata. Triangulation for both actual input and metadata by using two clustering results of a PISA dataset and two different algorithms/formulae showed very similar results for all methodological approaches and also for the two clustering results that were analyzed. Hence, it seems that the interpretation is not an artifact of the method used to analyze the data or only a result of the particular sample, but reflects genuine and overarching aspects of the data [12].

5 Discussion and conclusions

Large scale educational assessment data provide interesting and high quality resources for educational knowledge discovery. Although the data from these assessments are made available to the public a scarce pool of research outcomes exist that make use of those rich datasets because of the technical difficulties in them. Only one study [19] was identified, in which the whole PISA 2012 contextual data were clustered by taking the complexities of these data (especially the sparsity and the weights) into account. However, the work in [19] lacked a clear frame how to assess the importance of individual variables to interpret the clustering results.

In this study, we proposed weighted enlargements of the KW H test with different approaches, which as an independent statistical problem is not trivial. All approaches successfully enabled ranking of input and metadata. In particular, when applied to the two clustering results in [19], all approaches supported the finding that the students' *ESCS* is the most important variable determining the clusters—a fact that was also hypothesized in [19] but could not be statistically shown in there. Moreover, also the ranking of the other variables seem to support the interpretations made in [19].

The y-scales of Figures 1c and 1d illustrate the very large size of the KW test statistic(s) H for a large population, which in our case is characterized by over 24 million students worldwide. Hence, even if the nonparametric KW test can be used for testing large samples [9], the actual hypothesis testing seems practically useless. We tested the computation of the p-values for the original sample, for both clustering results and for all data and metadata variables, and found in each case that the p-value was equal to zero up to six decimal places. Hence, the hypothesis test itself does not provide any useful information for educational knowledge discovery.

Based on the high similarity of the results of the different ranking approaches, we suggest the direct KW formula with weights to be used for quick evaluation of significance of a variable on the population level. If the weighted estimates are used to derive, e.g., confidence intervals for the test statistics and the resulting rankings, the bootstrap-based approach should be used. This approach is also better aligned to the existing literature [8, 5, 22]. To this end, we conclude that the proposed approach supports quantified educational knowledge discovery from PISA and similar large-scale educational datasets.

Acknowledgments

The authors would like to thank PhD Salme Kärkkäinen for her kind and valuable suggestion to use bootstrapping.

References

1. Acar, E.F., Sun, L.: A Generalized Kruskal–Wallis Test Incorporating Group Uncertainty with Application to Genetic Association Studies. Biometrics 69(2), 427–435 (2013)
2. Aggarwal, C.C., Reddy, C.K.: Data clustering: algorithms and applications. CRC Press (2013)
3. Äyrämö, S.: Knowledge Mining Using Robust Clustering, Jyväskylä Studies in Computing, vol. 63. University of Jyväskylä (2006)
4. Ceccarelli, M., Maratea, A.: Assessing Clustering Reliability and Features Informativeness by Random Permutations. In: Knowledge-Based Intelligent Information and Engineering Systems: 11th International Conference, XVII Italian Workshop on Neural Networks, Proceedings. pp. 878–885. Springer (2007)

5. Cord, A., Ambroise, C., Cocquerez, J.P.: Feature selection in robust clustering based on Laplace mixture. Pattern Recognition Letters 27(6), 627–635 (2006)
6. Crabtree, D., Andreae, P., Gao, X.: QC4 - A Clustering Evaluation Method. In: Advances in Knowledge Discovery and Data Mining: 11th Pacific-Asia Conference, Proceedings. pp. 59–70. Springer (2007)
7. Dash, M., Liu, H.: Feature selection for clustering. In: Advances in Knowledge Discovery and Data Mining: 4th Pacific-Asia Conference, Proceedings. pp. 110–121. Springer (2000)
8. Efron, B.: Bootstrap Methods: Another Look at the Jackknife. Annals of Statistics 7, 1–26 (1979)
9. Elamir, E.A.: Kruskal-Wallis Test: A Graphical Way. International Journal of Statistics and Applications 5(3), 113–119 (2015)
10. Fagin, R., Kumar, R., Sivakumar, D.: Comparing Top K Lists. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 28–36. Society for Industrial and Applied Mathematics (2003)
11. Fung, P.C.G., Morstatter, F., Liu, H.: Feature Selection Strategy in Text Classification. In: Advances in Knowledge Discovery and Data Mining: 15th Pacific-Asia Conference, Proceedings. pp. 26–37. Springer (2011)
12. Gifi, A.: Nonlinear multivariate analysis. Wiley (1991)
13. Kim, Y., Lee, S.: A Clustering Validity Assessment Index. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 602–608. Springer (2003)
14. Koskela, A.: Exploring the differences of Finnish students in PISA 2003 and 2012 using educational data mining. Jyväskylä Studies in Computing, University of Jyväskylä (2016)
15. Kruskal, W., Wallis, W.: Use of Ranks in One-Criterion Variance Analysis. Journal of the American statistical Association 47(260), 583–621 (1952)
16. OECD: PISA 2012 Results: Excellence Through Equity: Giving Every Student the Chance to Succeed (Volume II). PISA, OECD Publishing (2013)
17. OECD: PISA 2012 Technical Report. OECD Publishing (2014)
18. Rutkowski, L., Rutkowski, D.: Getting It ”Better”: The Importance of Improving Background Questionnaires in International Large-Scale Assessment. Journal of Curriculum Studies 42(3), 411–430 (2010)
19. Saarela, M., Kärkkäinen, T.: Do Country Stereotypes Exist in PISA? A Clustering Approach for Large, Sparse, and Weighted Data. In: Proceedings of the 8th International Conference on Educational Data Mining. pp. 156–163 (2015)
20. Saarela, M., Kärkkäinen, T.: Weighted Clustering of Sparse Educational Data. In: Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. pp. 337–342 (2015)
21. Spurrier, J.D.: On the null distribution of the Kruskal–Wallis statistic. Nonparametric Statistics 15(6), 685–691 (2003)
22. Tölgyesi, C., Bátori, Z., Erdős, L.: Using statistical tests on relative ecological indicator values to compare vegetation units–Different approaches and weighting methods. Ecological Indicators 36, 441–446 (2014)
23. Verde, R., Lechevallier, Y., Chavent, M.: Symbolic clustering interpretation and visualization. The Electronic Journal of Symbolic Data Analysis 1(1) (2003)
24. Yang, H., Zhao, D., Cao, L., Sun, F.: A Precise and Robust Clustering Approach Using Homophilic Degrees of Graph Kernel. In: Advances in Knowledge Discovery and Data Mining: 20th Pacific-Asia Conference, Proceedings. pp. 257–270. Springer (2016)
25. Zaki, M.J., Meira Jr., W.: Data Mining and Analysis: Fundamental Concepts and Algorithms. Cambridge University Press (2014)

PVI

CLUSTERING-BASED REFERENCE POINTS SELECTION FOR THE MINIMAL LEARNING MACHINE

by

Joonas Hämäläinen, Tommi Kärkkäinen and João P. P. Gomes 2018

Manuscript

Clustering-based reference points selection for the minimal learning machine

Joonas Hämäläinen[1], Tommi Kärkkäinen[1], João P. P. Gomes[2]

