Sparse Similarity and Network Navigability for Markov Clustering Enhancement

DISSERTATION

to achieve the academic degree

Doktoringenieur (Dr.-Ing.)

Technische Universität Dresden

Faculty of Computer Science

Presented by

Claudio Patricio Durán Cancino

Dipl. Bioinformatics Engineer

1st Supervisor: Prof. Dr. Michael Schroeder (Technische Universität Dresden)

2nd Supervisor: Prof. Dr.-Ing. Carlo Vittorio Cannistraci (Former: Technische Universität Dresden. Current: Tsinghua University)

Dresden, 2021

Ⓢ This work was principally funded by the Deutscher Akademischer Austauschdienst (DAAD) promotion program number 57299294.

ACKNOWLEDGEMENTS

First, I want to thank my supervisor Dr. Carlo Vittorio Cannistraci to trust me to become one of his pupils. Without his tremendous support during my entire period as a Ph.D. student, I could not have achieved such a fruitful Ph.D. His constants words of advice and suggestions, not only about work but also about life, have grown in me and helped me pay my way to a luminous future.

Many thanks to Prof. Dr. Michael Schroeder for his support and critical advice, which undoubtedly helped me improve. His wise words always opened a box in me, whose sides are made of intriguing questions.

I want to thank my colleagues and friends, who accompanied me during the Ph.D. period, Aldo, Alessandro, Alberto, and Ilyes. And also to all my friends around the world, who made this experience easier.

I want to thank my biophysics professor from UTALCA, Dr. Wendy Gonzalez, who encouraged me to apply for a DAAD scholarship, travel a few kilometers to another continent, and learn some German.

A big thank you to my parents, who have always encouraged me and believed in me. To my siblings, Ricardo, Javiera, Carolina, Amparo, and Gonzalo, and my whole family for always having my back.

And last but not least, my partner of life, Melissa Adasme. We embarked on this trip together, and during the whole process, she has helped and supported me. She will surely continue doing so in the projects ahead of us together with our small family conformed by her, our dog Becks and me.

PUBLICATIONS

Included in this dissertation

1. Nonlinear machine learning pattern recognition and bacteria-metabolite multilayer network analysis of perturbed gastric microbiome.
   Claudio Durán, Sara Ciucci, Alessandra Palladini, et al.
   Nature Communications (2021).
   Contributions: C. Durán is the main contributor to this publication. He realized the majority of the unsupervised data analysis, including the programming of the novel algorithms PSI and MC-MCL. He realized the figures and tables, and wrote part of the article.
   Thesis: Chapter 3 is based on this publication.
2. Geometrical inspired pre-weighting enhances Markov clustering community detection in complex networks.
   Claudio Durán, Alessandro Muscoloni & Carlo Vittorio Cannistraci.
   Applied Network Science (2021).
   Contributions: C. Durán is the main contributor to this publication. He realized the data analysis pipeline, the figures and tables, and wrote the majority of the article.
   Thesis: Chapters 6 and 9 are based on this publication.
3. Nonlinear Markov clustering by minimum curvilinear sparse similarity.
   Claudio Durán, Aldo Acevedo, Sara Ciucci, Alessandro Muscoloni & Carlo Vittorio Cannistraci.
   ArXiv (2019) (Under Review in *IEEE Access*).
   Contributions: C. Durán is the main contributor to this publication. He realized the experiment analysis, implemented the code the

presented algorithm, realized the figures and tables and wrote most of the article.
Thesis: Chapters 4, 7 and 10 are principally based on this publication.

Excluded from this dissertation (off-topic)

Peer reviewed

4. Use of steroid profiling combined with machine learning for identification and subtype classification in primary aldosteronism.
   Graeme Eisenhofer, Claudio Durán, Carlo Vittorio Cannistraci, et al.
   JAMA network open (2020).
   Contributions: C. Durán analyzed the dataset with a machine learning pipeline, realized part of the statistical analysis and the figures pertinent to the machine learning topic. He wrote part of the article.
5. Steroid metabolomics: machine learning and multidimensional diagnostics for adrenal cortical tumors, hyperplasias, and related disorders.
   Graeme Eisenhofer, Claudio Durán, Triantafyllos Chavakis, et al.
   Current Opinion in Endocrine and Metabolic Research (2019).
   Contributions: C. Durán realized the machine learning pipeline figure and wrote the machine learning section of the article.
6. Cell mechanics based computational classification of red blood cells via unsupervised machine intelligence applied to morpho-rheological markers.
   Yan Ge, Philipp Rosenddahl, Claudio Durán, et al.
   IEEE/ACM Transactions on Computational Biology and Bioinformatics (2019).
   Contributions: C. Durán implemented and realized the supervised machine learning section. He wrote the pertinent section in the article.

7. Pioneering topological methods for network-based drug–target prediction by exploiting a brain-network self-organization theory.
   Claudio Durán, Simone Daminelli, Josephine M. Thomas, et al.
   Briefings in bioinformatics (2018).
   Contributions: C. Durán is the main contributor to this publication. He realized the analyses of the LCP-based, matrix factorization and supervised algorithms for bipartite link prediction in all datasets. He realized the majority of the figures and tables, and wrote part of the article.
8. Lipidomics in major depressive disorder.
   Andreas Walther, Carlo Vittorio Cannistraci, Kai Simons, Claudio Durán, et al.
   Frontiers in psychiatry (2018).
   Contributions: C. Durán had intellectual discussions with the main author of the manuscript. He also worked and added comments on the first draft of the article.
9. Enlightening discriminative network functional modules behind Principal Component Analysis separation in differential-omic science studies.
   Sara Ciucci, Yan Ge, Claudio Durán, et al.
   Nature Scientific reports (2017).
   Contributions: C. Durán is first shared author with Yan Ge and Sara Ciucci. He implemented the majority of the automatic code released upon article publication both in MATLAB and R versions. He corrected and gave input for the article drafts.

Archive

10. Measuring group-separability in geometrical space for evaluation of pattern recognition and embedding algorithms.
    Aldo Acevedo, Sara Ciucci, Ming-Ju Kuo, , Claudio Durán & Carlo Vittorio Cannistraci.
    ArXiv (2019) (Under review in *IEEE Access*).

Contributions: C. Durán realized the basic MATLAB code from this new concept of separability. He helped with the data analysis and with the generation of the figures as well as for concept-related aspect of the article. He wrote part of the article and corrected the general draft.

11. LIPEA: lipid pathway enrichment analysis.
    Aldo Acevedo, <u>Claudio Durán</u>, Sara Ciucci, et al.
    BioRxiv (2018)
    Contributions: C. Durán implemented the source code of the algorithm. He helped in the implementation of the lipid mapping, gave suggestions for the LIPEA webpage and wrote part of the article.

ABSTRACT

Markov clustering (MCL) is an effective unsupervised pattern recognition algorithm for data clustering in high-dimensional feature space that simulates stochastic flows on a network of sample similarities to detect the structural organization of clusters in the data. However, it presents two main drawbacks: (1) its community detection performance in complex networks has been demonstrating results far from the state-of-the-art methods such as Infomap and Louvain, and (2) it has never been generalized to deal with data nonlinearity.

In this work both aspects, although closely related, are taken as separated issues and addressed as such.

Regarding the community detection, field under the network science ceiling, the crucial issue is to convert the unweighted network topology into a ‘smart enough’ pre-weighted connectivity that adequately steers the stochastic flow procedure behind Markov clustering. Here a conceptual innovation is introduced and discussed focusing on how to leverage network latent geometry notions in order to design similarity measures for pre-weighting the adjacency matrix used in Markov clustering community detection. The results demonstrate that the proposed strategy improves Markov clustering significantly, to the extent that it is often close to the performance of current state-of-the-art methods for community detection. These findings emerge considering both synthetic ‘realistic’ networks (with known ground-truth communities) and real networks (with community metadata), even when the real network connectivity is corrupted by noise artificially induced by missing or spurious links.

Regarding the nonlinearity aspect, the development of algorithms for unsupervised pattern recognition by nonlinear clustering is a notable problem in data science. Minimum Curvilinearity (MC) is a principle that approximates nonlinear sample distances in the high-dimensional feature space by curvilinear distances, which are computed as transversal paths over their minimum spanning tree, and then stored in a kernel. Here, a nonlinear MCL algorithm termed MC-MCL is proposed, which is the first nonlinear kernel extension of MCL and exploits Minimum Curvilinearity to enhance the performance of MCL in real and synthetic high-dimensional data with underlying nonlinear patterns. Furthermore, improvements in the design of the so-called MC-kernel by applying base modifications to better approximate the data hidden geometry have been evaluated with positive outcomes. Thus, different nonlinear MCL versions are compared with baseline and state-of-art clustering methods, including DBSCAN, K-means, affinity propagation, density peaks, and deep-clustering. As result, the design of a suitable nonlinear kernel provides a valuable framework to estimate nonlinear distances when its kernel is applied in combination with MCL. Indeed, nonlinear-MCL variants overcome classical MCL and even state-of-art clustering algorithms in different nonlinear datasets.

This dissertation discusses the enhancements and the generalized understanding of how network geometry plays a fundamental role in designing algorithms based on network navigability.

CONTENTS

Part I. INTRODUCTION

This first part will deliver glances about concepts and algorithms needed to understand the work realized starting from Part II. It will clarify concepts such as clustering, community detection, and it will present algorithms related to such strategies from a qualitative and mathematical perspective. Some notions about their advantages and limits will be discussed to finalize with the motivation to realize the here presented work.

1. Clustering

Clustering can be seen as one of the oldest strategies to understand and to interpret pattern formation in our world: indeed, in daily life, people express their intelligence also in the action to group objects, items, or even time-series events in relation to the similarity or dissimilarity between their features [1]. Specifically, the term Clustering refers to an unsupervised pattern recognition methodology which, given an ensemble of objects or data, aims to recognize their organization in groups and subgroups starting from their features, such as the three groups detected in Figure 1. Nowadays, in artificial intelligence, clustering is defined as the automatic and unsupervised identification of groups of observations that are similar to one another and different from other groups in a dataset [2]. Indeed, clustering aims mainly to identify distributions and patterns in the underlying data, generating a partitioning of a given dataset into different groups called clusters [3]. In this sense, the patterns of the observations that are grouped in the same cluster should be similar (in the feature space) to each other, while patterns of observations that result in different clusters should not.

*Figure 1. **Basic clustering example.** Features are given by shapes and colours.*

1.1. The era of big data

Nowadays, in the era of Big Data, there is a tremendous amount of high-dimensional data available due to the progress in storage procedures, and the ubiquitous growth and exploitation of technologies that generate high-dimensional datasets; trends that will persist in the next decades [4]. Reason that evidence the successful employment of clustering algorithms in diverse areas of applications which comprises, but are not limited to, image processing [5]–[7], pattern recognition [8]–[10], market research [11], [12], etc. However, in fields such as systems biology and molecular medicine, the realization of controlled experiments that can provide observations or samples to investigate a scientific hypothesis can be very time-consuming (recruitment of patients, lab experiments etc.) and also expensive [13]. For such a reason, in these fields, pilot studies that generate a few samples, in order to test the validity of a scientific hypothesis before making the decision to scale to the big numbers, is a frequent practice [13]. Therefore, it is important under this context to account for diverse algorithms which focus to tackle certain aspects of a data problem.

1.2. Types of clustering

Many clustering strategies have been developed to deal with specific obstacles that might arise in data, that can be related to their shape (concave vs non- concave), dimensionality, denseness, between cluster interaction (linear vs nonlinear), etc. and each of them can be encapsulated in a clustering category [14]. Although many categories might exist, special attention will be drawn here towards five of them: partitioning, hierarchical, density-based, graph-based, and deep-based methods. Note that some categories might enclose another.

1.2.1. Partitioning methods

Clustering methods based on partitioning are characterized by grouping the data samples into *k* groups or partitions in the space. The *k* groups are usually specified a priori by the user. The quality of the partitions is iteratively improved by a specific objective function that the algorithm attempts to maximize or minimize, depending on the clustering method.

K-means

K-means – introduced as an idea by Steinhaus [15] in 1956 and termed K-means by MacQueen [16] in 1967 – is a well-known and one of the oldest data clustering algorithms still widely used due to its simplicity and effectiveness. K-means' strategy to find clusters consist of splitting the data into a set of *k* desired clusters, defined a priori by the user as aforementioned. It starts with an initial random partition of the data, to use consequently an iterative control strategy to optimize the objective function *J:* average squared Euclidean distance (1).

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| x_i^{(j)} - C_j \right\|^2 \tag{1}$$

Each cluster is represented by the gravity center of cluster *C*. In other words, it determines *k* representatives (centers) by minimizing the objective function *J*, then it assigns each sample *x* to the cluster with its closest representative (Figure 2). A major restriction is that the shapes of the clusters found by this algorithm are convex (linear data).

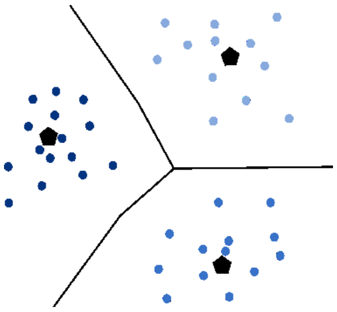
Figure 2. ***Partitioning of three clusters K-means example.*** *The lines denote the partitions whilst the pentagons the respective cluster's*

1.2.2. Hierarchical methods

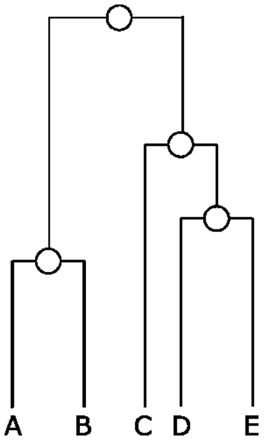

Figure 3. ***Dendrogram example in hierarchical clustering.***

Different from partitioning methods, hierarchical methods do not need as input the *k* number of clusters, but it is rather inferred from a dendrogram (Figure 3), a tree-based graphical representation of clusters, which can be constructed from a distance matrix. The *k* can be determined depending on the 'cut' applied to the dendrogram. Generally, there are two types of approaches for tree realization: Agglomerative (bottom-up) and divisive (top-down) [17]. The agglomerative method starts from the leaves (each single data sample) to join into couples the closest samples together. Subsequently continues to join close groups until it arrives at the root of the dendrogram. On the contrary, divisive methods start from the root containing the whole universe of data samples to split it into two nodes afterwards. This process is repeated for each new node until the single data samples (or leaves) are reached.

Single-linkage

An example of an agglomerative hierarchical algorithm is the method single-linkage [18], [19]. Hierarchical methods are one of the oldest clustering approaches. Particularly, Single linkage that was already presented in the 1950s [19]. Single linkage consists of iteratively joining leaves (in the first iteration) or clusters (in the subsequent iterations) with the smallest minimum pairwise distance. The output of this clustering also corresponds to an approximate and weighted minimum spanning tree (MST; for more details please refer to the section 1.2.4 Graph-based methods). Between the drawbacks of this method, the misinterpretation of the dendrogram and closeness misrepresentation of points in clusters are common [20].

1.2.3. Density-based methods

Similar to the hierarchical methods, density-based algorithms infer the number of clusters directly from the data. The rationale behind it states that clusters are formed by data samples in a contiguous region of high density and separated from the rest of the clusters by regions of lesser density. Usually, samples in lower density regions are considered as noise [21].

Density based spatial clustering of applications with noise

Density based spatial clustering of applications with noise (DBSCAN) - introduced by Ester et al. in 1996 [22]- is one of the most successful density-based clustering algorithms. It is a method that requires two density input parameters: MinPts and Eps. If selected any point j in the space, MinPts is the minimum number of points inside a neighbourhood (of the selected point j) defined as a circle of radius Eps. DBSCAN defines as *core points* all the points that have at least MinPts points (including itself) in their Eps neighbourhood. If a point is reachable by a *core point* but does not satisfy the MinPts in its Eps neighbourhood, it is called a *border point*. The main idea behind this algorithm is that a group of points that are mutually reachable through core points (because they are included in the neighbourhood of radius Eps of core points) forms a cluster. All points not reachable from *core points* and that do not satisfy the MinPts and Eps parameters are *outliers* or *noise points* (Figure 4).