[1] *University of Jyvaskyla, Faculty of Information Technology, P.O. Box 35, FI-40014 University of Jyvaskyla, Finland*
[2] *Federal University of Cearà - UFC, Department of Computer Science, Fortaleza-CE, Brazil*

Abstract

Randomized learning machines are scalable and increasingly popular supervised methods, which may suffer from a lack generalization ability and stability due to the machines' random nature. Of such techniques, the minimal learning machine (MLM) is based on learning the mapping between input and output distance matrices, where distances are calculated for a subset of points called reference points. In the basic formulation, the reference points are selected randomly. Such a strategy coincides with classical suggestions to initialize unsupervised clustering. Thus, in this paper, we assess several clustering-based methods for selecting reference points to improve the performance of the MLM in regression tasks. We also propose and test an efficient method for solving the multilateration problem related to MLMs. Based on an extensive empirical evaluation, we conclude that the proposed methods provide scalable and useful extensions of the original MLM. In particular, for a small number of reference points, the clustering-based methods outperform the random selection method.

1 Introduction

Minimal learning machine (MLM, [1]) is a supervised learning algorithm based on linear mapping between input and output distance matrices. In the original formulation, the MLM's distance matrices comprise the distances between all the training set points and a subset of randomly selected training points, named reference points. According to this definition, the MLM can be classified as a learning algorithm that has a nonlinear approximation capability based on a random projection of the input points. Other examples of popular randomized learning machines include extreme learning machine (ELM, [2]) and random vector functional link (RVFL, [3]).

Interest in randomization-based methods has been increasing because such methods usually require only a few hyperparameters to be tuned and are based on non-iterative training procedures [4]. Moreover, such methods have reported remarkable performances in various applications, such as predicting software defects [5], forecasting electricity loads [6], and classifying images [7]. Despite the recent success, due to the random learning machines' random nature, they may suffer from the non-optimal generation of some parameters [8, 9], which, in turn, can have a negative impact on the method's generalization capability and stability. The randomness of the MLM can discard potentially useful data that could be important for the induction process. This effect is even more likely to happen when the number of reference points is small [1].

Another issue that affects generalization in the MLM is the number of reference points, as empirically shown in [1]. The more points the MLM uses, the more complex the model becomes. Thus, concerning the generalization performance of the MLM, two issues are relevant: *i*) the number of reference points, K, and *ii*) how to choose a specific combination of samples among $\binom{N}{K}$ possible ones from the training set. Therefore, finding a relatively small set of suitable reference points while attaining high accuracy rates is highly desirable.

Clustering is an essential task in data mining and machine learning. It aims to group data points into clusters, so that the observations in a cluster are similar to each other and dissimilar to the data in other clusters, with a given similarity measure, such as the Euclidean distance. Over many decades [10], a large pool of different clustering algorithms have been developed based on, e.g., tree structures (hierarchical) [11], squared error [12], density [13], and neural networks [14]. However, the classical and most common division is to classify clustering algorithms simply into hierarchical and partitional categories [15]. Typically, in hierarchical clustering, a tree structure of the clusters from one cluster (divisive) or from all points (agglomerative) is constructed. In contrast, a partitional clustering algorithm divides the dataset into a single-layered clustering structure. Some partitional clustering algorithms, such as the well-known K-means, are sensitive to the initialization [16, 17], the selection of the initial points or the initial partition. The selection of the initial points for partional clustering and partitional clustering analysis as a whole have similar aspects in the problem that they try to solve as the reference points selection for the MLM.

During the last few years, many studies have been carried out to improve and augment the basic form of the MLM, related to missing values [18, 19], outliers [20], ensemble learning [21], speeding up the computations [21, 22], combining the MLM and the ELM [23], the reject option in classification [24], and co-training [25]. Moreover, the MLM has shown its potential in applications and related classifier comparisons, such as in mobile robot localization [26, 27]. Recently, a new suggestion to select reference points for the MLM was given in [28], where an opposite neighborhood-based approach was proposed. The method is based on finding reference points around the class boundaries with a strategy that any point from a subset of points from the class boundary area is prohibited from being selected as a reference point.

This work tackles the problem of selecting reference points in the MLM by proposing several clustering-based methods. The basic hypothesis is that a set of well-spread reference points in the data space will improve the performance of the MLM compared to random selection of the reference points. As said, this is also the purpose of the proper selection of the initial cluster centroids [16, 17]. We exclude the reference points selection method proposed in [28] from the experiments because the opposite neighborhood method is designed for classification tasks, whereas we focus on regression tasks. We validate the contributions of this paper through experiments with 13 regression datasets. Based on the experimental results, we can conclude that the proposed methods are valid alternatives to the original random selection method because they can build a more accurate model when a small number of selected reference points are used. The remainder of the paper is organized as follows. Section 2 presents the formulation of the MLM. In Section 3, the proposed methods for selecting reference points are described. Section 4 gives experimental results for the proposed methods and the baseline. Finally, Section 5 provides conclusions.

2 Minimal learning machine

The MLM is a distance-based machine learning method with a randomized kernel. The basic algorithm [1, 29] is composed of two main steps: *i*) regression estimation using a distance-based kernel and *ii*) distance-based interpolation of the new output. For clarity, we describe these two steps in the following. For this purpose, let $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$ be a set of training inputs, where $\mathbf{x}_i \in \mathbb{R}^n$, and $\mathcal{Y} = \{\mathbf{y}_i\}_{i=1}^N$ the set of the corresponding outputs, for $\mathbf{y}_i \in \mathbb{R}^m$, respectively. Moreover, we define the set of (input) reference points $\mathcal{R} = \{\mathbf{r}_k\}_{k=1}^K$ as a non-empty subset of $\mathcal{X}$, $\mathcal{R} \subseteq \mathcal{X}$, and let $\mathcal{T} = \{\mathbf{t}_k\}_{k=1}^K$ refer to the outputs of the corresponding reference inputs, i.e., $\mathbf{r}_k \mapsto \mathbf{t}_k$.

Next, we define two distance matrices $\mathbf{D}_x \in \mathbb{R}^{N\times K}$ and $\mathbf{D}_y \in \mathbb{R}^{N\times K}$ using the Euclidean distance $\|\cdot\|$ as follows:

$$\mathbf{D}_x = \big[\|\mathbf{x}_i - \mathbf{r}_k\|\big] \quad i = 1, \dots, N,\ k = 1, \dots, K; \tag{1}$$

$$\mathbf{D}_y = \big[\|\mathbf{y}_i - \mathbf{t}_k\|\big] \quad i = 1, \dots, N,\ k = 1, \dots, K. \tag{2}$$

The key idea for the first step of the MLM is the assumption of a regression model between the distance matrices: $\mathbf{D}_y = g(\mathbf{D}_x) + E$, where E denotes the residuals/error in this transformation. Assuming that the unknown regression model g is of the linear form $\mathbf{B} \in \mathbb{R}^{K\times K}$ allows one to estimate it, by using the well-known ordinary-least-squares formulation, as follows

$$\mathbf{B} = \left(\mathbf{D}_x^T\mathbf{D}_x\right)^{-1}\mathbf{D}_x^T\mathbf{D}_y. \tag{3}$$

To assure unique solvability of (3), $\mathbf{D}_x^T\mathbf{D}_x$ should be replaced with $\mathbf{D}_x^T\mathbf{D}_x + \varepsilon\mathbf{I}$ for $\varepsilon > 0$ and $\mathbf{I} \in \mathbb{R}^{K\times K}$ denoting the identify matrix. The linear mapping $\mathbf{B}$ is the result of the first step of the MLM.