As commented above, this algorithm does not need to input the desired number of clusters. Instead, it finds them automatically according to the tuning of the two above mentioned parameters. Nevertheless, the finding of these correct parameters MinPts and Eps is a nontrivial problem [23]. Moreover, in datasets with varying densities, DBSCAN can phase problems to detect meaningful clusters [24].

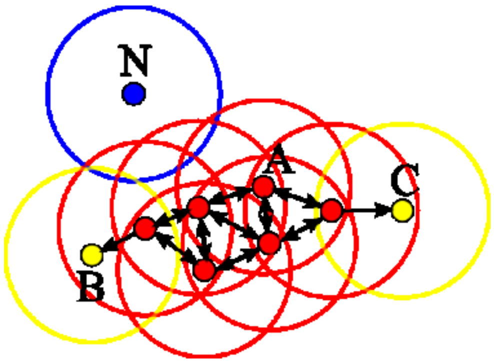

Figure 4. ***DBSCAN illustration.*** *A represent core points, B and C represent border points and N represent an outlier or noise. By Chire - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=17045963.*

Density peaks

Density peaks is a relatively recent algorithm proposed by Rodriguez et al. [25]. The algorithm has its basis in the assumptions that cluster centers (a.k.a. density peaks) are relatively far away from other cluster centers, and that they are generally surrounded by points with lower local density (Figure 5). With this in notion, the primordial step in this method is to select the density peaks. This is accomplished by computing two values: (a) the local density ρ and (b) the distance to higher density points δ. ρ is calculated for each data point by counting the elements of its surrounding area given a certain threshold (Figure 5A), and δ is the minimum distance of a point to any other element of higher density. These two values are latterly employed to construct a density peak decision graph (Figure 5B) used, as the name suggests, to select the cluster centers or density peaks. Finally, points are assigned to the clusters from the closest density peak. An exception can be evidence in outliers, which present in the decision graph a relatively high δ but a low ρ.

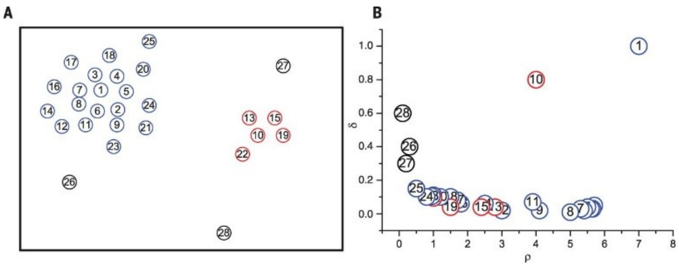

Figure 5. ***Density peaks exemplification.*** *(A) 2D data representation. Points are labeled according to their densities (ρ) in ascending order. (B) Density peaks decision graph. Points 1 and 10 are selected as the density peaks (centers) of two different clusters. Points 26 27 and 28 are designated as outliers or noise. Image extracted from* [25].

Shortest path-based density peaks

An improvement to this technique was provided in 2019 by Pizzagalli and colleagues [26]. In this study, the authors argue that one main drawback of the density peaks algorithm is the strategy to designate the data points to each cluster after selecting the centers. The fact that the cluster assignation of all data points is given by their closeness to a density peak in the Euclidean space (Figure 6A) neglects one of the principles of density based clustering, which is to uncover successfully clusters of arbitrary shapes (as in the case of DBSCAN). In this circumstance, if another large non-globular cluster is close to another globular one, the classical density peaks may assign points of the non-globular cluster to its neighbor cluster (Figure 6B). Now, Pizzagalli and colleagues introduced a shortest path strategy for the point-class member assignation that follows a minimax path-cost function (Figure 6C). This action is translated as following paths with more points but tinier gaps between points, instead of fewer points and bigger gaps. Such an approach conserves the density property by correctly detecting clusters of different shapes (Figure 6D).

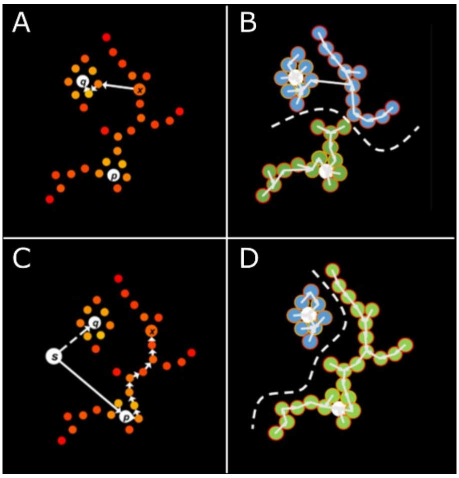

Figure 6. ***Classical density peaks vs shortest path density peaks.*** *(A) Classical cluster assignation through the Euclidean space. (B) Clustering result of the classical approach. (C) Cluster assignation through the shortest path. (D) Clustering result of shortest path approach. Image extracted from* [26].

1.2.4. Graph-based methods

This type of algorithms represent data into graphs, where nodes correspond to the data samples and the edges between nodes correspond to their relation measured as similarities or distances in the feature space. The clusters are thus obtained by selecting strongly interconnected sub-graphs respective to a given criterion. The advantage of these type of methods relies on the ability to detect clusters of diverse shapes and sizes (when clearly separated).

Affinity Propagation

AP – introduced by Frey et al. in 2007 [27] – is a clustering algorithm based on a message-passing procedure that takes as input a similarity value (in general codified as negative distance/dissimilarity values) between pairs of data points or samples. The messages are propagated between data points until a high-quality set of exemplars (data points selected as centers of clusters) and corresponding clusters gradually appear [27]. The AP algorithm does not take as input the predefined number of clusters, but requires for each data sample a real number termed "preference". Samples with larger preferences are more likely to be chosen as exemplars to form a cluster. The values of the input preferences influence the number of identified exemplars (lastly the number of clusters), but also emerges from the message passing procedure [27]. If a priori, all samples are equally suitable as exemplars, the preferences should be set to a common value. This value can be varied to produce different numbers of clusters. The shared preference value could be the maximum of the input similarities (resulting in a large number of clusters) or their minimum (resulting in a small number of clusters). One drawback of this method, and as it naturally may come to mind, is the need to specify the preference values a-priori, which is translated on having a bit of prior knowledge to select probable exemplars [28]. The algorithm also may fail to converge into the respective clusters. However, this can be avoided by setting a maximum number of iterations for message passing if convergence is never reached.

MST-based clustering

Other graph-based clustering methods are based on the minimum spanning tree (MST). An acyclic subgraph that contains all graph vertices is known as a spanning tree. If the edges of a graph are weighted, an MST is the spanning tree with minimum edge-weights (Figure 7). MST-based clustering strategy focuses on finding clusters by removing "inconsistent edges" from the MST followed by an "inconsistent measure" [29]. An example is the Euclidean MST (EMST) method [30] that, based on an MST whose edges are weighted by Euclidean distances, detects clusters by minimizing intra-cluster distance and maximizing inter-cluster distance. MST-based clustering approaches have emerged from EMST, such as standard EMST (SEMST), the Zahn's EMST (ZEMST) and the maximum standard deviation reduction (MSDR), whose difference relies on the inconsistency measure used to remove edges in the constructed MST. Whilst the SEMST strategy sorts the edges according to weights and starts to remove them by the highest weighted edge, the ZEMST strategy is based in Zahn's inconsistency measure[1] [30], and the MSDR tries to find a local minimum of the standard deviation reduction function [29].

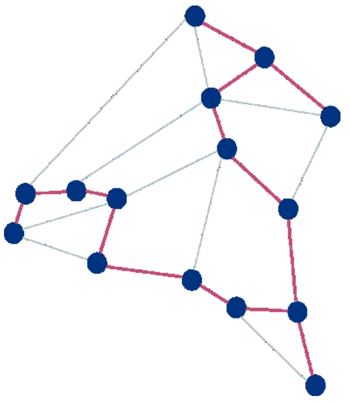

Figure 7. ***MST subgraph representation.*** *The MST is highlighted in purple.*

[1] The Zahn's inconsistency measure consists in deleting edges of an MST, whose weights are significantly larger than the average weights of nearby edges.

Cluster center initialization MST

A recent MST-based method termed cluster center initialization MST (cciMST) was proposed with the aim to improve the definition of inconsistency edges using a strategy similar to the classical density peaks with certain adjusted to be able to find clusters of arbitrary shapes [31]. In this study, the authors employed geodesic distances and dual densities in order to initialize the cluster' centers. Then, the inconsistent edges are removed by means of a shortest path strategy [31].

One of the drawbacks of MST-based clustering methods is the possibility to create "islands" when removing inconsistent edges. Special care should be taken for datasets of clusters with different densities.

Markov clustering

Finally, the last here presented graph-based clustering algorithm is termed Markov Clustering (MCL). MCL [32] – introduced by Stijn van Dongen in 2000 - is an algorithm for data clustering based on simulations of stochastic flows (random walks) in networks. A random walk is defined as a mathematical procedure that describes a succession of random steps through a mathematical space, consisting in this case of a network. The possible paths to 'walk' through are its edges, which are weighted with certain probabilities to pass through them. MCL works with an iterative process by alternating two

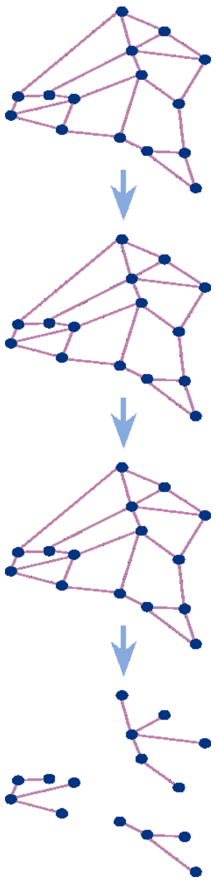

Figure 8. ***MCL workflow***. *Edges weights intra-clusters are enhanced while inter-clusters are weaken. The weights relate with the random walks probabilities for network navigation.*

operators called expansion and inflation. The expansion operator corresponds to the computation of random walks of higher length (many steps), which associates new probabilities between each pair of nodes. In practice, the expansion serves to associate higher probabilities to paths within clusters rather than in between clusters, because in general, there are more ways to go from one node to another in the same cluster. In contrast, the inflation operator has the effect of boosting intra-cluster walk probabilities and lowering inter-cluster walks. In practice, the inflation is the MCL parameter that serves to detect clustering patterns on different scales of granularity.

For clustering samples of a multidimensional dataset, the workflow starts with the computation of similarities (generally Pearson correlations) between the samples, by creating an edge between each pair, where the edge-weight assumes the value of the respective pairwise sample similarity. This produces the weighted similarity network upon which to simulate stochastic flows and detect the structural organization of clusters in the data.

Limitations of this technique discussed by the same creator include its difficulties in clustering tree graphs in sparse networks where the cardinality (or number) of the edges is close to the number of nodes and/or in graphs with a large diameter of natural clusters [32].

1.2.5. Deep-based methods

Deep clustering with sample-assignment invariance prior

With the popularization of deep learning, different fields have taken advantages of such algorithms for their successful usage in data analysis.

Between the many data types that can be analyzed with it, linguistics with natural language processing, numeric with Multilayer perceptron and Images with Convolutional Neural Networks, are of popular choice. Following the huge increase in deep learning methods, some algorithms were generated not only for supervised or reinforcement learning but also for unsupervised analysis. The autoencoder is a type

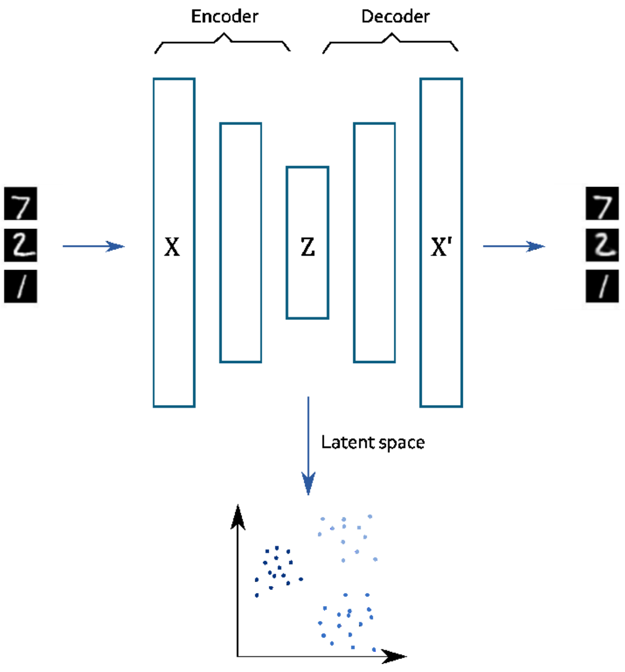

Figure 9. ***Deep clustering Illustration***. *The autoencoder, composed by an encoder and a decoder, is trained on images X for their reconstructions in X'. The bottleneck represented as Z is the latent space created by the encoder, which can be exploited by clustering algorithms.*

of artificial neural network exploited in clustering analyses due to its ability to data-reconstruction. It is able to transform the data feature space into a latent space thanks to an encoder, which is then transformed back as close as possible to the original space thanks to a decoder [33]. After the autoencoder network is trained, the latent space, output of the encoder, serves as input for clustering algorithms, such as K-means. The workflow idea is illustrated in **Error! Reference ource not found.**.

Recently, different methods from the same family were proposed considering the aforementioned autoencoder principle [34]. Following the workflow of Peng et al. algorithm, a denoising autoencoder is firstly pre-trained on noisy data with the aim to reconstruct the same data without noise. After pre-training, the algorithm evolves and replace the decoder side with another module. This new module applies clustering directly to the latent space created from the encoder. The module takes this new space and generates different probabilities distributions (clustering memberships) according to different distance measures and with respect to certain clusters centers. Then, a function (in this case, a Kullback-Leibler (KL) divergence loss) tries to minimize the distances between the probabilities distribution. In other words, clustering is being applied to the latent space using different distance measures and the same cluster centers (obtained with K-means). Each of these clustering with a certain distance will give a cluster memberships to each data point. The minimization function will try to generate an agreement as close as possible with respect to the clusters assignments. The result of this function minimization results in the so called sample-assignment invariance. In this context, the network is trained to learn better representations on the data (encoder) and at the same time improve the assignment of cluster memberships to each data point. In Figure 10 appears the representation of the network structured used by Peng et al.

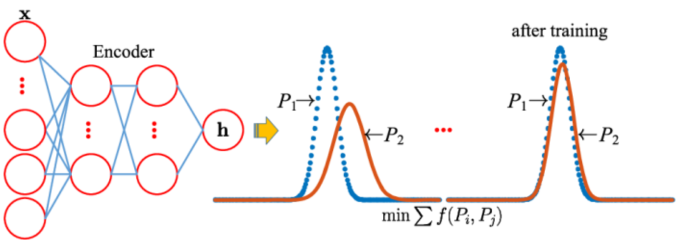

Figure 10. ***Network structure illustration used in Peng et al****. The first module (left side) consists of an encoder. The second module (right side) considers the latent space h generated by the encoder and creates different clustering memberships assignments, also called probabilities distribution (P1 and P2). With a minimization function, it aims to match both as close as possible until convergence is reached. Image extracted from* [34].

Here, x represents a data point (one image), *h* the latent space representation of the encoder, P_1 and P_2 the clustering membership distributions to be matched as close as possible by the KL divergence loss function. The deep-clustering algorithm names are HOMO, HOE, HIT, and HOT. Their difference relies on the distance to utilize being Euclidean distance, Cosine distance, city-block distance, and Pearson correlation-based distance, respectively.

2. Community detection

Introduced in network science, community detection is synonymous of clustering and refers to the well-known task in which complex networks are partitioned into communities. Like clusters, a community is an ensemble of nodes that are more likely to be interconnected between each other rather than to out-connect to other groups of nodes [35]. Why is this important? The reason is that many real-world systems are expressed as a network in different domains, like protein-protein interactions in biology, communication networks like airport interactions, social networks like Facebook, etc. Many of these networks have inherent and hidden communities, information that can be exploited depending on the interests of the analyst. Moreover, the detection of communities has grown into an essential and highly pertinent problem in network science with several applications. First, it allows unveiling the existence of a non-trivial internal network organization at a higher level, which permits us to infer special relationships between nodes that might not be able to emerge from direct empirical tests easily. Second, it helps to perceive better properties of processes occurring in a network [36]. A didactic example pertinent to our times is the spreading processes of a pandemic disease highly affected by the community structure of the graph. There is a logic why states around the world tried to break direct contact communities into smaller pieces during the years 2020 and 2021.

Naturally, graph-based clustering methods, such as MCL, also work as community detection algorithms. However, in this study, they are going to be seen as separate problems, where in clustering, we refer to the ability to analyze multidimensional datasets. In contrast, in community detection, we are going to concentrate on real and synthetic networks.

There are algorithms tailored specifically for this purpose. Two of the most successful methods are Infomap [37] and Louvain [38]. They have demonstrated high performances in synthetic benchmarks and small- and large-size real networks [39]–[42].

2.1. Infomap

The Infomap algorithm [37] finds the community structure by minimizing the expected description length (MDL) of a random walk trajectory using the Huffman coding process [43], [44]. This coding process is used to assign codewords to nodes to describe them in relation to a random walk path (Figure 11A). It works by assigning short codewords to common events (regular paths in this case) and long codes to uncommon ones. In other words, the Huffman coding process assigns a code to a node derived from the node visit frequency in relation to the random walk. After complete network encoding, a description specified in bits can be computed (Figure 11B).

Furthermore, when a random walk enters a community, it tends to stay inside the community for a long time. Using prefix codes with the Huffman coding process, one can determine certain regions (community) in the network and then use a unique code for each node inside the community (Figure 11C). Note that nodes from different communities can be assigned the same Huffman codeword. Applying this modality, a new description computation (in bits) can be calculated. Consequently, the algorithm aims to optimize the MDL. Infomap uses the hierarchical map equation [37], further development of the map equation, to detect community structures on more than one level. The hierarchical map equation indicates the theoretical limit of how concisely a network path can be specified using a given partition structure. In order to calculate the optimal partition (community) structure, this limit can be computed for different partitions, and the community annotation that gives the shortest path length is chosen.

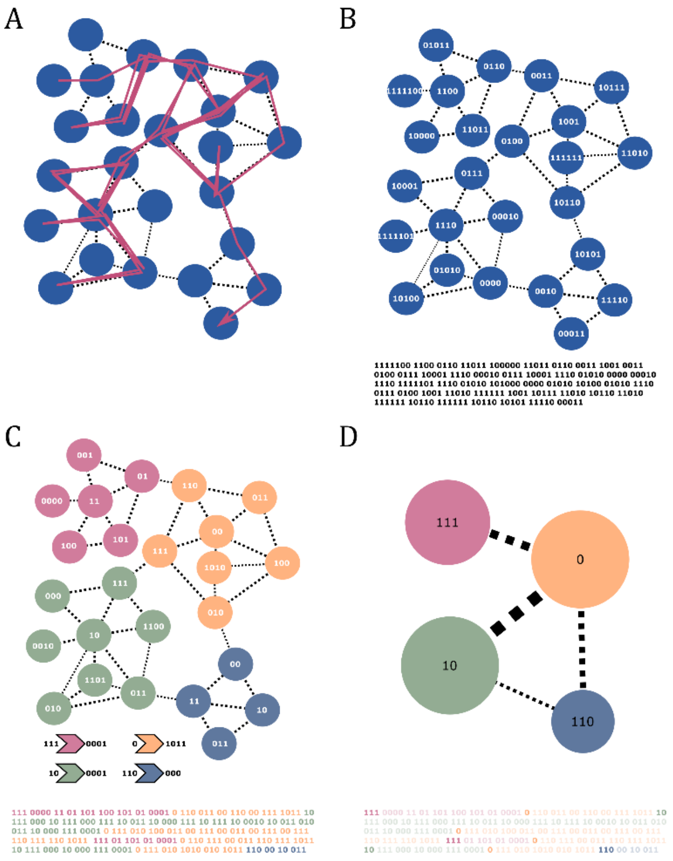

Figure 11. ***Detecting communities by compressing the description of information flows on networks.*** *(*A*) Description of a random walk trajectory on the network. The magenta line shows one sample trajectory. (*B*) The idea is to assign a code to each single node such that important structures have unique names. The Huffman code illustrated here is an efficient way to do so. The 314 bits under the network describe the sample trajectory in* A*, starting with 1111100 for the first node on the walk in the upper left corner, and ending with 00011 for the last node on the walk in the lower right corner. (*C*) A two-level description of the random walk, in which major clusters receive unique names, but the names of nodes within clusters are reused, yields on average a 32% shorter description for this network. The codes naming the modules and the codes used to indicate an exit from each module are shown to the left and the right of the arrows under the network, respectively. Using this code, we can describe the walk in* A *by only 243 bits, shown under the network in* C*. The first three bits 111 indicate that the walk begins in the red module, the code 0000 specifies the first node on the walk, etc. (*D*) Reporting only the module names, and not the locations within the modules, provides an efficient coarse graining of the network [44].*

2.2. Louvain

The Louvain algorithm [38] is separated into two phases, which are repeated iteratively. At first, every node in the (weighted) network represents a community in itself. In the first phase, for each node *i*, it considers its neighbours *j* and evaluates the gain in modularity[2] that would take place by removing *i* from its community and placing it in the community of *j*. The node *i* is then placed in the community *j* for which this gain is maximum, but only if the gain is positive. If no gain is possible node *i* stays in its original community. This process is applied until no further improvement can be achieved. In the second phase, the algorithm builds a new network whose nodes are the communities found in the first phase, whereas the weights of the links between the new nodes are given by the sum of the weight of the links between nodes in the corresponding two communities. For unweighted networks, the weights between new nodes translate into the number of links from one community to another. Links between nodes of the same previous community structure lead to self-loops of weight $2n$, where n is the weighted sum of the links or number of links inside the original community for unweighted graphs (Figure 12). Once the new network has been built, the two-phase process is iterated until there are no more changes and maximum modularity has been obtained. The number of iterations determines the height of the hierarchy of communities detected by the algorithm.

[2] Modularity is a measure related to network structure that quantifies the strength of the division of a graph into modules (a.k.a. communities) in relation with the number of edges inside and outside the community [93].

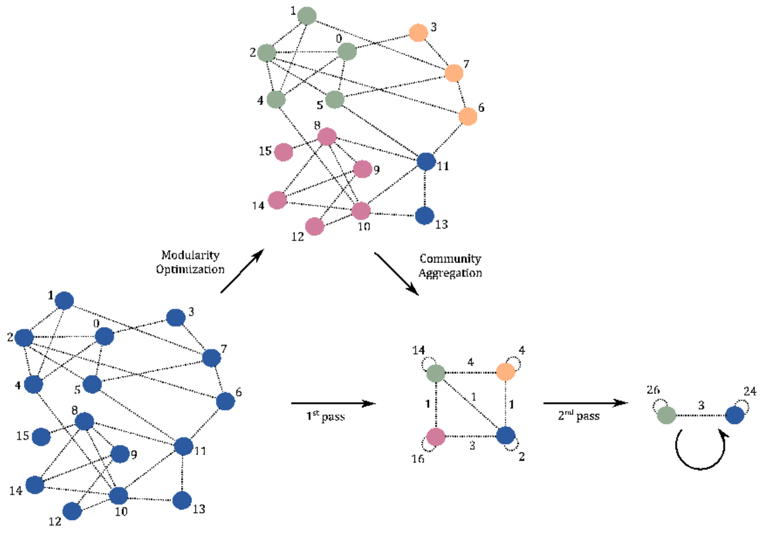

Figure 12. ***Visualization of the steps of Louvain's algorithm.*** *Each pass is made of two phases: (1) the modularity is optimized by allowing only local changes of communities; (2) the communities found are aggregated in order to build a new network of communities. The passes are repeated iteratively until no increase of modularity is possible.*

3. Motivation

Up to this point, a clarification on concepts like clustering, community detection, and different strategies to address such problems have been presented. However, no notion about the title of this work has been mentioned. This chapter will explain the motivation for the here proposed novel algorithms and the rationale behind them.

This journey starts with a biological dataset about gastric mucosa microbiomes from patients suffering from dyspepsia (please refer to chapter 10.1. 'High dimensional dataset description' for more details). This type of biological datasets are regularly analyzed by linear algorithms, followed by conclusions and statements drawn from

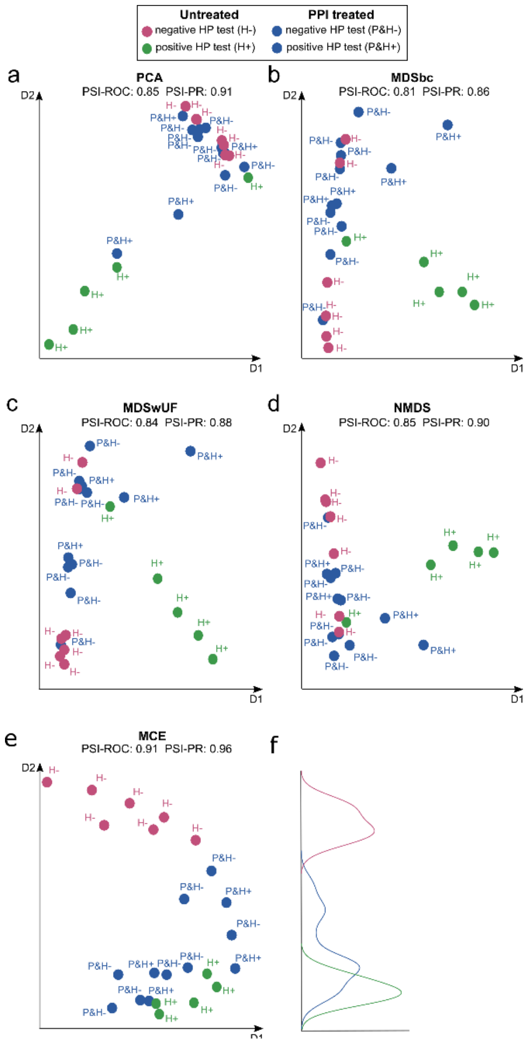

Figure 13. ***Dimension reduction techniques applied to the gastric mucosa dataset.*** *The plots represent the best dimension reduction results based on PSI-PR and PSI-ROC projection-based separability indices (PSI) for the three different labels (P-treated, untreated H+ and untreated H-), evaluated in the 2D embedding space. Dimensionality reductions applied: (a) PCA; (b) MDS with Bray-Curtis dissimilarity (MDSbc); (c) MDS with weighted UniFrac distance (MDSwUF); (d) non-metric MDS with Sammon Mapping (NMDS); (e) MCE. Blue dots represent PPI-treated samples, while magenta and green dots are the untreated samples which resulted either negative (magenta) or positive (green) to the H. pylori test. (f) The curves in three different colours (magenta, blue and green) highlight the different distributions of the three groups on the second dimension for the MCE plot (e) [51].*

their results and published in scientific journals. With the aim to corroborate such conclusions, the dataset was analyzed by means of different dimensionality reduction techniques, linear and nonlinear based, and their results were compared with a separability measured termed projection separability index (PSI).

Dimensionality reduction refers to unsupervised learning algorithms whose aim is to decrease the number of dimensions of a dataset into a meaningful lower space, ideally with the intention to represent as close as possible the intrinsic dimensionality[3] [45]. Two famous and widely employed linear dimensionality reduction techniques are called principal component analysis (PCA) [46], [47] (Figure 13A) and Multidimensional scaling (MDS) [48]–[50] (Figure 13B-D).

Thus, the different algorithm embedding results from linear (PCA, MDS) and nonlinear algorithms (MCE) were compared (Figure 13). Evidently, the nonlinear algorithm MCE could detect a pattern not visible with the linear techniques and could segregate PPI naïve patients without *H. pylori* (H-) infection from the patients with PPI intake (P) along the second dimension of embedding (Figure 13E, F). These results can be translated into very different conclusions compared with the results of the linear algorithm versions and therefore, the need of more investigations in this directions was considered. Important to note, is that the MCE algorithm is in reality a nonlinear version from the PCA algorithm, whose principle is based in the computation of an MST-based kernel termed minimum curvilinearity (MC) (Please refer to the chapter 7.1. 'Minimum curvilinearity' for more details).

As next, the following question arises: can these results be demonstrated as well in clustering analyses? To this aim, the efforts

[3] The intrinsic dimensionality of data is the minimum number of parameters needed to account for the observed properties of the data [94].

were focused on the MCL clustering algorithm due to its success in the field. Moreover, since it is known that the functionality of MCL is based on stochastic flows (random walks) through a network, it is here hypothesized that a well-defined network that approximates the hidden network geometry as input to MCL can boost its performance thanks to the principles behind network geometry and network navigability. This was proved in an article analyzing the nonlinear pattern of the gut microbiota dataset [51], as well as in other studies included in this dissertation related to the tasks of clustering with general high dimensional datasets (where the MC principle was already proven to increase performance of other machine learning algorithms in the unsupervised scenario for nonlinear data-pattern analysis [52], [53]), and to the task of community detection, with many real and synthetic networks.

All points discussed in this chapter will be presented in more detail below.

Part II. ENHANCED MARKOV CLUSTERING

Now that the notions of clustering, community detection, and related algorithms have been clarified together with the motivation of the current dissertation, this second part will present the work applied to the clustering algorithm MCL for its improvement as a pre-processing step of data within a network perspective.

4. Markov clustering

As commented in the Introduction Chapter 1.2.4 Graph-based (clustering) methods, Markov clustering (MCL) algorithm applies a strategy termed random walk for navigating the network and finding clusters in it. It works with two input parameters: expansion and inflation. Although in practice, the inflation is the parameter that regulates the cluster pattern findings at different scales of granularity (number of clusters). In the MCL website (https://micans.org/mcl), the MCL author Dr. van Dongen suggests applying inflation values between [1.1,10], with starting points to try 1.4, 2 and 6. In principle, MCL is a clustering algorithm that does not need the number of clusters to search as input parameter, different from other clustering methods like KNN, but it rather suggests them to the user influenced by the inflation parameter. Nevertheless, due to evaluations purposes in the here presented study, and since the number of clusters in the different analyzed datasets is known, an integration of a binary search was implemented. Therefore, the inflation parameter is automatically obtained by binary search, where the search stops when the correct number of clusters are found. Precisely, the value of inflation is searched in this case between the range of [1.1, 20] at different resolutions or steps [0.1, 0.01 and 0.001] to ensure the finding of the correct number of clusters. Suppose the first resolution (0.1) is not enough. In that case, the search continues at a lower resolution between the last two searched bounds until obtaining the desired number of clusters or arriving at the lowest resolution. The range of search between 1.1 and 20 is defined in order to span a large values interval (compared to the one suggested by the author of the algorithm, which is between 1.1 and 10) that accounts for the different scales of granularity of possible analyzed datasets.

As commented in Chapter 1.2.4 graph-based methods, MCL receives as input a similarity network for the clustering calculation. In this case, Pearson correlation, Spearman correlation and Euclidean similarities (ES) were employed. ES was defined according to the function in equation (2):

$$ES(x) = (1 - x/\max(x)) \tag{2}$$

Where x is a variable that indicates the Euclidean distance between a pair of samples and $\max(x)$ is the largest Euclidean distance between all pairs of samples.