For the second step, let $\tilde{\mathbf{x}}$ be a new input vector whose output needs to be estimated. Thus, based on the distance regression model from the first step, we seek for the corresponding output $\tilde{\mathbf{y}}$ that satisfies

$$\|\tilde{\mathbf{y}} - \mathbf{t}_k\| \approx \delta_k \quad \forall k = 1, \dots, K, \tag{4}$$

where

$$\delta = \big[\|\tilde{\mathbf{x}} - \mathbf{r}_k\|\big]_{k=1}^K \mathbf{B}.$$

The solution to the multilateration problem (4) can also be obtained using the least-squares formulation by letting

$$\tilde{\mathbf{y}}^* = \operatorname{argmin} \mathcal{J}(\tilde{\mathbf{y}}), \quad \text{where} \ \ \mathcal{J}(\tilde{\mathbf{y}}) = \sum_{k=1}^K \left(\|\tilde{\mathbf{y}} - \mathbf{t}_k\|^2 - \delta_k^2\right)^2. \tag{5}$$

As stated in [1], there exist many possible solvers for (5). However, because of the quadratic (actually quartic) form of the problem, the most efficient local solver is provided by the classical Newton's method with an iteration step $l \mapsto l+1$ as follows: $\tilde{\mathbf{y}}^{l+1} = \tilde{\mathbf{y}}^{l} - \left[\nabla^2 \mathcal{J}(\tilde{\mathbf{y}}^{l})\right]^{-1} \nabla \mathcal{J}(\tilde{\mathbf{y}}^{l})$ (e.g., [30] and articles therein). More precisely, it is straightforward to calculate the derivative and Hessian of (5) in a matrix-vector form. The most important facet for the efficiency of Newton's method is then the initial guess, which should be accurate enough to assure quadratic convergence which is obtained only locally. We suggest a simple initialization, based on considering (4) and (5) in a completely componentwise form

$$\sum_{k=1}^{K} \sum_{i=1}^{m} \left(\tilde{\mathbf{y}}^0 - \mathbf{t}_k\right)_i^2 \approx \sum_{k=1}^{K} \delta_k^2 = \sum_{k=1}^{K} \sum_{i=1}^{m} \frac{1}{m} \delta_k^2,$$

which yields

$$\tilde{\mathbf{y}}_i^0 = \frac{1}{K} \sum_{k=1}^{K} \left(\mathbf{t}_k \pm \frac{\delta_k}{\sqrt{m}}\right)_i.$$

In practice, all sign combinations for all components of $\tilde{\mathbf{y}}^0$ should be tested, and the candidate with the smallest value of the cost function in (5) selected. For safeguarding, because of the heuristic initialization strategy, we apply a general nonlinear programming solver if Newton's method fails.

3 Clustering-based selection of reference points

In the following, we propose four clustering-based methods for the reference point selection problem, two nondeterministic and two deterministic methods. All the methods are based on a common strategy, where the reference point selection is performed in the input space only, and then simply the corresponding points (indices) are selected as the output references. Therefore, in the following, we focus on the input space processing only when we describe the proposed methods.

Today, the K-means++ initialization method [17] is the most popular method for K-means initialization. The first method that we propose is to use the K-means++ initialization with the Euclidean distance to select the reference points. See [31] for a depiction of the algorithm. From now on, we refer to this approach as reference points selection with K-means++ (RS-K-means++).

The second proposed method first runs the K-means++ initialization with the Euclidean distance and then refines the initial prototypes with Lloyd's algorithm [12] until convergence. Finally, the observation closest to each final prototype (medoid) is picked as the reference point. These closest points then establish the set of selected reference points. From now on, this method is referred to as the RS-K-medoids++. The RS-K-medoids++ and RS-K-means++ methods are nondeterministic, because of the random sampling of the initial prototypes based on the Euclidean distance constructed probability distribution (see [31] and articles therein).

Unweighted pair group method with arithmetic mean (UPGMA) [11] is an agglomerative clustering algorithm, which starts clustering from the initial state in which each point forms one cluster. Then, in each step, two clusters that have the smallest average distance between the cluster members are joined together. The third proposed method utilizes the UPGMA first, then computes the mean prototypes for each cluster, and finally, again selects the point closest to the prototype as a reference point. Similarly to the RS-K-medoids++ method, these closest points construct the set of selected reference points. We refer to this method as the RS-UPGMA method.

The fourth proposed method is based on the maximin clustering initialization algorithm [32]. The original method starts with a random initial point and then picks each new point similarly as in the K-means++ method. However, differently from the K-means++ method, a point that has the largest distance to the closest already selected point is chosen as a new point. Our modification of the maximin first selects the point closest to the data mean as the first point, conceiving the whole algorithm as completely deterministic. This proposal is referred to as the RS-maximin method. The latter two proposed methods, the RS-UPGMA and the RS-maximin, are deterministic.

The proposed methods are straightforward and easy to implement. Moreover, the MLM has only one hyper-parameter, the number of reference points K, to be selected. The proposed methods keep this unchanged because the only parameter that is needed for the proposed methods is the number of reference points. A summary of the methods is shown in Table 1, where the time complexities are also presented with respect to the number of observations N. The RS-K-means++, RS-K-medoids++, and RS-maximin methods have a linear time complexity. The optimal time complexity for the UPGMA is quadratic [33]. Therefore, the RS-UPGMA method has quadratic complexity because the post-processing after UPGMA clustering has a linear time complexity. Because the MLM training phase has a time complexity of $\mathcal{O}(K^2N)$

[1], the reference points selection method with a linear computational cost (with respect to N) and the ability to build an accurate model with a small K is highly desirable.

Table 1: Summary of the proposed methods.

Method	Based on	Deterministic	Type	Complexity
RS-K-means++	K-means++ initialization	No	Partitional	$\mathcal{O}(N)$
RS-K-medoids++	K-means++ initialization and K-medoids clustering	No	Partitional	$\mathcal{O}(N)$
RS-UPGMA	Aggloremerative clustering	Yes	Hierarchical	$\mathcal{O}(N^2)$
RS-maximin	Maximin clustering initialization	Yes	Partitional	$\mathcal{O}(N)$

4 Experiments and results

4.1 Experimental setup

We selected 11 real datasets and two synthetic datasets (S1,BNK) to evaluate the reference point selection methods. The selected datasets are summarized in Table 2. All datasets have one-dimensional output values. The S1 dataset was modified for a regression task. We randomly selected 1000 observations from the S1 dataset, scaled the values to the range of $[0, 1]$, and then the output values $f(x_1, x_2)$ were computed with a function $\sin(2 * \pi * x_1) + \sin(2 * \pi * x_2)$. The original S1 dataset is available at (`http://cs.uef.fi/sipu/datasets/`). The rest of the datasets are available at (`http://www.dcc.fc.up.pt/~ltorgo/Regression/DataSets.html`).