As suggested in the MCL user manual (https://micans.org/mcl/MCL), a network construction and reduction step usually improves the clustering. It means that a sparsification of the weighted similarity matrix - that shapes (construction phase) a network topology by pruning (reduction phase) links with low similarity - is recommended before starting the clustering procedure. For example, the authors mention in their user guide to arbitrarily threshold and then discharge similarities lower than 0.7. After, they suggest rescaling the remaining value between [0,0.3]. This should be intended as a rescaling between zero and the maximum similarity value in the similarity matrix minus the threshold because the rescaling ensures stability in the stochastic flow clustering procedure. However, there are no indications for a general strategy to follow. In practice, there is a free parameter to tune for the similarity threshold, and there is no automatic procedure available. Unlikely, this threshold value should be arbitrarily specified by the user.

4.1. Enforcing network sparsity in Markov clustering

In order to overcome the network threshold issue described at the end of the previous paragraph, a simple but effective technical innovation to enforce sparsity of the similarity network is introduced. For the here presented MCL implementation, a strategy is proposed, according to which the threshold selection is done automatically by progressively pruning and rescaling the similarity network at increasing similarity threshold values (the unique values of the network weights are ranked and, starting from the lowest value in the list, they are increasingly tested as threshold). The function used for pruning and rescaling is expressed in equation (3):

$$f(x) = ReLU(x - t) = (x - t)^{+} = \max[0, (x - t)] \qquad (3)$$

Where x is the similarity matrix and t is the threshold (with values including 0 and lower than 1) tested at a certain iteration of the progressive pruning. When the network loses its topological integrity and separates in a number of components larger than one, the procedure stops, and this last threshold value is discharged, while the second last threshold value is selected to prune and to rescale the similarity values. In brief, this is a strategy to maximize sparsification of the network topology while retaining its one-component connectivity. The

MATLAB code implementation (Code 1) for enforcing network sparsity is displayed hereunder:

```
function [x,nc]=choose_cut(x, max_nc)
    % Inputs:
    % x: input network
    % max_nc: max number of components allowed

    % Outputs:
    % x: sparsified network
    % nc: number of components of the network

    uniq_weigths = unique(round(x,2));
    idxs = find(uniq_weigths>0);

    for i=1:length(idxs)

        cutoff = uniq_weigths(idxs(i));

        tmp_x1 = x1;
        tmp_x1(tmp_x1<cutoff) = 0;
        tmp_x1(tmp_x1>=cutoff) = tmp_x1(tmp_x1>=cutoff)-
cutoff;

        S=sparse(tmp_x1);
        [nc,~]=graphconncomp(S,'Directed', false);

        if nc > max_nc

            if i == 1
                warning('For the first cutoff the number
of components %d is already greater than %d',nc,max_nc);
                break;
            end
            cutoff = uniq_weigths(idxs(i-1));
            tmp_x = x;
            tmp_x(tmp_x<cutoff) = 0;
            tmp_x(tmp_x>=cutoff) = tmp_x(tmp_x>=cutoff)-
cutoff;

            S=sparse(tmp_x);
            [nc,~]=graphconncomp(S,'Directed', false);
            Break;
        end
    end

    x(x<cutoff) = 0;
    x(x>=cutoff) = x(x>=cutoff)-cutoff;
end
```

Code 1. Enforcing network sparsity function. The function takes as input a network and an integer related to the maximum number of components allowed while searching for the threshold (cutoff). max_nc is in this case 1.

5. Network navigability

One of the most fundamental and difficult problems in complex networks is the challenge to understand the relation between a network structure and its function [54]. The structure of the network refers not only to its visible topology, but also to its 'hidden metric space'. The power of understanding the hidden network topology is transformed into a more effective fashion to navigate through the network applying local knowledge rather than by using the global network information. In Boguñá et al. [54], the authors highlight two important properties of real complex networks, upon which the network navigability depends: (1) scale-free node degree (power-law) distribution (heterogeneous node degree), and (2) the number of triangles (clustering) in the network. Previous to Boguñá, Kleinberg [55] gave notions about what a model of navigable network requires. First, the network should contain (mostly) short paths between pairs of nodes. Secondly, nodes need partial knowledge about their structure network environment (which relates with the local information for efficient navigability of Boguñá) because too much information could cause a considerable volume of traffic.

It is here hypothesized that a boost in performance for clustering and community detection problems should be evidenced for MCL if the topology of the network being analyzed can efficiently approximate its hidden metric geometry space. This is achieved by favouring paths over others through MCL random walk following a greedy routing process over a network based on distance similarities between the nodes because they should approximate the hidden nonlinear manifold of the graph geometry.

6. Latent geometry inspired Markov clustering

As discussed in Chapter 4. ‘Network navigability’, the proposed rationale states that, in order to favour the simulation of random walks in MCL, the graph similarities (or dissimilarities) should approximate the closeness (or distances) on the hidden nonlinear manifold that characterizes the graph geometry [54], [56]. Indeed, in many networks, the information can efficiently flow according to a greedy routing procedure because their topology is emerging from this hidden geometry [54], whose hyperbolic and tree-like structure facilitates the greedy propagation [53], [54], [56]–[58]. Recently, Muscoloni et al. [59], [60] proposed two latent geometry-based pre-weighting techniques (one local and one global) as valuable strategies for approximating the pairwise geometrical distances between connected nodes of an unweighted network. In a later study of the same authors, the clustering algorithm affinity propagation was applied to the community detection task adopting two related dissimilarity matrices, containing dissimilarity values both for connected and disconnected nodes, which proved to simulate a more navigable geometry than other kernels previously designed for this purpose [61]. Here, in the context of community detection and according to the MCL algorithm requirements, the previous pre-weighting techniques are converted into similarity measures giving birth to an enhanced technique termed Latent Geometry Inspired - Markov Clustering (LGI-MCL). The converted similarities contain and merge two fundamental properties that characterize the hidden geometry of many real complex networks and thus might serve to improve stochastic flow simulations: node similarity (proximity or homophily), related with the network clustering and the concept of local attraction between common neighbours, and node popularity (centrality), related with the node degree [56].

The first approach - which is called the repulsion-attraction rule (RA) [59], [60] – assigns an edge weight adopting only the local information related to its adjacent nodes (neighbourhood topological information). The repulsive part behind RA involves that adjacent nodes with a high external degree (where the external degree is computed considering the number of neighbours not in common) should be geometrically far. Indeed, they represent hubs without neighbours in common, which - according to the theory of navigability of complex networks presented by Boguñá et al. [54] - tend to dominate geometrically distant regions. On the contrary, the attractive part of RA exploits that adjacent nodes sharing a high number of common neighbours should be geometrically close because, most likely, they have many things in common and therefore are similar. Thus, the RA (see below for the precise mathematical formula) is a simple and efficient approach that quantifies the trade-off between hub repulsion and common-neighbours-based attraction [59], [60]. The algorithm to compute the RA similarity for each link (*i, j*) in the network is the following (note that the dissimilarity value is marked with an asterisk):

I. Compute the RA pre-weighting as in equation (4) [59], [60]:

$$RA_{ij}^{*} = \frac{1 + e_i + e_j}{1 + cn_{ij}} \quad (4)$$

e_i is the number of external links of the node i (links that do not connect either to common neighbours with j or to j), e_j is the same for the node j; cn_{ij} is the number of common neighbours of the link (i, j).

II. Convert into a similarity value as in equation (5):

$$RA_{ij} = 1 + \frac{1}{1 + RA_{ij}^{*}} \quad (5)$$

Although inspired by the same rationale, the second similarity is global (exploits the entire network topology to compute each similarity value between pairs of nodes). In fact, as a first step, it makes a global-information-based pre-weighting of the links, using the edge-betweenness-centrality (EBC) to approximate distances between nodes and regions of the network [60]. EBC is indeed a global topological network measure that assigns to each link a value of centrality related to its importance in propagating information across different network regions. The assumption is that central edges are bridges that tend to connect geometrically distant regions of the network, while peripheral edges tend to connect nodes in the same neighbourhood. The higher the EBC value of a network link, the more information will pass through that link. The algorithm to compute the EBC similarity for each link (i, j) in the network is the following:

I. Compute the EBC pre-weighting as in equation (6) [60]:

$$EBC_{ij}^{*} = \sum_{s,t} \frac{\sigma(s,t|e_{ij})}{\sigma(s,t)} \qquad (6)$$

s,t is any combination of network nodes; $\sigma(s,t)$ is the number of shortest paths between *s* and *t*; $\sigma(s,t\backslash e_{ij})$ is the number of shortest paths between *s* and *t* passing through the link (i, j).

II. Convert into a similarity value as in equation (7):

$$EBC_{ij} = 1 + \frac{1}{1 + EBC_{ij}^{*}} \qquad (7)$$

A novel similarity measure (ER) that merges the previous ones (EBC and RA) for each link (i, j) in the network is also introduced as follows:

I. Compute the pre-weightings RA_{ij}^{*} and EBC_{ij}^{*}.

II. Convert into a unique similarity value as in equation (8):

$$ER_{ij} = 1 + \frac{1}{1 + RA^*_{ij}} + \frac{1}{1 + EBC^*_{ij}} \tag{8}$$

6.1. Software availability

The LGI-MCL code is freely available under the Github repository: https://github.com/biomedical-cybernetics/LGI-MCL.

7. Minimum curvilinear Markov clustering

7.1. Minimum curvilinearity

Minimum Curvilinearity (MC) [57] – introduced by Cannistraci et al. in 2010 - was invented with the aim to reveal nonlinear patterns in data, especially in the case of datasets with few samples and many features. Nonlinearity is often driven by hierarchy and - under the hypothesis that at least part of data nonlinearity is associated to a generative process that forces sample hierarchy - the basic idea behind MC is to exploit the hierarchical organization and structure of the samples in the feature space to approximate their pairwise nonlinear relationship. Indeed, the MC principle suggests that nonlinear curvilinear distances between samples can be estimated as transversal paths over their Minimum Spanning Tree (MST), which is constructed according to a certain distance (Euclidean, correlation-based, etc.) in a multidimensional feature space. The illustration in Figure 14 reflects a case where the computation of the Euclidean distance between points in the space might be impossible to compute due to certain energetic constraints, which means that points will never lay in the zone of the purple line between points P_1 and P_2 and therefore the Euclidean distance does not reflect the real distance between those points (Figure 14A). Contrarily, when computing the distance between points P_1 and P_2 as a function of the MST edge weights (calculated previously with a specific distance function), it allows exploiting the data organization and structure to estimate nonlinear relationships avoiding the possible data constraints (Figure 14B).

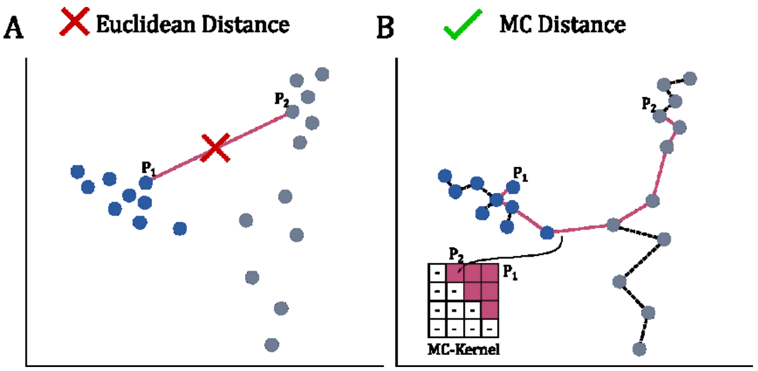

Figure 14. Illustration of MC-kernel computation. (A) Issues to compute Euclidean distance between points due to data constraints, i.e.. zone energetically inaccessible by data points. (B) Distance computation between points following a greedy routing through the MST.

In this work, Pearson-correlation-based, Spearman-correlation-based and Euclidean-based distances to compute the MST are considered. The collection of all MC pairwise distances forms a distance matrix called the MC-distance matrix or MC-kernel, which can be used as input in algorithms for dimensionality reduction, clustering, classification and generally in any type of machine learning [53], [57].

7.2. From a linear to a nonlinear approach

With the purpose of creating and testing a nonlinear variant of the MCL algorithm in a clustering framework, a method termed minimum curvilinear Markov clustering (MC-MCL) is here proposed. The idea is the following: the MC-kernel (refer to Chapter 7.1. Minimum curvilinearity for more details) is a nonlinear kernel that expresses the pairwise relations between samples as a value of distance: a small samples distance indicates high sample similarity, while a large samples distance indicates low sample similarity. As anticipated in Chapter 7.1 Minimum curvilinear, in this study, three different distances (Pearson-correlation-based, Spearman-correlation-based and Euclidean-based) are considered to build the MST to construct the MC-kernel. Two different procedures to derive the MC-similarity kernels are described below, considering correlation-based distances and the Euclidean distance. In case the MST and the associated MC-distance kernel are built with Pearson-correlation-distance or Spearman-correlation-distance, the MC-distance kernel is inverted to get a MC-similarity kernel, and all negative values (in case of t=0) or all values lower than a threshold t are put to 0, where $t \in [0,1)$, using the function in equation (9):

$$f(x) = ReLU(1 - x - t) = (1 - x - t)^{+} = \max[0, (1 - x - t)] \tag{4}$$

Where: x is the original value of the pairwise MC distance; t is the same threshold defined in equation (3) in Chapter 4.1. Enforcing network sparsity in Markov clustering to enforce the network sparsity (and it is automatically detected using the same strategy described in that Chapter); and $f(x)$ is the derived value of the pairwise MC similarity. Therefore, small $f(x)$ values (close to zero) indicate low sample similarity, and large $f(x)$ values (close to one) indicate high sample similarity.

Now, a clarification to an important property of the MC-similarity defined in equation (9) is highlighted, together with the reason of why this inversion is well-posed. The MST is computed on a correlation-based distance (CD) that is defined as in equation (10):

$$0 \leq CD(y) = (1 - y) \leq 2, \qquad with - 1 \leq y \leq 1 \tag{10}$$

Where y is the original Pearson correlation value and CD = 0 means high similarity, CD = 1 means random similarity, and CD = 2 means anti-similarity (nothing can be more dissimilar than the opposite trend).

As a consequence of this mathematical codification of *CD*, any MC distance that is larger than 1 tends to overcome an intrinsic threshold of random similarity. Hence MC distances larger than one can be interpreted as less significant than random. This mechanism, which seems naïve, is in reality refined and allows directly to assess that any MC-distance smaller than 1 is under the natural threshold of random sample similarity association (and should be accepted); therefore any MC-distance larger than 1 can be neglected because is less significant than random similarity. And this is actually what is defined mathematically with the ReLU function applied after the 1-*x*-*t* inversion in (9). For example: if we fix $t = 0$, a MC-distance $x = 1.2$ is larger than 1 and therefore should be neglected as MC-similarity, indeed $f(x) = \mathrm{ReLU}(1 - 1.2) = 0$. More in general, the equation (9) suggests that we can learn a similarity threshold $t \geq 0$ (on the weights of the network), which preserves the network structure and discharge links that are not significant to preserve the integrity of the network flows (because they do not disconnect the network). If $t = 0$, sample similarities (links) that are less significant than random similarities are discharged. If $t > 0$, also sample similarities (links) that are not significant to preserve the stochastic flows are discharged. This naïve strategy allows to induce sparsity in the MC-similarity kernel by means of an intrinsic and self-adaptive thresholding mechanism that neglects connectivity with similarity worse than random and, as a matter of fact, it avoids that the stochastic flows of MCL run on network branches or zones that would suffer unreliable connectivity.

In case the MST and the associated MC-distance kernel are built with Euclidean-distance, the MC-distance kernel is inverted to get an MC-similarity kernel according to the following function in equation (11):

$$f(x) = ReLU\left(1 - \frac{x}{max(x)} - t\right) = \left(1 - \frac{x}{max(x)} - t\right)^{+} = max[0, \left(1 - \frac{x}{max(x)} - t\right)] \qquad (11)$$

Where x is a variable that indicates the Euclidean-based MC-distance between a pair of samples; $max(x)$ is the largest Euclidean-based MC-distance between all the pairs of samples; and t is the same threshold defined in equation (3) in Chapter 4.1. Enforcing network sparsity in Markov clustering, to enforce the network sparsity (and it is automatically detected using the same strategy described there). A technical detail is that for the computation of the MC-distance kernel (hence before the inversion procedures described in equation (2)), three alternatives are used: 1) original distances in the MC-kernel (MC-MCLo), 2) their square root $x^{1/2}$ (MC-MCLs), or 3) their logarithm $\log(1+x)$ (MC-MCLl). As already investigated in [57], the square root and the log operators can attenuate the estimation of large distances and, on the contrary, amplifies the estimation of short distances. Consequently, they help to regularize the nonlinear distances inferred over the MST in order to use them for message passing [57] (such as for AP) or stochastic flow simulation (such as for MCL) clustering algorithms (for more details on the MC-similarity construction, please refer to the MATLAB code in Code 2).

The final steps are the same automatic threshold selection described in Chapter 4.1 in order to build the sparse similarity network for the classical MCL, and then to run the standard MCL algorithm on the MC-similarity sparse network. In practice, this new algorithm for clustering is a nonlinear and sparse version of the classical MCL, where the nonlinearity is MC-driven and the sparsity is self-learned using the threshold that maximizes pruning without losing the one-component similarity network connectivity (refer to Chapter 4.1. 'Enforcing network sparsity in Markov clustering' for more details).

```
% dist refers to the distance used for the MST calculation
% e.g. Euclidean, Pearson correlation-based, etc.

if factor==1
    matr=squareform(pdist(x,dist));
elseif factor==2
    matr=sqrt(1+squareform(pdist(x,dist)));
elseif factor==3
    matr=log(1+squareform(pdist(x,dist)));
end

xx =
graphallshortestpaths(adjacence(minspantree(graph(matr),'m
ethod','sparse')),'directed','false');
if strcmp(dist,'euclidean')
    x = xx./max(max(xx));
end
x = 1-xx;
```

Code 2. MC-Similarity kernel construction. The MATLAB build-in function minspantree receives as input and delivers as output a graph object. The custom adjacence function transform the graph object into a sparse matrix, which is the needed input for the MATLAB built-in function graphallshortestpaths, in charge of computing the pairwise node distances along the MST. The expression 1 – xx transforms the values from dissimilarities to similarities.

7.3. Minimum curvilinear Markov clustering multi-MST variants

In order to enhance the proposed MC-MCL algorithm, several MC-MCL variants based on different topological properties were here tested, so as to improve the stochastic random walk through the approximation of the network's hidden geometry, thus improving network navigability (refer to Chapter 5. 'Network navigability' for more details). Wherefore, the efforts were put into alternative constructions for the MC-kernel. All variants are based on the generation of the base MST with the union of an alternate MST aiming to enhance the local connectivity of the network.

7.3.1. MC-MCL - MST high degree removal

This MC-MCL strategy, here referred as to MC-MCLhdr, assumes that by removing hubs from the MST, new paths will link neighbors of the hub, thus increasing the shortest paths possibilities between nodes through zones with high traffic, yet preserving the idea that the node pairwise distances over the minimum spanning tree approximates the hidden and nonlinear network geometry space (Figure 15A-C). This variant introduces the necessity of specifying a parameter to determine high degree nodes in the graph, whose value will depend on the data-MST topology. This value is entered as the quantile of high degree nodes that will be removed for the computation of the second MST. As illustration, the pink arrow in Figure 15A denotes the node with a high degree (4 links) to be removed. Subsequently, a second MST without the removed node(s) is computed (Figure 15B), to formerly apply the union between both MSTs and compute the MC kernel (Figure 15C).

In Code 3 is displayed the function for the MC-MCLhdr kernel computation. It receives as input two parameters: the pairwise distance matrix from the samples and the quantile parameter value to determine which 'high degree' nodes to remove. It gives as output the MChdr-distance kernel, which will be later transformed into the so-called MC-similarity (Please refer to Chapter 7.2. 'From a linear to a nonlinear approach' for details on the MC-similarity computation).

```
function xx = MSTHighDegreeRemoval(matr,q)
    % first MST computation
    mst =
adjacence(minspantree(graph(matr),'method','sparse'));

    % getting the degree of each node in MST
    dgrs = degree(graph(mst,'lower'));

    % getting idx of the high degree node(s) as function
of q
    highDegreeN = dgrs >= quantile(dgrs,1-q);

    % calculating MST without highest degree nodes
    numbSamples = size(matr,1);
    tempMst = sparse(numbSamples,numbSamples);
    tempMst(~highDegreeN,~highDegreeN) =
adjacence(minspantree(graph(matr(~highDegreeN,~highDegreeN
)),'method','sparse'));

    % union of MSTs
    [val,idxMst] = setdiff(tempMst,mst);
    for j = 1:length(idxMst)
        mst(idxMst(j)) = val(j);
    end

    % kernel computation
    xx = graphallshortestpaths(mst,'directed','false');
    clear mst tempMst matr
end
```

Code 3. Computation of the MC-MCLhdr kernel. The function receives as input the full distance matrix between nodes and the quantile parameter. It gives as output the MC-dissimilarity kernel.

7.3.2. MC-MCL – MST high NBC removal

Similar to the MST high degree removal, this strategy, here termed MC-MCLhnr, seeks to decrease the traffic in zones of high information movements. This is achieved by calculating the node-betweenness-centrality (NBC) of each node, and removing those with high values. NBC is a measure of centrality based on shortest paths. Nodes with higher betweenness-centrality values tend to dominate the network because more information passes through them. Equivalently to the high degree removal variant, a parameter for determining high NBCs needs to be specified, whose value is dataset dependent.

The procedure starts by selecting the node(s) with high NBC values to be removed, green arrow in Figure 15D. Subsequently, a second MST without the removed node(s) is computed (Figure 15D), to formerly apply the union between both MSTs and compute the MC kernel (Figure 15E).

In Code 4 is displayed the function for the MC-MCLhnr kernel computation. As for MC-MCLhnr, it receives as input two parameters: the pairwise distance matrix from the samples and the quantile parameter value to determine which 'high NBC' ranked nodes to remove. It gives as output the MChnr-distance kernel, which will be later transformed into the so-called MC-similarity (Please refer to

Chapter 7.2. ‘From a linear to a nonlinear approach’ for details on the MC-similarity computation).

```
function xx = MSTHighNBCRemoval(matr,q)
    % first MST computation
    mst =
adjacence(minspantree(graph(matr),'method','sparse'));

    % getting the NBC value of each node in MST
    NBC = betweenness_centrality(mst);

    % getting idx of the high NBC node(s) as function of q
    highNBCN = NBC >= quantile(NBC,1-q);

    % calculating MST without highest NBC ranked nodes
    numbSamples = size(matr,1);
    tempMst = sparse(numbSamples,numbSamples);
    tempMst(~highNBCN,~highNBCN) =
adjacence(minspantree(graph(matr(~highNBCN,~highNBCN)),'me
thod','sparse'));

    % union of MSTs
    [val,idxMst] = setdiff(tempMst,mst);
    for j = 1:length(idxMst)
        mst(idxMst(j)) = val(j);
    end

    % kernel computation
    xx = graphallshortestpaths(mst,'directed','false');
    clear mst tempMst matr
end
```

Code 4. Computation of the MC-MCLhnr kernel. The function receives as input the full distance matrix between nodes and the quantile parameter. It gives as output the MC-dissimilarity kernel.

7.3.3. MC-MCL Dual

The last strategy for MC-kernel improvement considers the here so-called dual MST, and therefore termed MC-MCLdual. The dual term refers to the generation of a second MST with the constraint that all edges from the first one cannot be accessed by the construction of the new MST (Figure 15F). This process can be repeated many times where, in every new MST construction, the edges from all previous MST networks are blocked and cannot be used. Finally, the union of all (original and dual) MSTs generated is employed to calculate the

MC kernel (Figure 15G). Naturally, the parameter to select is the number of dual MSTs to generate, which can be data-dependent. Although regularly, this value to consider is low. The function to compute the kernel of MC-MCLdual is provided in Code 5.

```
function [xx,mst] = dualMST(matr,exh)
    % first MST computation
    mst = 
adjacence(minspantree(graph(matr),'method','sparse'));

    for i = 1:exh
        % deleting distances from nodes in input distance 
matrix ‘matr’ for dual MST generation
        tempMst = mst;
        ind = find(tempMst~=0);
        [row,col] = ind2sub(size(tempMst),ind);
        indInv = sub2ind(size(tempMst),col,row);

        % deleting in lower part of matrix
        matr(ind) = 0;

        % deleting in upper part of matrix for symmetry
        matr(indInv) = 0;

        % computing the dual MST
        tempMst = 
adjacence(minspantree(graph(matr),'method','sparse'));

        % check if Dual MST gives more than one unique 
component
        if graphconncomp(tempMst,'Directed', false) ~= 1
            break;
        end

        % union of MSTs
        [val,idxMst] = setdiff(tempMst,mst);
        for j = 1:length(idxMst)
            mst(idxMst(j)) = val(j);
        end
    end

    xx = graphallshortestpaths(mst,'directed','false');
    clear mst tempMst matr
end
```

Code 5. Computation of the MC-MCLdual kernel. The function receives as input the full distance matrix between nodes and the number of dual MSTs to generate. It gives as output the MC-dissimilarity kernel.

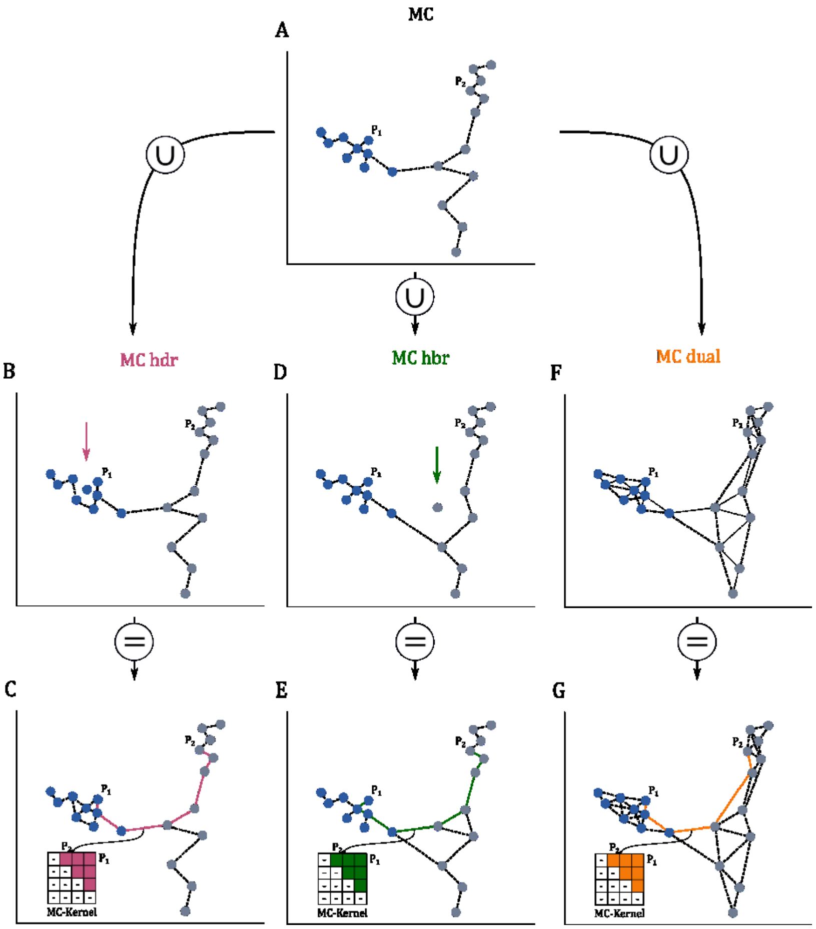

Figure 15. Ilustration of the three MC kernel multi-MST variants for MC-MCL. Hdr in magenta, hbr in green and dual in orange.. The arrows between panels point to the union of original MST (panel A) and a variant (panels B, D or F) and the resulting network from the union used for the MC-kernel computation (panels C, E or G) (A) original MST from where to construct the MC original kernel. (B) Second MST originated after removing the high degree node (the pink arrow points toward the removed node). (C) Union of original and hdr-based MST for the MC hdr-kernel computation. (D) Second MST originated after removing the high NBC ranked node (the green arrow points toward the removed node). (E) Union of original and hbr-based MST for the MC hbr-kernel computation. (F) dual MST representation originated after blocking the original MST (grey) links. (G) Union of original and dual MST for the MC dual-kernel computation.

7.4. Isomap-inspired Markov clustering

Isomap [62] is an algorithm tailored for dimensionality reduction, an unsupervised learning technique that aims to decrease the number of dimensions of a dataset into a meaningful lower space (please refer to chapter 3. ‘Motivation’ for more details). A famous and widely employed linear dimensionality reduction technique is called principal component analysis (PCA) [46], [47]. Unlike PCA, Isomap is a nonlinear technique that focuses on estimating the hidden geometrical data manifold through neighbourhood connections. It needs a parameter k to determine the number of connections of each node with its closest neighbours for constructing the so-called proximity network. Then, it computes the pairwise shortest path (distance) between the nodes, to finally apply the embedding into a lower dimension.

Taking inspiration from Isomap, the presented MCL variant, termed isoMCL, takes advantage of the neighbourhood network connectivity (Figure 16). As for Isomap, it constructs the iso-kernel by creating a proximity graph where each node is connected to the k closest neighbours without losing the 1 unique component connectivity (e.g. regularly k = 1 creates a network with many separated modules, and therefore k needs to be higher) (Figure 16A). Then, the computation of

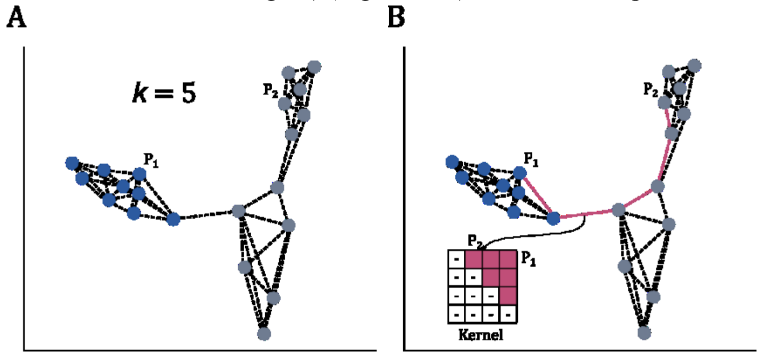

Figure 16. Illustration if the isoMCL kernel computation. (A) Proximity network construction inspired from the Isomap algorithm with k = 5. (B) Construction of the isoMCL kernel by pairwise node distance over the proximity network.

all node pairwise distances is calculated (Figure 16B). The dissimilarity iso-kernel is transformed to similarities following the same strategy utilized as for the MC kernel (Please refer to Chapter 7.2. ‘From a linear to a nonlinear approach’ for more details). The function to compute the kernel of isoMCL is provided in Code 6.

```
function  [xx,nc] = isoKernel(x, k,norm)
    %Maps the high-dimensional samples in 'x' to a low
dimensional space using
    %Isomap or ISO (coded 5-FEBRUARY-2011 by Gregorio
Alanis-Lobato) - Modified by Claudio Durán 10-NOVEMBER-
2020

    %INPUT
    %   x => Matrix with samples on rows and features on
columns
    %   k => Number of nearest neighbours to construct the
proximity graph
    %   norm => type of norm to compute the distance
    %OUTPUT
    %   xx => isoMCL dissimilarity kernel
    %   nc => number of components

    %Number of samples
    samples = size(x, 1);
    dist = pdist2(x, x, norm);

    %Trick so that the diagonal 0 distances are not
considered
    dist(logical(eye(samples))) = Inf;

    % Allocate space for the proximity graph and construct
it
    graph = sparse(samples,samples);

    for i = 1:samples
        %Find the k nearest neighbours of sample i and
connect them to i in the proximity graph
        [~, idx] = sort(dist(i, :)); idx=idx(1:end-1);
        for j = 1:k
            graph(i, idx(j)) = dist(i, idx(j));
        end
    end

    % creating symmetrical matrix
    graph = max(graph, graph');

    % kernel computation
    xx = graphallshortestpaths(graph, 'directed',
'false');
```

```
    % number of graph components (it should always be 1)
    nc = graphconncomp(graph,'Directed',false);
    if nc > 1
        warning('The number of component in the network is
greater than 1');
    end
end
```

Code 6. Computation of the isoMCL kernel. The function receives as input the matrix with samples in the high dimensional space, the k value for the neighbourhood proximity network and the norm (i.e. Euclidean) of the distance calculation. It gives as output the isoMCL-dissimilarity kernel and the number of components of the proximity network generated.