For rigorous comparison of the behavior of different reference points selection algorithms, we combine two basic and well-known methods in model selection and assessment: division into train-validation-test sets and cross-validation (see, e.g., [34], Chapter 7). More precisely, we use the 3-DOB-SCV [35] approach to divide each dataset into a training set and a test set. Therefore, the test set is forced to approximately follow the same distribution as the training set, thus making the comparison more reliable for the case where the concept drift is not considered. Moreover, we archive three training sets and three test sets for each dataset, that have 2/3 and 1/3 of the number of points, respectively. In training, we used the 10-DOB-SCV approach to select the optimal number of reference points. Thus, 18/30 of the number of points were used to train the model, and 2/30 of the number of points were

Table 2: Characteristics of the datasets.

Dataset	**# Observations**	**# Features**
Auto Price (AP)	159	15
Servo (SRV)	167	4
Breast Cancer (BC)	194	32
Computer Hardware (CHA)	209	6
Boston Housing (BH)	506	13
Stocks (STC)	950	9
S1 (S1)	1000	2
Bank (BNK)	4499	8
Ailerons (ALR)	7129	5
Computer Activity (CA)	8192	12
Elevators (ELV)	9517	6
California Housing (CH)	20640	8
Census (CNS)	22784	8

used to compute the validation error. Therefore, we have a two-level division of the datasets.

The quality of the models was evaluated using root mean square error (RMSE). In addition to validation error computation, test error was computed for all 10-DOB-SCV training sets, resulting in 10 test RMSEs for each training set and 30 test RMSEs for the whole dataset. In training, the number of reference points K was varied in a range of 1% to 100% of the training data points. The step size was set to 1%. We used Newton's method with the proposed heuristic initialization strategy, as described in Section 2, for solving the multilateration problem in (5). Furthermore, we ran the experiments with the bias $\varepsilon\mathbf{I}$ and without it (see Section 2). However, because there was no clear benefit of computing $\mathbf{B}$ with the bias, for clarity, only the results without the bias are included. We scaled all training datasets to a range of $[0, 1]$, and corresponding training dataset variable-wise minimum and maximum values were used to scale the test datasets as well. All the experiments were conducted in the MATLAB environment.

4.2 Performance of Newton's method for the multilateration problem

The proposed initialization strategy for Newton's method performed well. Safeguarding was needed only in a few rare cases, which occurred for the AP, SRV, STC, BNK, CH, and CNS datasets. The highest average fold-wise failure rate was only 0.7%, which was observed for the SRV dataset with the RS-maximin method. Otherwise, the failure rate was less than 0.1%. All the reference points selection methods induced the failures. Overall, there were no differences between the reference points methods for the convergence speed of Newton's method. Typically, Newton's method converged after 2–4 iterations. Thus, the suggested approach for the multilateration problem provided a solver of linear complexity to realize the second phase of the MLM.

4.3 Results for optimal K

In Table 3, the median test RMSE and the best number of reference points $K/N(\%)$ are shown. The optimal number of reference points was selected based on the smallest mean validation RMSE. The symbol $**$ means that there is a statistically significant difference between test RMSEs based on the Kruskal-Wallis H test with a significance level of 0.05. Symbols $*$, †, ‡, §, and ‖ denote that a method has a statistically significantly smaller RMSE in pairwise comparison to the random, RS-K-means++, RS-K-medoids++, RS-UPGMA, and RS-maximin methods, respectively. In the pairwise comparisons, the significance level was also set to 0.05. The Kruskal-Wallis H test assumes equal variances for groups. Therefore, we tested the equality of the variances with the Brown-Forsythe test. Based on the Brown-Forsythe test, the variances related to the optimal K results are not equal for the BC dataset. Otherwise, they are equal. The best median test RMSE and the set of the smallest number of reference points (related to the mean value) are bolded for each dataset. Note that in Table 3 there are three optimal K values for each method because we used the 3-DOV-SCV approach in the experiments. Rounded Kruskal-Wallis scores are shown inside the brackets, and the best scores are bolded. The dataset-wise ranking of the methods is calculated from the raw Kruskal-Wallis scores. Based on these rankings, the total ranking of the methods is shown in the bottom of Table 3.

As shown in Table 3, the RS-K-medoids, RS-UPGMA, and RS-maximin methods performed equally well in the total ranking of the methods. The random and RS-K-means++ methods performed equally well in the total ranking of the methods. Based on the Kruskal-Wallis test, in general, there are no differences between the methods, and in terms of the best K, there is no clear difference between the methods. However, the best K varied from 2 to 100 among the datasets. We noticed from the fold-wise validation and test RMSE graphs that for most of the cases the best K corresponding to the smallest mean validation RMSE was not related to the best test RMSE. Moreover, the best K selection based on the smallest mean validation RMSE is dubious for some of the datasets, as in this way the complexity of the model is not taken into account. For example, for a large dataset, if increasing $K/N(\%)$ from 50 to 100 leads to only marginal improvement in the mean validation RMSE, then the model with a higher K and smaller mean validation RMSE is selected. For future work, there is still room for improvement in this direction.

4.4 Results for fixed K

In Tables 4–7, the test RMSEs are presented similarly as in Table 3, but with a fixed number of reference points. The variances in the error distributions are not equal for the SRV($K/N(\%) = 5, 10, 20$), CHA($K/N(\%) = 5, 10, 20$, STC($K/N(\%) = 5, 10, 20, 40$), and S1 ($K/N(\%) = 5, 10, 20, 40$) datasets, based on the Brown-Forsythe test of group variances. Therefore, the results given by the Kruskal-Wallis test are questionable for these cases; however, the ordering of the methods can be compared.

As expected, based on the total ranking, all the proposed methods have better RMSE than the random method when the number of reference points is small ($K/N(\%) = 5, 10$, Tables 4 and 5), where the RS-maximin method has the best RMSE. The RS-K-means++ and RS-K-medoids++ methods have very similar RMSEs for $K/N(\%) = 5, 10$. Thus, refinement of the reference points with K-means does not seem to be beneficial for the small K. In contrast to $K/N(\%) = 20, 40$, accuracy is improved with K-means refinement. For $K/N(\%) = 20, 40$, the RS-UPGMA is the best based on the total ranking. Therefore, running the whole clustering (not only initialization) seems to work better for higher K values.

Based on Tables 4–7, the performance of the random is the worst and that of the RS- maximin method is the best. A drawback of the RS-K-medoids++, RS-UPGMA, and RS-maximin method is that if the data contains anomalies, these methods are likely to select the anomalies as reference points.

This is probably why the random method outperforms the RS-UPGMA and RS-maximin methods for the CH dataset. The CH dataset is known to contain some large anomalies. Therefore, we combined a simple anomaly detection method (k-nearest neighbors) with the RS-UPGMA and tested it with the CH dataset. It was observed that anomaly detection improved the test error for the RS-UPGMA ($K/N(\%) = 5, 10, 20, 40$). Similar observations can also be drawn from the results for the S1 dataset. S1 was the cleanest dataset in the experiments: All input points were mapped to the output points with a sine-based function without any distortions. Based on Tables 4–7, the RS-UPGMA and RS-maximin methods have the largest error difference compared to the random method for the S1 dataset than for any other dataset. Therefore, a robust variant of the MLM combined with the RS-UPGMA or RS-maximin method, should be considered for regression tasks with anomalies.

4.5 Case S1: comparison of methods

To demonstrate the differences between the five methods, we ran only the reference point selection methods for S1 for 100 reference points (10%) and plotted the selected reference points (marked as blue pentagrams) which are shown in Figure 1. From Figure 1, one can notice that the proposed methods cover the data space better than the random method. Notably, the difference between the random and RS-maximin method is clear. The RS-maximin method selects a set of reference points that are sparsely spread to the input space with approximately evenly spaced. Contrary to the random approach, the selected reference points accumulate near the cluster centers and are near each other.

In Figure 2, the smallest 500 pairwise Euclidean distances for the selected 100 reference points for S1 are plotted in ascending order. The selected 100 reference points for each method are the same as in Figure 1. Figure 2 also illustrates the differences between the reference point methods. We see that, overall, the random is the worst method and RS-maximin the best method for identifying separate and input space well covered sets of reference points.

5 Conclusion

In this paper, we proposed four clustering-based methods for selecting the reference points for the MLM. We focused on experimenting with the proposed methods against the random approach in regression tasks with 13 datasets. An extensive experimental evaluation of the methods showed that the clustering-based methods can improve the performance of the MLM. A good set of reference points for the MLM covers the data space well. When an optimal number of reference points is desired, the RS-K-medoids++, RS-UPGMA, and RS-maximin methods are valid choices. With respect to the accuracy for fixed K, the RS-maximin method is the best choice for low K values ($K/N(\%) = 5, 10$). For higher K values ($K/N(\%) = 20, 40$) the RS-UPGMA is the best choice. However, the RS-maximin method is the most efficient proposed method, because the computational cost with respect to N is linear compared to the RS-UPGMA which has a quadratic time complexity with respect to N. For higher K values, this method has almost as good accuracy as the RS-UPGMA. Together with a linearly scaling Newton's method for the second step of the MLM, we obtain, as a whole, a computationally very efficient approach.

Although the maximin method is not recommended to be used for the K-means initialization based on the extensive study in [16], according to the present study, this method is valid for selecting the reference points in the MLM. This reflects that reference point selection has a slightly different aim than K-means initialization. For example, based on the experiments, the maximin method selects extreme points that are very valuable points, if they are not anomalies, for constructing the MLM model. In contrast, in terms of K-means initialization, these points are far from the cluster centers. Thus, these points are not optimal initial points. Furthermore, the proposed methods are less robust for outliers than the random approach. Therefore, integrating outlier detection into reference points selection or utilizing a robust approach for input and output distance matrix mapping should be considered for distorted datasets. In future work, it would be interesting to adapt and evaluate the proposed methods for classification tasks as well and to analyze how methods are affected if the number of reference points is different for the input and output space.

Acknowledgements

The work of TK has been supported by the Academy of Finland from the projects 311877 (Demo) and 315550 (HNP-AI).

References

[1] A. H. de Souza Junior, F. Corona, G. A. Barreto, Y. Miche, and A. Lendasse, "Minimal Learning Machine: A novel supervised distance-based approach for regression and classification," *Neurocomputing*, vol. 164, pp. 34–44, Sep 21 2015.

[2] G. Huang, Q. Zhu, and C. Siew, "Extreme learning machine: A new learning scheme of feedforward neural networks," in *2004 IEEE International Joint Conference on Neural Networks, Vols 1-4, Proceedings*, ser. IEEE International Joint Conference on Neural Networks (IJCNN), 2004, pp. 985–990.

[3] C. Chen, "A rapid supervised learning neural network for function interpolation and approximation," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1220–1230, SEP 1996.

[4] Y. Ren, P. Suganthan, N. Srikanth, and G. Amaratunga, "Random vector functional link network for short-term electricity load demand forecasting," *Information Sciences*, vol. 367368, pp. 1078 – 1093, 2016.

[5] D. P. Mesquita, L. S. Rocha, J. P. P. Gomes, and A. R. R. Neto, "Classification with reject option for software defect prediction," *Applied Soft Computing*, vol. 49, pp. 1085 – 1093, 2016.

[6] K. Lang, M. Zhang, and Y. Yuan, "Improved neural networks with random weights for short-term load forecasting," *PLOS ONE*, vol. 10, no. 12, pp. 1–14, 12 2015.

[7] W. Li, K. Chen, and D. Wang, "Industrial image classification using a randomized neural-net ensemble and feedback mechanism," *Neurocomputing*, vol. 173, Part 3, pp. 708 – 714, 2016.

[8] G. Feng, Z. Qian, and X. Zhang, "Evolutionary selection extreme learning machine optimization for regression," *Soft Computing*, vol. 16, no. 9, SI, pp. 1485–1491, SEP 2012.

[9] X. Xue, M. Yao, Z. Wu, and J. Yang, "Genetic ensemble of extreme learning machine," *Neurocomputing*, vol. 129, no. SI, pp. 175–184, APR 2014.

[10] A. K. Jain, “Data clustering: 50 years beyond k-means,” *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651 – 666, 2010, award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[11] R. R. Sokal, “A statistical method for evaluating systematic relationship,” *University of Kansas science bulletin*, vol. 28, pp. 1409–1438, 1958.

[12] S. Lloyd, “Least squares quantization in pcm,” *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[13] M. Ester, H.-P. Kriegel, J. Sander, X. Xu *et al.*, “A density-based algorithm for discovering clusters in large spatial databases with noise.” in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[14] T. Kohonen, “The self-organizing map,” *Neurocomputing*, vol. 21, no. 1-3, pp. 1–6, 1998.

[15] R. Xu and D. Wunsch, “Survey of clustering algorithms,” *IEEE Transactions on neural networks*, vol. 16, no. 3, pp. 645–678, 2005.

[16] M. E. Celebi, H. A. Kingravi, and P. A. Vela, “A comparative study of efficient initialization methods for the k-means clustering algorithm,” *Expert systems with applications*, vol. 40, no. 1, pp. 200–210, 2013.

[17] D. Arthur and S. Vassilvitskii, “k-means++: The advantages of careful seeding,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[18] D. P. P. Mesquita, J. P. P. Gomes, and A. H. de Souza Junior, “A minimal learning machine for datasets with missing values,” in *22nd International Conference on Neural Information Processing - ICONIP 2015*, 2015, pp. 565–572.

[19] D. P. Mesquita, J. P. P. Gomes, A. H. de Souza Junior, and J. S. Nobre, “Euclidean distance estimation in incomplete datasets,” *Neurocomputing*, vol. 248, pp. 11–18, 2017.