7.5. Nonlinear MCL time complexities

Because the new variants are the design of similarity kernels that goes into MCL, the first step to clarify their time complexity is by calculating the complexity of MCL alone. Stjin van Dongen, the author of MCL, claims that the complexity time of this algorithm is `O(N k`^{2}`)`, where `N` is the number of nodes in the graph, and `k` is the number of resources allocated per node. Regarding `k`, it is also stated that 'the maximum number of resources allocated per node directly translates to the maximum number of nonzero entries kept per column' due to a use of a sparse matrix, explaining his time complexity. Therefore, `k` is actually related to the edges of the network. For this reason, we denote the time complexity of MCL as `O(N E`^{2}`)`. This time complexity is achieved with regular MCL when given a sparse network, and a certain unique inflation parameter. Here, two previous steps are added in order to make MCL automatic. First, since it is worked with high dimensional data, regularly the computation of the similarity between nodes ends up with a full (and not sparse) matrix. Therefore, as aforementioned, a sparcification on the network is applied. This step is governed by the amount of positive similarity values (edge weights > 0 are kept in the network). Thus, the minimum positive weighted edges are pruned one by one until the one unique component is broken. Therefore, this step has a linear time complexity of `O(E)`, where `E` is related to the number of edges in the network. Note that in practice, the one unique component needs a certain amount of edges in the network and therefore the mentioned time complexity will never be achieved. Secondly, since the number of clusters to find is known, the inflation (a parameter that defines the cluster membership outputs of MCL at different granularities) is automatically explored by a binary search strategy. Therefore, this second step would have a time complexity of `O(log I)`, being I the inflations to search where the correct number of clusters is found. Nonetheless, `I` in this case is a constant, because we search through a specific range for `I`. Finally, taking into account all time complexities previously discussed, the time complexity of the automatic MCL remains as $O(N\ E^2)$ followed by the fact that the runtime is always dominated by the highest power.

Regarding the nonlinear variants, for the MC kernel versions, the time complexity of this kernel is governed mainly by two steps: The generation of the MST, and the calculation of the distances (shortest paths) over the MST. The MST calculation is done by means of the kruskal's algorithm, whose time complexity is `O(E log N)`, where `E` refers to the number of edges and `N` the number of nodes. In the case of the calculation of all shortest paths over the MST, the Johnson's algorithm is applied, whose time complexity is `O(N*log(N)+N*E)`, bein `E` the number of edges and `N` the number of nodes. Thus, the time complexity of MC-MCL remains with the highest power $O(N\ E^2)$.

Similarly, in the case of isoMCL, the kernel computation is governed by the number of nodes, and the k closest neighbours to add to each node. For the 'closest' neighbours, sorting the distances from one node to the rest is needed, being its time complexity of `O(E log N)`, where `N` is the number of nodes. Therefore its time complexity is `O(N^2 log N).` Thus, the time complexity of isoMCL remains as well as $O(N\ E^2)$.

7.6. Software availability

The MC-MCL code is freely available in a github repository under: https://github.com/biomedical-cybernetics/minimum-curvilinear-Markov-clustering.

Part III. CASE STUDIES

8. Evaluation framework

The evaluations for community detection and clustering problems slightly differ. For community detection, a measure widely employed and here adopted is termed normalize mutual information (NMI), whereas for clustering, besides NMI, the measures accuracy and adjusted rand index (ARI) were additionally adopted.

NMI is based on entropy, which can be defined as the information contained in a distribution $p(x)$ as in equation (12):

$$H(X) = \sum_{x \in X} p(x) \log p(x) \tag{12}$$

The mutual information is the shared information between two distributions (equation (13)):

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log\left(\frac{p(x,y)}{p_1(x)p_2(y)}\right) \tag{13}$$

To normalize the value between 0 and 1 the formula in equation (14) can be applied:

$$NMI = \frac{I(X,Y)}{\sqrt{H(X)H(Y)}} \tag{14}$$

Considering a partition of the nodes in communities as a distribution (probability of one node falling into one community), the previous equations (12, 13, 14) allow computing the matching between the annotations obtained by the community detection algorithm and the ground-truth communities of a network. A MATLAB implementation available at http://commdetect.weebly.com was here used. As suggested in the code, when $\frac{N}{C} \leq 100$, where N represents the number of nodes and C the number of communities, the NMI should be adjusted in order to correct for chance [63], [64].

Accuracy (Acc in tables) is a common measure that evaluates the number of correctly predicted labels with respect to the total number of predictions. Given a set of S of n elements, and two partitions of those elements, namely $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, the accuracy can be computed by counting the agreements between both partitions and dividing it by the number of elements of those partitions as in equation (15).

$$Accuracy = \frac{|X \cap Y|}{n} \qquad (15)$$

Adjusted Rand Index (ARI) [65], [66], like NMI, assesses the agreement between two partitions, in this case between the true labels of the data and the labels assigned by the clustering algorithm. The rationale behind ARI is related to pair counting measures, which are calculated based on the cluster and class membership of pairs of data points agreement. The overlap information between the two partitions can be written as a contingency table.

Given a set of S of n elements, and two partitions of those elements, namely $X = \{X_1, X_2, \ldots, X_r\}$ and $Y = \{Y_1, Y_2, \ldots, Y_s\}$, the overlap between X and Y can be expressed as a contingency table where each entry n_{ij} denotes the number of agreements (intersection) between X and Y (Table 1) [65].

$X \backslash Y$	Y_1	Y_2	$\dots$	Y_s	sum
X_1	n_{11}	n_{12}	$\dots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\dots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\dots$	n_{rs}	a_r
sum	b_1	b_2	$\dots$	b_s	

Table 1. Contingency table expressing the overlap between two partition X and Y.

Consequently, ARI is calculated employing the values in Table 1 as in equation (16).

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_j}{2}\right]/\binom{n}{2}}{\frac{1}{2}\left[\sum_i\binom{a_i}{2} + \sum_j\binom{b_j}{2}\right] - \left[\sum_i\binom{a_i}{2}\sum_j\binom{b_j}{2}\right]/\binom{n}{2}} \qquad (16)$$

In the case of the clustering methods, the results reported in each table for each dataset are the best results considering the most effective combination of normalization, distance options (including factors) and optimal parameter (if applied). Best meaning the result that offers the highest values according to a mean rank taking into account accuracy, ARI and NMI.

9. Community detection analysis

In the next section, after describing the procedure behind MCL (please refer to chapter 4. 'Markov clustering' for more details), and recall a collection of network science notions, at the interface between network topology and network geometry [56], [59], [60], [67]–[69], based on which the proposed LGI rationale can guide the steps to design similarity measures to boost algorithms based on network navigability protocols (please refer to chapter 6. 'Latent geometry inspired Markov clustering' for more details), the respective community detection analysis starts. Here, the aim is to investigate the extent to which LGI measures can be employed to improve MCL community detection. The analyses performed in this chapter compare the LGI-MCL variants against the original MCL and the state of the art methods Infomap and Louvain. After presenting the results of wide evaluations both on real networks, real networks with noisy information and on a large benchmark of synthetic 'realistic' networks, finally, a discussion with advantages and limitations of the LGI-MCL approach will be addressed.

9.1. Real network datasets

The community detection methods have been tested on 8 real networks, which represent differing systems: Karate; Opsahl_8; Opsahl_9; Opsahl_10; Opsahl_11; Polbooks; Football; Polblogs. The networks have been transformed into undirected, unweighted, without self-loops, and only the largest connected component has been considered. The information of some basic statistics is available in Table 2. *N* is the number of nodes. *E* is the number of edges. The parameter *m* refers to half of the average node degree, and it is also equal to the ratio *E/N*. *Cl* is the average clustering coefficient, computed for each node as the number of links between its neighbours over the number of possible links [42]. The parameter γ is the exponent of the power-law degree distribution, fitted from the observed degree sequence using the maximum likelihood[4] procedure developed by Clauset et al. [70] and released at http://tuvalu.santafe.edu/~aaronc/powerlaws/. *C* is the number of ground-truth communities.

[4] A maximum likelihood estimation is a method for estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable.

	N	***E***	***m***	***Cl***	*γ*	***C***
karate	34	78	2.3	0.59	2.1	2
opsahl 8	43	193	4.5	0.61	8.2	7
opsahl 9	44	348	7.9	0.68	5.9	7
opsahl 10	77	518	6.7	0.66	5.1	4
opsahl 11	77	1088	14.1	0.72	4.9	4
polbooks	105	441	4.2	0.49	2.6	3
football	115	613	5.3	0.40	9.1	12
polblogs	1222	16714	13.7	0.36	2.4	2

Table 2. Statistics of real networks. Number of nodes N, number of edges E, half of average node degree m, clustering coefficient Cl, power-law degree distribution exponent γ, number of communities C.

Karate Club

The first network is about the Zachary's Karate Club [71], it represents the friendship between the members of a university karate club in US. The communities are formed by a split of the club into two parts, each following one trainer.

Opsahl

The networks from the second to the fifth (Table 2) are intra-organisational networks from [72] and can be downloaded at https://toreopsahl.com/datasets/#Cross_Parker. Opsahl_8 and Opsahl_9 come from a consulting company, and nodes represent employees. In Opsahl_8 employees were asked to indicate how often they have turned to a co-worker for work-related information in the past, where the answers range from: 0 - I don't know that person; 1 - Never; 2 - Seldom; 3 - Sometimes; 4 - Often; 5 - Very often. Directions were ignored. The data was turned into an unweighted network by setting a link only between employees that have at least asked for information seldom (2).

In the Opsahl_9 network, the same employees were asked to indicate how valuable the information they gained from their co-worker was. They were asked to show how strongly they agree or disagree with the following statement: “In general, this person has expertise in areas that are important in the kind of work I do.” The weights in this network are also based on the following scale: 0 - Do Not Know This Person; 1 - Strongly Disagree; 2 - Disagree; 3 - Neutral; 4 - Agree; 5 - Strongly Agree. A link was set if there was an agreement (4) or strong agreement (5). Directions were ignored.

The Opsahl_10 and Opsahl_11 networks come from the research team of a manufacturing company, and nodes represent employees. The annotated communities indicate the company locations (Paris, Frankfurt, Warsaw and Geneva). For Opsahl_10 the researchers were asked to indicate the extent to which their co-workers provide them with the information they use to accomplish their work. The answers were on the following scale: 0 – I do not know this person / I never met this person; 1 – Very infrequently; 2 – Infrequently; 3 – Somewhat frequently; 4 – Frequently; 5 – Very frequently. An undirected link was set when there was at least a weight of 4.

For Opsahl_11 the employees were asked about their awareness of each other’s knowledge (“I understand this person’s knowledge and skills. This does not necessarily mean that I have these skills and am knowledgeable in these domains, but I understand what skills this person has and domains they are knowledgeable in.”). The weighting was on the scale: 0 – I do not know this person / I have never met this person; 1 – Strongly disagree; 2 – Disagree; 3 – Somewhat disagree; 4 – Somewhat agree; 5 – Agree; 6 – Strongly agree. A link was set when there was at least a 4, ignoring directions.

Polbooks

The Polbooks network represents frequent co-purchases of books concerning US politics on amazon.com. Ground-truth communities are given by the political orientation of the books as either conservative, neutral or liberal. The network is unpublished but can be downloaded at http://www-personal.umich.edu/~mejn/netdata/, as well as with the Karate, Football and Polblogs networks.

Football

The Football network [35] presents games between division IA colleges during regular season fall 2000. Ground-truth communities are the conferences that each team belongs to.

Polblogs

The Polblogs [73] network consists of links between blogs about the politics in the 2004 US presidential election. The ground-truth communities represent the political opinions of the blogs (right/conservative and left/liberal).

9.2. Synthetic networks generated by the nPSO model

The Popularity-Similarity-Optimization (PSO) model [56] is a generative network model recently introduced in order to describe how random geometric graphs grow in the hyperbolic space. In this model, the networks evolve optimizing a trade-off between node popularity, abstracted by the radial coordinate, and similarity, represented by the angular distance. The PSO model can reproduce many structural properties of real networks: clustering, small-worldness (concurrent low characteristic path length and high clustering), node degree heterogeneity with power-law degree distribution and rich-clubness[5]. However, being the nodes uniformly distributed over the angular coordinate, the model lacks a non-trivial community structure.

[5] Rich-clubness refers to nodes with large number of edges that tend to be well connected between each other and form a compact group [95].

The nonuniform PSO (nPSO) model [74], [75] is a variation of the PSO model that exploits a nonuniform distribution of nodes over the angular coordinate in order to generate networks characterized by communities, with the possibility to tune their number, size and mixing property. The adoption of a Gaussian mixture distribution of angular coordinates, with communities that emerge in correspondence with the different Gaussians, and the parameter setting suggested in the original study [74], [75] was considered. Given the number of components C, they have means equidistantly arranged over the angular space, $\mu_i = \frac{2\pi}{C} \cdot (i-1)$, the same standard deviation fixed to 1/6 of the distance between two adjacent means, $\sigma_i = \frac{1}{6} \cdot \frac{2\pi}{C}$, and equal mixing proportions, $\rho_i = \frac{1}{C}$ $(i = 1 \dots C)$. The community memberships are assigned considering for each node the component whose mean is the closest in the angular space. The other parameters of the model are the number of nodes N, half of the average node degree m, the network temperature T[6] (inversely related to the clustering) and the exponent γ of the power-law degree distribution. Given the parameters (N, m, T, γ, C), for details on the generative procedure, please refer to the original study [74], [75].

9.3. Real network analysis results

In Table 3 the performance comparison of MCL in its original form, the three LGI-MCL variants (EBC, RA and ER) and the state of the art methods for community detection Infomap and Louvain are reported. In addition, two in-silico experiments were made to test the robustness of the techniques in the case of noise injection in the real topologies. In the first case, the network structure was perturbed by the random deletion of 10% of the links. This procedure was repeated for 100 realizations, and the average results are reported in Table 4. This

[6] The temperature of a network regulates its clustering. At T = 0, the clustering is maximized, with T close to 1, the network can be seeing as one unique cluster [96].

experiment simulates the behaviour of the algorithms in case of partial (10%) missing topological information. In the second case, the network structure was perturbed by the random addition of 10% of the links. This procedure was repeated for 100 realizations, and the average results are reported in Table 5. This experiment simulates the behaviour of the algorithms in the case of partial (10%) addition of wrong topological information.