[20] J. P. P. Gomes, D. P. M. A. L. Freire, A. H. de Souza Junior, and T. Kärkkäinen, “A robust minimal learning machine based on the m-estimator,” in *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2017*, 2017, pp. 383–388.

[21] D. P. Mesquita, J. P. P. Gomes, and A. H. de Souza Junior, “Ensemble of efficient minimal learning machines for classification and regression,” *Neural Processing Letters*, vol. 46, no. 3, pp. 751–766, 2017.

[22] L. B. Marinho, A. H. de Souza Junior, and P. P. Rebouças Filho, “A new approach to human activity recognition using machine learning techniques,” in *International Conference on Intelligent Systems Design and Applications*. Springer, 2016, pp. 529–538.

[23] T. Kärkkäinen, “Extreme minimal learning machine,” in *26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2018*, 2018, pp. 237–242.

[24] A. C. de Oliveira, J. P. P. Gomes, A. R. R. Neto, and A. H. de Souza Junior, “Efficient minimal learning machines with reject option,” in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS)*, 2016, pp. 397–402.

[25] W. L. Caldas, J. P. P. Gomes, and D. P. Mesquita, “Fast Co-MLM: An efficient semi-supervised Co-training method based on the minimal learning machine,” *New Generation Computing*, vol. 36, no. 1, pp. 41–58, 2018.

[26] L. B. Marinho, J. S. Almeida, J. W. M. Souza, V. H. C. Albuquerque, and P. P. Rebouças Filho, “A novel mobile robot localization approach based on topological maps using classification with reject option in omnidirectional images,” *Expert Systems with Applications*, vol. 72, pp. 1–17, 2017.

[27] L. B. Marinho, P. P. Rebouças Filho, J. S. Almeida, J. W. M. Souza, A. H. de Souza Junior, and V. H. C. de Albuquerque, “A novel mobile robot localization approach based on classification with rejection option using computer vision,” *Computers & Electrical Engineering*, vol. 68, pp. 26–43, 2018.

[28] M. L. D. Dias, L. S. de Souza, A. R. da Rocha Neto, and A. H. de Souza Junior, "Opposite neighborhood: a new method to select reference points of minimal learning machines," in *26th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning - ESANN 2018*, 2018, pp. 201–206.

[29] A. H. de Souza Junior, F. Corona, Y. Miche, A. Lendasse, G. A. Barreto, and O. Simula, "Minimal learning machine: A new distance-based method for supervised learning," in *International Work-Conference on Artificial Neural Networks.* Springer, 2013, pp. 408–416.

[30] C. T. Kelley, *Solving nonlinear equations with Newton's method.* Siam, 2003, vol. 1.

[31] J. Hämäläinen, S. Jauhiainen, and T. Kärkkäinen, "Comparison of internal clustering validation indices for prototype-based clustering," *Algorithms*, vol. 10, no. 3, p. 105, 2017.

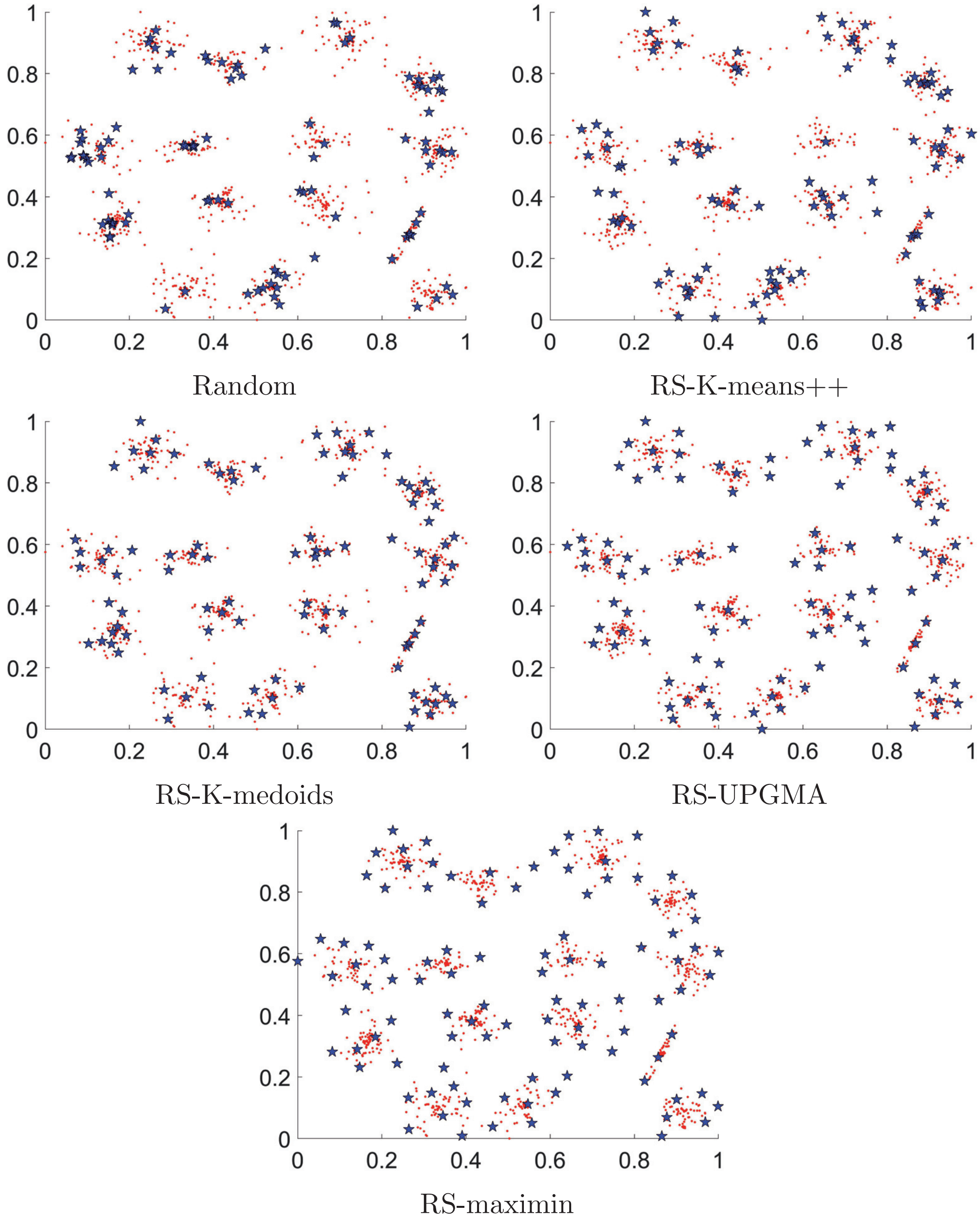
[32] T. F. Gonzalez, "Clustering to minimize the maximum intercluster distance," *Theoretical Computer Science*, vol. 38, pp. 293–306, 1985.

[33] I. Gronau and S. Moran, "Optimal implementations of upgma and other common clustering algorithms," *Information Processing Letters*, vol. 104, no. 6, pp. 205–210, 2007.