	Infomap	**Louvain**	**LGI-MCL ER**	**LGI-MCL RA**	**LGI-MCL EBC**	**MCL**
karate	0.55	0.46	**0.83**	**0.83**	0.73	0.73
opsahl 8	**0.69**	0.55	0.59	0.55	0.55	0.55
opsahl 9	**0.47**	0.41	0.39	0.40	0.40	0.43
opsahl 10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
opsahl 11	**1.00**	0.96	0.96	0.75	0.75	0.68
polbooks	0.52	0.50	**0.57**	**0.57**	**0.57**	**0.57**
football	0.92	**0.93**	**0.93**	**0.93**	**0.93**	**0.93**
polblogs	0.52	**0.64**	0.00	0.00	0.00	0.00
mean NMI	**0.71**	0.68	0.66	0.63	0.62	0.61
mean ranking	**3.06**	3.69	3.19	3.56	3.81	3.69

Table 3. ***Community detection on real networks.*** *The table reports the Normalized Mutual Information (NMI) computed between the ground truth communities and the ones detected by every community detection algorithm for 8 real networks. NMI = 1 indicates a perfect match between the two partitions of the nodes. The methods are ranked by mean NMI over the dataset. The best result for each network, as well as the best mean results, are marked in bold.*

As a first key result, LGI-MCL outperforms the original MCL in all three scenarios. Remarkably, LGI-MCL ER displays a higher mean NMI than the other LGI-MCL variants in the original topologies and in the random removal experiment, whereas they equally perform in the random addition framework. Furthermore, LGI-MCL ER reaches a mean NMI close to the state of the art method Louvain and a better mean ranking, highlighting the importance of merging the RA and EBC measures in a unique combined similarity. Lastly, Infomap attains overall the best result in the original topologies and in case of missing information. However, it turns out to be the most unstable when spurious links are added, since in two cases (Opsahl_9, Opsahl_11) it detects the whole network as a unique community (NMI = 0).

	Infomap	**Louvain**	**LGI-MCL ER**	**LGI-MCL RA**	**LGI-MCL EBC**	**MCL**
karate	0.54	0.49	0.72	0.73	0.72	**0.74**
opsahl 8	0.55	0.51	**0.56**	**0.56**	**0.56**	**0.56**
opsahl 9	**0.49**	0.42	0.38	0.39	0.39	0.41
opsahl 10	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
opsahl 11	**0.96**	**0.96**	0.90	0.82	0.79	0.63
polbooks	0.50	0.49	**0.57**	**0.57**	**0.57**	**0.57**
football	**0.92**	0.90	**0.92**	**0.92**	**0.92**	**0.92**
polblogs	0.51	**0.63**	0.00	0.00	0.00	0.00
mean NMI	**0.68**	**0.68**	0.63	0.62	0.62	0.60
mean ranking	**3.25**	4.00	3.56	3.31	3.63	**3.25**

Table 4. ***Community detection on real networks perturbed with random removal of links.*** *For each real network, 100 perturbed networks have been generated removing at random the 10% of links. The table reports the Normalized Mutual Information (NMI) computed between the ground-truth communities and the ones detected by every community detection algorithm for the 8 real networks, averaged over the 100 repetitions. NMI = 1 indicates a perfect match between the two partitions of the nodes. The methods are ranked by mean NMI over the dataset. The best result for each network as well as the best mean results are marked in bold.*

	Louvain	**LGI-MCL RA**	**LGI-MCL ER**	**LGI-MCL EBC**	**MCL**	**Infomap**
karate	0.45	**0.76**	0.75	0.70	0.68	0.53
opsahl 8	0.51	0.53	0.54	0.54	**0.55**	**0.55**
opsahl 9	**0.42**	0.39	0.38	0.40	0.41	0.00
opsahl 10	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**
opsahl 11	**0.96**	0.73	0.76	0.69	0.53	0.00
polbooks	0.49	**0.57**	**0.57**	**0.57**	**0.57**	0.50
football	0.90	**0.93**	**0.93**	0.92	0.92	0.92
polblogs	**0.41**	0.08	0.07	0.19	0.20	0.31
mean NMI	**0.64**	0.62	0.62	0.62	0.61	0.47
mean ranking	3.81	**3.19**	3.25	3.44	**3.19**	4.13

Table 5. ***Community detection on real networks perturbed with random addition of links.*** *For each real network, 100 perturbed networks have been generated adding at random the 10% of links. The table reports the Normalized Mutual Information (NMI) computed between the ground-truth communities and the ones detected by every community detection algorithm for the 8 real networks, averaged over the 100 repetitions. NMI = 1 indicates a perfect match between the two partitions of the nodes. The methods are ranked by mean NMI over the dataset. The best result for each network as well as the best mean results are marked in bold.*

9.4. Synthetic network analysis results

In order to provide additional and more detailed results regarding the behaviour of the clustering methods, a comparative test was performed on artificial networks produced by the nonuniform Popularity-Similarity-Optimization (nPSO) model [74], [75]. Indeed, the nPSO is an efficient generative model recently proposed to grow realistic complex networks, which not only are clustered, small-word, scale-free and rich-club, but also present communities whose number and size can be a priory defined (please refer to chapter 9.2. 'Synthetic networks generated by the nPSO model' for more details). These artificial networks with known community structure offer the ground-

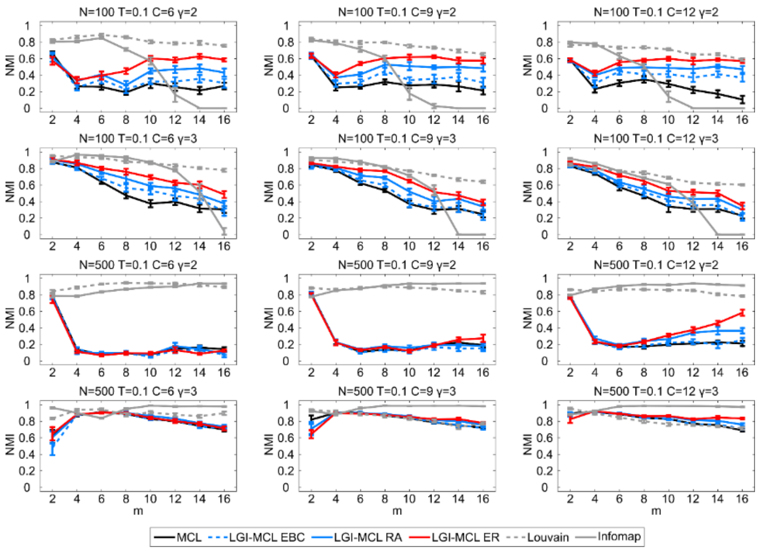

Figure 17. ***Community detection on nPSO networks (1st setting: T fixed; N, γ, m and C changing).*** *Synthetic networks have been generated using the nPSO model with parameters N = [100, 500] (network size) γ = [2, 3] (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = 0.1 (temperature, inversely related to the clustering coefficient) and C = [6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions and show that LGI-MCL, compared to MCL, significantly improves the performance for small N and low T, regardless of γ changes. Instead, for middle-size networks, this is mainly true for large C, large m and low γ.*

truth to build a valid benchmark to test the performance of algorithms for community detection.

The results of wide-range simulations (Figure 17-*Figure 20* and Appendix Figure A. 1-Figure A. *9*) - where synthetic networks were obtained by tuning several parameter combinations of the nPSO model - highlight similarities with respect to the results on real networks. First, LGI-MCL, compared to MCL, improves significantly the

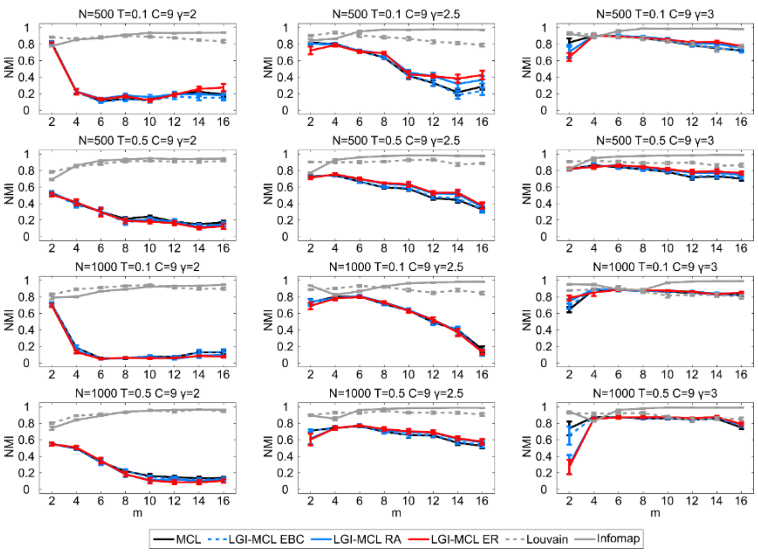

Figure 19. ***Community detection on nPSO networks (3rd setting: C fixed; N, γ, m and T changing)****. Synthetic networks have been generated using the nPSO model with parameters N = [500, 1000] (network size) γ = [2, 2.5, 3] (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.5] (temperature, inversely related to the clustering coefficient) and C = 9 (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions and show that MCL improves its performance with the increase of γ for middle and large size networks.*

community detection performance for small-size networks (N = 100) and high clustering (T = 0.1), regardless of γ changes. Instead, for middle-size networks (N = 500), this is mainly true when there are more communities (larger C), higher average degree (m) and $\gamma = 2$. The ranking of the performance of the LGI-MCL variants, from the highest to lowest, is generally LGI-MCL ER, LGI-MCL RA and LGI-MCL EBC (Figure 17), similarly to the real networks. Second, the performance of MCL increases and stabilizes with increasing network size (N) at $\gamma = 3$, independently from changes in temperature (T) and the number of communities (C), achieving performances close to the

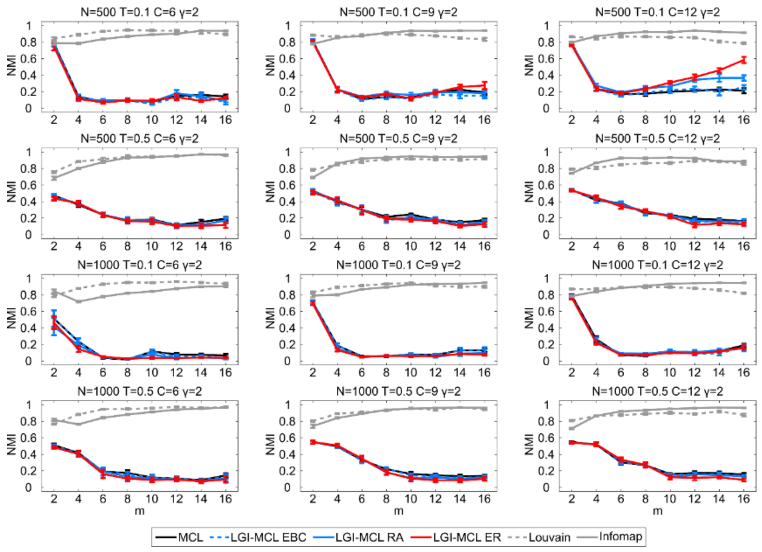

Figure 20. ***Community detection on nPSO networks (4th setting: γ fixed; N, m, T and C changing).*** *Synthetic networks have been generated using the nPSO model with parameters N = [500, 1000] (network size) γ = 2 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.5] (temperature, inversely related to the clustering coefficient) and C = [6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions and show that, at low γ, the MCL performance is close to state of the art methods for low m, whereas it decreases for higher m.*

state of the art algorithms Louvain and Infomap (Figure 18). In this parameter setting, it can be noticed that Infomap attains a slightly higher NMI than Louvain in several cases, but, on the other side, it drastically drops to NMI = 0 when the network is too dense (low N and high m), as already pointed out by the experiments of random link addition on real topologies. Third, MCL presents problems to correctly detect the communities in networks of middle (N = 500) and large (N = 1000) size at γ = 2, but improves and stabilizes the performance for increasing γ (Figure 19). An exception to this situation is found at a very low average degree (mostly m = 2) (Figure 20), where there is a peak of performance for the middle ($N = 500$) and large size ($N = 1000$) networks.

9.5. Advantages and limitations of LGI-MCL

The eight considered real networks represent a benchmark with ground-truth annotation generally adopted to test algorithms for non-overlapping community detection on real network topologies. However, the results here obtained suggest that this benchmark, collecting networks of different size (from tenths to thousands of nodes), seems enough complete and diversified to adequately investigate the performance of each method suggested in this work. In fact, LGI-MCL should offer better results than pure MCL, because the similarity pre-weighting is derived from dissimilarity measures that approximate a network geometry. This theoretical expectation is confirmed not only on the original real networks, but also when their topology is perturbed by noise simulated by random deletion of links (missing topological information) or random addition of links (spurious topological information), where the three LGI-MCL variants achieve a greater mean NMI than the unweighted MCL, corroborating the rationale on how to design similarity measures that favour the stochastic simulation procedure of MCL. On the other hand, when considering the synthetic networks as ground-truth benchmark, LGI-MCL clearly improves the performance compared to MCL in certain scenarios, mostly for small ($N = 100$) and medium ($N = 500$) size networks, whereas for large size networks ($N = 1000$) the improvement is often missing or less notorious.

Despite the improvements that LGI measures can bring to MCL, the method is still affected by certain types of network topologies. For example, in Figure 19, at low γ the MCL performance is dramatically reduced and far from state of the art. This can be explained because with lower γ there is a stronger presence of hubs, central nodes with a large degree acting as bridges between different regions of the network, which increases the likelihood for a random walk to move from one cluster to another one, and therefore makes more difficult for MCL to correctly infer the boundaries of the clusters. Similarly, the peak of MCL performance at low average degree (Figure 20) can be explained because the network topology is very sparse and therefore, it is less likely for a random walk to reach a hub and later move to another cluster. One goal of the wide experiments was indeed to point out the topological configurations affecting the MCL inference, so that further studies might investigate how to improve the performance in the presence of these structural patterns and make the method more robust.

10. Clustering analysis

In the next section, after describing the procedure behind MCL (please refer to chapter 3. 'Markov clustering' for more details), and recall a collection of network science notions, at the interface between network topology and network geometry [56], [59], [60], [67]–[69], with the purpose to design similarity measures to boost algorithms based on network navigability protocols (please refer to chapter 6. 'Minimum curvilinear Markov clustering' for more details), with respect to clustering analysis starts. Here, it is exhibited a performance comparison between nonlinear MCL with baseline and more advanced clustering algorithms such as classical MCL [32], AP [27], its nonlinear version MC-AP [57], density-based spatial clustering of applications with noise (DBSCAN) [22], density peaks shortest path-based (DPSP) [26], single linkage [18], [19], cciMST [31], K-means [16] and deep clustering-based algorithms [34] for the MNIST and CIFAR datasets (please refer to chapter 1. 'Clustering' for more details on the different clustering methods). They have been compared both on real and synthetic high-dimensional datasets and using different metrics (Accuracy, NMI and ARI, please refer to chapter 8. 'Evaluation framework' for more details) to evaluate their performances. Finally, a discussion with advantages and limitations of the MC-MCL approaches (including variants) will be addressed.

10.1. Clustering case studies and algorithm performance comparison.

Six different high-dimensional and nonlinear datasets were analyzed in order to perform a comparative analysis of the clustering methods.

Gastric mucosa microbiome

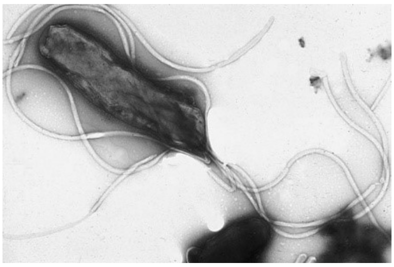
Figure 21. Electron micrograph of Helicobacter pylori bacterium. Picture from Professor Yukata Tsutsumi, Department of Pathology Fujita, Health University School of Medicine.