[34] J. Friedman, T. Hastie, and R. Tibshirani, *The elements of statistical learning.* Springer series in statistics New York, 2001, vol. 2.

[35] J. G. Moreno-Torres, J. A. Sáez, and F. Herrera, "Study on the impact of partition-induced dataset shift on k-fold cross-validation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 8, pp. 1304–1312, 2012.

Appendix A Figures



Random

RS-K-means++

RS-K-medoids

RS-UPGMA

RS-maximin

Figure 1: The 100 reference points selected for the S1 dataset.

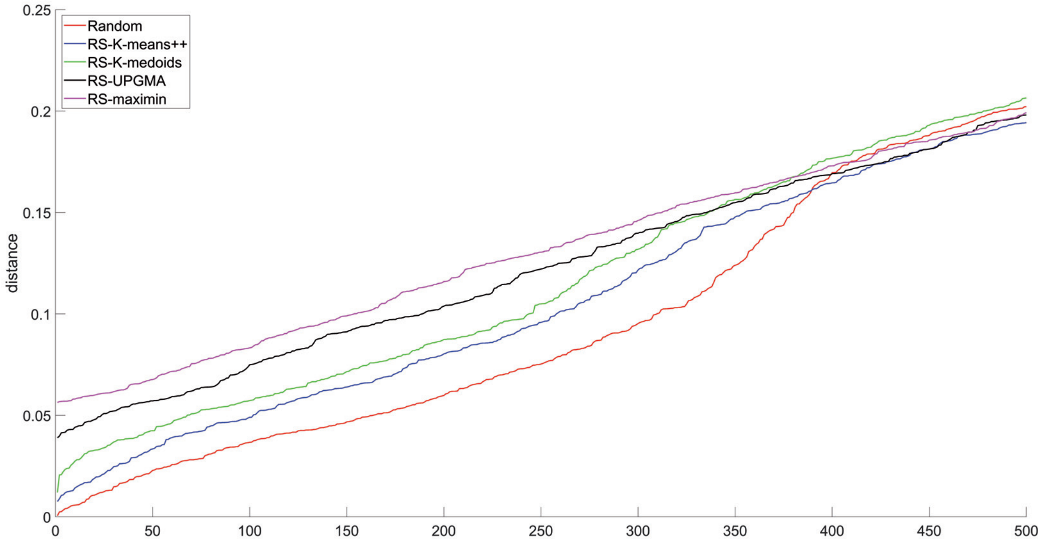


Figure 2: The smallest 500 pairwise Euclidean distances for the 100 reference points selected for the S1 dataset in ascending order. Clustering-based methods select a set of reference points that are more separeted from each other compared to the random approach.

Appendix B Tables

Table 3: RMSE for optimal K

	Random		**RS-K-means++**		**RS-K-medoids++**		**RS-UPGMA**		**RS-maximin**	
Dataset	RMSE	$K/N(\%)$	RMSE	$K/N(\%)$	RMSE	$K/N(\%)$	RMSE	$K/N(\%)$	RMSE	$K/N(\%)$
AP	0.0630 (**69**)	94, 93, 90	0.0645 (77)	66, 59, 69	0.0614 (70)	63, 79, 68	**0.0613** (71)	93, 93, 63	0.0691 (90)	**93, 36, 53**
SRV	0.0929 (80)	**75, 92, 70**	0.0913 (76)	67, 90, 88	0.0922 (76)	69, 95, 88	0.0921 (76)	78, 100, 82	**0.0878** (**70**)	92, 97, 100
BC	0.2684 (**68**)	19, 17, 9	0.2685 (71)	**11, 9, 8**	0.2726 (92)	7, 11, 31	**0.2682** (76)	8, 15, 14	0.2686 (71)	7, 15, 9
CHA**	0.0438 (78)	84, 52, 79	0.0502 (96)	18, 6, 24	**0.0392** (70)	25, 22, 32	0.0397† (**62**)	**8, 16, 20**	0.0433 (73)	37, 29, 44
BH	0.0734 (93)	**97, 62, 88**	0.0703 (74)	96, 80, 98	**0.0699** (71)	98, 99, 95	**0.0699** (**70**)	98, 99, 95	**0.0699** (**70**)	99, 99, 96
STC	0.0226 (76)	96, 100, 100	0.0226 (83)	**87, 91, 98**	**0.0225** (**72**)	94, 94, 100	**0.0225** (74)	94, 100, 98	**0.0225** (73)	95, 100, 99
S1	**0.0050** (**75**)	100, 100, 97	**0.0050** (76)	93, 99, 89	**0.0050** (**75**)	**89, 94, 84**	**0.0050** (76)	95, 99, 92	**0.0050** (76)	95, 99, 82
BNK	0.0439 (78)	91, 92, 90	0.0439 (77)	97, 77, 65	**0.0438** (75)	84, 86, 78	**0.0438** (**73**)	98, 44, 100	0.0439 (75)	**100, 37, 100**
ALR**	**0.0420** (65)	15, 13, 9	**0.0420** (70)	6, 11, 10	**0.0420**§ (61)	9, 10, 15	0.0423 (93)	6, 7, 12	0.0422 (89)	**4, 8, 4**
CA	0.0290 (80)	84, 88, 78	**0.0288** (**72**)	**54, 79, 64**	**0.0288** (73)	46, 79, 78	0.0289 (76)	100, 76, 60	0.0289 (75)	72, 96, 64
ELV	0.0554 (74)	3, 3, 2	0.0554 (73)	**2, 2, 2**	0.0555 (84)	4, 2, 3	**0.0553** (**70**)	3, 2, 3	0.0554 (77)	2, 2, 4
CH	0.1141 (82)	**56, 45, 66**	**0.1138** (76)	75, 69, 52	**0.1138** (78)	87, 72, 67	**0.1138** (**69**)	100, 100, 99	**0.1138** (73)	100, 97, 88
CNS	0.0605 (82)	18, 31, 16	0.0605 (81)	12, 18, 18	0.0603 (78)	10, 23, 18	**0.0598** (71)	**5, 15, 16**	0.0601 (**66**)	9, 19, 13
Rank	5(47)		4(45)		**1(34)**		**1(34)**		3(35)	

Table 4: RMSE for $K/N(\%) = 5$

Dataset	**Random**	**RS-K-means++**	**RS-K-medoids++**	**RS-UPGMA**	**RS-maximin**
AP**	0.1046(89)	0.1034(88)	0.1052(83)	0.0912(64)	**0.0819**(**53**)*†‡
SRV**	**0.1873**(**61**)$^{\Vert}$	0.1904(64)$^{\Vert}$	0.1968(81)	0.1942(73)	0.2035(100)
BC	0.2747(78)	0.2759(76)	0.2772(81)	0.2729(80)	**0.2684**(**61**)
CHA**	0.0710(101)	0.0621(81)	**0.0558**(65)*	**0.0558**(**56**)*	0.0600(74)
BH**	0.1096(67)	0.1089(68)	0.1182(91)	0.1128(86)	**0.1072**(**64**)
STC**	0.0497(123)	0.0457(82)*	0.0443(49)*†	**0.0437**(**45**)$^{*\dagger\Vert}$	0.0456(79)*
S1**	0.0310(120)	0.0288(97)	0.0279(86)*	0.0238(52)*†‡	**0.0199**(**21**)*†‡§
BNK**	0.0496(91)	0.0499(84)	0.0510(101)	**0.0465**(53)*†‡	**0.0465**(**50**)*†‡
ALR	**0.0420**(68)	0.0422(**67**)	0.0422(73)	0.0423(86)	0.0423(83)
CA**	0.0336(104)	0.0318(75)	0.0314(67)*	0.0319(77)	**0.0310**(**55**)*
ELV	0.0553(78)	0.0553(81)	**0.0552**(74)	**0.0552**(74)	0.0553(**71**)
CH**	**0.1196**(**44**)$^{\ddagger\S\Vert}$	0.1204(58)$^{\S\Vert}$	0.1214(76)	0.1224(101)	0.1227(98)
CNS	0.0616(91)	0.0614(80)	0.0611(79)	**0.0605**(67)	**0.0605**(**60**)*
Rank	5(47)	3(41)	4(43)	2(37)	**1(27)**

Table 5: RMSE for $K/N(\%) = 10$

Dataset	**Random**	**RS-K-means++**	**RS-K-medoids++**	**RS-UPGMA**	**RS-maximin**
AP**	0.0878(91)	0.0902(85)	0.0848(76)	0.0806(66)	**0.0739**(**60**)*
SRV**	0.1499(71)§	0.1457(68)§	0.1499(74)§	0.1589(108)	**0.1454**(**56**)§
BC	0.2683(81)	0.2692(80)	0.2691(81)	0.2676(80)	**0.2632**(**55**)
CHA**	0.0656(106)	0.0509(79)	0.0487(70)*	**0.0409**(**41**)$^{*\dagger\|}$	0.0521(82)
BH	**0.0979**(**67**)	0.1012(72)	0.1040(85)	0.1018(82)	0.1025(71)
STC**	0.0390(124)	0.0367(95)	0.0351(65)*	**0.0342**(**41**)*†	0.0352(52)*†
S1**	0.0206(126)	0.0139(90)*	0.0131(80)*	0.0111(60)*	**0.0077**(**22**)*†‡§
BNK**	0.0474(100)	0.0463(84)	0.0465(90)	**0.0448**(53)*†‡	**0.0448**(**50**)*†‡
ALR**	**0.0419**(**61**)$^{\|}$	0.0420(66)	0.0420(64)$^{\|}$	0.0424(90)	0.0425(97)
CA**	0.0316(106)	0.0303(77)	0.0300(67)*	0.0302(70)*	**0.0299**(**59**)*
ELV	0.0555(81)	0.0555(79)	0.0555(81)	0.0554(69)	**0.0553**(**67**)
CH**	**0.1173**(**44**)$^{\S\|}$	0.1179(58)$^{\S\|}$	0.1189(74)	0.1209(102)	0.1207(100)
CNS	0.0611(90)	0.0605(80)	0.0605(79)	**0.0598**(**64**)	0.0601(**64**)
Rank	5(51)	3(42)	3(42)	2(35)	**1(25)**

Table 6: RMSE for $K/N(\%) = 20$

Dataset	**Random**	**RS-K-means++**	**RS-K-medoids++**	**RS-UPGMA**	**RS-maximin**
AP	0.0840(90)	0.0795(82)	0.0799(78)	0.0742(**62**)	**0.0730**(65)
SRV	0.1180(69)	0.1186(70)	**0.1147**(**65**)	0.1223(84)	0.1221(89)
BC	**0.2663**(**73**)	0.2686(78)	0.2685(78)	0.2687(74)	0.2679(75)
CHA**	0.0545(102)	0.0438(79)	0.0454(71)*	**0.0376**(**46**)$^{*\dagger\|}$	0.0478(79)
BH**	0.0900(87)	0.0885(83)	0.0868(75)	**0.0844**(**53**)*	0.0881(79)
STC**	0.0319(131)	0.0301(105)	0.0289(73)*†	0.0280(39)*†‡	**0.0276**(**30**)*†‡
S1**	0.0112(130)	0.0084(91)*	0.0081(80)*	0.0068(48)*†‡	**0.0056**(**29**)*†‡
BNK**	0.0452(99)	0.0451(90)	0.0448(86)	0.0442(55)*†‡	**0.0440**(**48**)*†‡
ALR**	**0.0421**(**57**)$^{\S\|}$	0.0423(67)	0.0423(65)	0.0426(93)	0.0427(96)
CA**	0.0304(103)	0.0297(76)	**0.0293**(69)*	**0.0293**(**64**)*	**0.0293**(66)*
ELV	0.0561(82)	0.0561(84)	0.0561(83)	0.0558(67)	**0.0557**(**62**)
CH**	**0.1156**(**42**)$^{\S\|}$	0.1161(62)$^{\S\|}$	0.1165(67)$^{\S\|}$	0.1191(100)	0.1192(101)
CNS	0.0606(86)	0.0603(76)	0.0602(75)	**0.0599**(**68**)	0.0603(72)
Rank	4(48)	5(49)	3(36)	**1(27)**	2(35)

Table 7: RMSE for $K/N(\%) = 40$

Dataset	**Random**	**RS-K-means++**	**RS-K-medoids++**	**RS-UPGMA**	**RS-maximin**
AP**	0.0740(88)	0.0728(84)	0.0685(76)	**0.0658**(**55**)*	0.0691(74)
SRV	**0.0967**(**69**)	0.1014(78)	0.1000(75)	0.0997(72)	0.1038(83)
BC	0.2713(79)	0.2700(76)	0.2728(82)	**0.2694**(**67**)	0.2709(74)
CHA**	0.0499(101)	0.0427(72)	**0.0390**(**65**)*	0.0395(69)*	0.0414(70)*
BH**	0.0793(88)	0.0808(92)	0.0795(83)	**0.0747**(**55**)*†	0.0762(60)*†
STC**	0.0269(131)	0.0257(98)*	0.0252(73)*	0.0245(**37**)*†‡	**0.0244**(40)*†‡
S1**	0.0071(121)	0.0057(77)*	0.0060(75)*	0.0054(54)*	**0.0052**(**50**)*
BNK**	0.0445(94)	0.0442(79)	0.0441(75)	0.0441(73)	**0.0439**(**57**)*
ALR	**0.0426**(**60**)	0.0429(75)	0.0430(79)	0.0429(86)	0.0428(78)
CA**	0.0294(98)	**0.0289**(71)	**0.0289**(71)	**0.0289**(**68**)	**0.0289**(70)
ELV	0.0572(80)	0.0570(80)	0.0570(79)	**0.0569**(70)	**0.0569**(**68**)
CH**	**0.1144**(**54**)$^{\S\|}$	**0.1144**(60)$^{\S\|}$	0.1146(63)$^{\S\|}$	0.1168(101)	0.1166(100)
CNS	0.0606(80)	0.0604(**70**)	**0.0602**(**70**)	0.0606(79)	0.0607(78)
Rank	5(51)	4(46)	3(38)	**1(29)**	2(31)