The dataset was generated by Paroni Sterbini and colleagues [76] and it consists of 24 biopsy specimens of the gastric antrum from 24 individuals who were referred to the Department of Gastroenterology of Gemelli Hospital (Rome) with dyspepsia symptoms (i.e. heartburn, nausea, epigastric pain and discomfort, bloating, and regurgitation). Twelve of these individuals had been taking PPIs (P) for at least 12 months, while the others were not being treated (naïve) or had stopped treatment at least 12 months before sample collection. In addition, 9 (5 treated and 4 untreated) were positive for H. pylori infection (Figure 21), where H. pylori positivity (H+) or negativity (H-) was determined by histology and rapid urease tests. The number of features is 187 and indicates different microbial abundance. The metagenomics sequence data were processed, replicating the bioinformatics workflow followed by Paroni Sterbini et al. [76], in order to obtain the dataset for the clustering algorithms. This dataset was analyzed for three clusters: H+ (n=5), H- (n=7) and P (n=12). The PPI (P) patients with (P&H+) and without (P&H-) the presence of H. pylori are considered a unique class, because it is known from previous studies [77], [78] that PPI significantly changes the gastric environment and covers the effect of other factors such as *H. pylori* presence. Furthermore, Durán et al. [51] evidenced in a recent study that, taken into account the dimensionality reduction and clustering analysis, the idea of three groups in the dataset seems more congruous than for the four groups case.

The data is publicly available in the NCBI Sequence Read Archive (SRA) (http://www.ncbi.nlm.nih.gov/sra, accession number SRP060417).

As commented in Chapter 3. 'Motivation', the dataset that implanted the idea for a nonlinear MCL was the gastric mucosa dataset from Sterbini et al. [76]. It all started from the need to express in numbers, what could be already appreciated by eyes in the segregation of groups with nonlinear patterns (Figure 13), a problem that persisted with different clustering techniques. Note that the clustering algorithms analyzed the datasets in the HD space directly, without considering the embeddings presented in Figure 13.

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
MC-MCL hdr	euc	SQRT	LOG	0.75	0.36	0.37	1.33
isoMCL	corr		LOG	0.75	0.33	0.31	2.00
MC-MCL hbr	corr	-	LOG	0.71	0.29	0.31	3.33
MC-MCL dual	corr	-	LOG	0.71	0.29	0.31	3.33
MC-MCL	corr	-	LOG	0.71	0.29	0.31	3.33
cciMST	corr		-	0.71	0.24	0.26	5.67
DBSCAN	corr		-	0.58	0.28	0.38	6.00
Kmeans	corr		LOG	0.67	0.20	0.26	7.33
MC-AP	corr		LOG	0.67	0.20	0.24	8.00
AP	corr		LOG	0.67	0.20	0.24	8.00
MCL	corr		LOG	0.67	0.19	0.21	9.67
DPSP	corr		-	0.54	0.15	0.21	11.67
Single linkage	euc		LOG	0.50	0.01	0.01	13.00

Table 6. ***Clustering performance in Gastric mucosa microbiome data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]), factor (for MC-MCL-variants) and*

normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are nonlinear MCL variants with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.

As theoretically expected, all MC-kernel variations clearly improve the performances of MCL and move it from one of the last places to leading positions, particularly for MC-MCL hdr variant, first in rank performance between all 13 methods (Table 6). Interestingly, the isomap inspired MCL also demonstrates good performances and is situated as the second-best clustering method. Remarkably, all MST-based methods, including cciMST and the MCL variants, achieve the highest mean rank using 5/6 of the highest places, demonstrating the successful MST property to approximate the hidden data geometry correctly.

DBSCAN seems to have problems in accuracy with a value rather low (0.58), but in ARI and NMI the situation differ, where its ARI value is close to the new MCL variants, and surprisingly outperform all the rest of the method in NMI (0.38), closely followed by the here proposed MC-MCL hdr (0.37). K-means, MCL, AP and MC-AP have low performances with accuracy of ~0.67, ARI of ~0.2 and NMI between 0.21 and 2.6. Surprisingly, Single linkage could not find the three proposed clusters having the lowest performance in the three measures. The improved density peaks version (DPSP) also had troubles being the second-lowest.

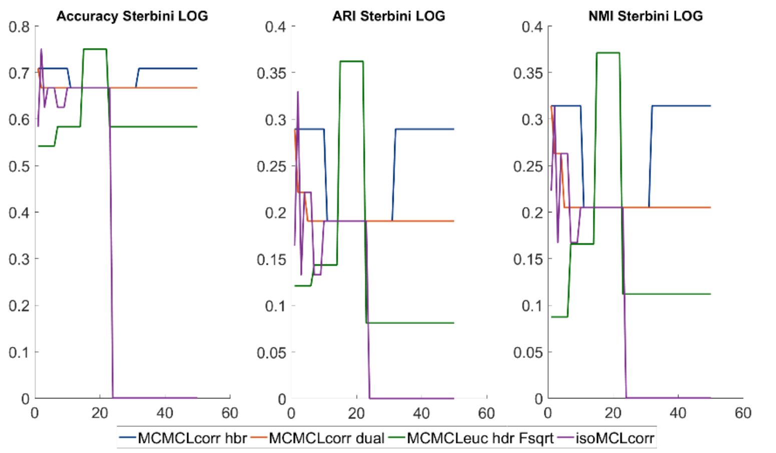

Figure 22. ***Parameter search for MC-MCL and isoMCL variants in gastric mucosa dataset with LOG normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

As mentioned in chapters 7.3 'Minimum curvilinear Markov clustering multi-MST variants' and 7.4 'Isomap-inspired MCL', the different proposed MCL algorithms introduce the need to select parameters. For each variant, 50 values of their respective parameters were searched, and the best results were placed in the respective table performance comparison (Table 6). The parameter search simulation for the gastric mucosa dataset can be found in Figure 22 for the LOG normalization (Note that one plot is generated for each normalization [LOG and no normalization], just one of both plots, LOG normalized, is here presented, whereas the no normalized can be found in the Appendix section Figure A. 10). Consider that the x-axis represents a different unit depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the *k* value to create the proximity graph.

Regarding the best distance and factor applied, the best performing MCL variants were compared to evince the parameter influence on their performances. Depending of the parameter value, a great fluctuation exists in the MCL variant performances (Figure 22), being the MC-MCL hdr the most notorious (green line), where from 0.55 in accuracy with 5% percentile, the value can rise to 0.75 with ~20% percentile. A similar trend also occurs in the case of the ARI and NMI measures.

This important performance fluctuation could be partially explained by the gastric mucosa dataset reduced sample size with only 24 observations. Removing a small portion of samples can be translated into a great MST topological change, i.e. by connecting distant regions in the graph.

Radar signal

The data is composed of 350 radar signals targeting free electrons in the ionosphere, where each radar signal consisted of 34 features that are measurements of electromagnetic pulses. It was collected by the Space Physics Group of the Johns Hopkins University Applied Physics Laboratory [79]. The two groups are defined as: (1) 225 good radar

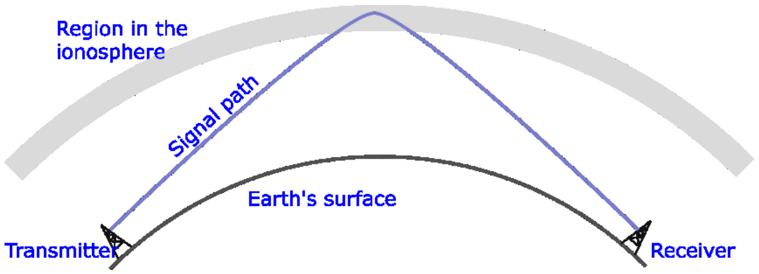

Figure 23. ***Radar signal illustration.*** *Good radar signals returned evidence of free electrons from the ionosphere, whereas bad radar signals passed through the ionosphere and returned noise.*

signals, characterized by those signals that returned evidence of free electrons in the ionosphere, and (2) 125 bad radar signals which were those signals that passed through the ionosphere and returned background noise (Figure 23). Hence, good radar signals are similar, and bad radar signals might be dissimilar.

Table 7 exhibits the performance comparison between the clustering algorithms for this dataset. MC kernel shows an improvement in performance compared to its linear algorithm MCL, being the dual variant the highest performer obtaining the first place in mean ranking with an accuracy of 0.78, ARI of 0.29 and NMI of 0.32, followed by the hbr variant with performances of 0.80, 0.35 and 0.25 in accuracy, ARI and NMI respectively (Table 7).

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
MC-MCL dual	spea	LOG	-	0.78	0.29	0.32	1.67
MC-MCL hbr	euc	SQRT	-	0.80	0.35	0.25	2.00
MC-MCL hdr	spea	LOG	-	0.77	0.25	0.28	2.67
isoMCL	spea		-	0.77	0.25	0.28	2.67
Kmeans	euc		-	0.71	0.18	0.13	5.33
AP	euc		-	0.71	0.17	0.13	5.67
MC-MCL	euc	LOG	-	0.71	0.17	0.12	6.33
DBSCAN	corr		-	0.68	0.10	0.14	7.67
MC-AP	euc		-	0.69	0.14	0.09	8.33
DPSP	corr		-	0.65	0.02	0.03	10.67
MCL	euc		-	0.60	0.04	0.06	11.00
cciMST	euc		-	0.64	0.00	0.01	11.67
Single linkage	euc		-	0.64	0.00	0.01	11.67

*Table 7. **Clustering performance in Radar (two clusters) data**. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are nonlinear MCL variants with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.*

Curiously, Kmeans and AP perform comparable and even slightly better than the MC-MCL algorithm using the regular MC kernel. The MC-based nonlinear version of AP, MCAP, performed lower than its counterpart; and DPSP, MCL, cciMST and single linkage could not effectively find the two clusters and assign the sample majority to one unique class. Note that cciMST, although based in the MST alike the MC-MCL variant, was rather far away from their performance, suggesting that the MST alone does not always correctly approximate the hidden geometrical data space. This is evidenced as well by the fact that the 'multiple MST' MC-MCL variants outperform the MC-MCL employing the original MC kernel.

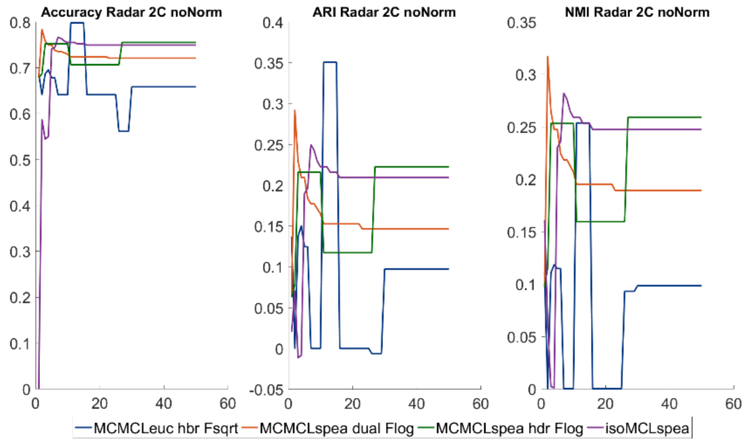

Figure 24. ***Parameter search for MC-MCL and isoMCL variants in Radar dataset (2 clusters) without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

The parameter search simulation for the Radar dataset can be found in Figure 24 without normalization application. The performances across parameters seem more stable in the accuracy measurement compared to ARI and NMI, whose values greatly fluctuate from 0 to ~0.30. The most evident fluctuation is exhibited by the variant MC-MCL hbr, which achieves the highest accuracy (0.80) and ARI (0.35) values and is only outperformed in NMI by the rest of the nonlinear-MCL variants. A curious trend can be appreciated for both MC-MCL dual and isoMCL, where their pick performances are achieved with a low parameter value (number of MST duals to construct and *k* number of neighbors for the proximity graph construction respectively) and then are rapidly decreased until arriving to a plateau.

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
MC-MCL hdr	spea	LOG	-	0.74	0.35	0.32	2.00
MC-MCL dual	spea	-	-	0.75	0.34	0.30	2.33
MC-MCL hbr	spea	-	-	0.74	0.34	0.32	2.33
MC-MCL	corr	SQRT	-	0.74	0.27	0.38	3.00
isoMCL	spea		-	0.75	0.32	0.28	3.33
AP	euc		-	0.66	0.23	0.25	6.00
Kmeans	euc		-	0.62	0.16	0.15	8.33
cciMST	euc		-	0.62	0.15	0.14	9.00
MC-AP	corr		-	0.56	0.08	0.22	9.33
DPSP	corr		-	0.65	0.02	0.03	9.67
DBSCAN	corr		-	0.64	0.01	0.02	10.67
Single linkage	euc		-	0.64	0.01	0.02	10.67
MCL	euc		-	0.42	0.03	0.05	11.00

Table 8. ***Clustering performance in Radar (three clusters) data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are nonlinear MCL variants with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.*

In the study of Cannistraci et al. [53], it was suggested that, actually, the bad radar signals might be segregated into two different groups (given by the result of a nonlinear dimensionality reduction embedding). Therefore, the dataset is additionally analyzed for three clusters (Table 8).

From this analysis, the method that benefits the most from this new grouping is the MC-MCL algorithm with original MC, and it is moved from a 7th place to a 4th just after the MC-MCL variants using multiple MSTs in their kernels. It increases its performance from 0.71, 0.17 and 0.12 to 0.74, 0.27 and 0.38 in accuracy, ARI and NMI, respectively, achieving the highest NMI compared to all other methods. Overall, this

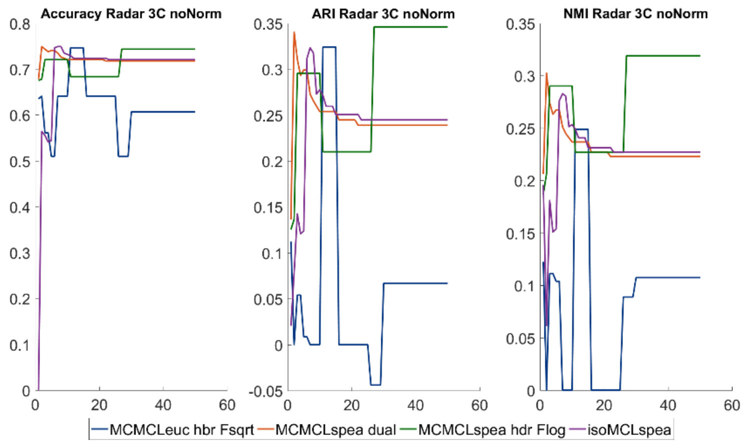

Figure 25. ***Parameter search for MC-MCL and isoMCL variants in Radar dataset (3 clusters) without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

new grouping seems to negatively affect the accuracy of the best-performing methods (MC-MCL multi MST variants), and curiously increases the ARI and NMI performances for the majority of the clustering methods, with few exceptions. A clear exception is DBSCAN, which is evidently negatively affected by this grouping, decreasing its performance from 0.68, 0.10 and 0.14 to 0.64, 0.01 and 0.02 in accuracy, ARI and NMI, respectively (Table 8). AP still outperforms its nonlinear counterpart on the three measures, and DPSP, single linkage, and MCL join DBSCAN with their low performances.

The parameter search simulation for the Radar dataset with three clusters can be found in Figure 25 without normalization application. The line trends here are certainly similar to Radar's case with two clusters, being an exception the noticeable boost of ARI and NMI for the MC-MCL hdr (green) variant after the ~30% percentile.

As for the suggestion of Cannistraci and colleagues [53] about the adequate grouping of the radar dataset, the idea seems to be congruous due to the general increase in ARI and NMI measures across the clustering methods. Nonetheless, the change in performance is not so notorious when comparing the best-performing methods in each grouping case (the MC-MCL variants), and therefore an affirmation that the radar dataset should be analyzed by a three grouping problem rather than a two one is still not strongly supported.

Tripartite-Swiss-Roll

In order to 'objectively' (using a ground truth) test how the clustering algorithms could detect nonlinear relationships, we additionally performed an analysis on the Tripartite-Swiss-Roll dataset (Figure 26): an artificial dataset characterized by evident nonlinear patterns and generated as a discretization of the manifold associated to a Swiss-Roll function [62] in a three-dimensional (3D) space. Indeed, it is a synthetic dataset composed by 723 points obtained as the partition in three sections of a discrete Swiss-Roll manifold depicted in three-dimensional space [62]. It reproduces the typical nonlinearity (given by the Swiss-Roll shape) and the discontinuity (given by the tripartition of the manifold, and therefore three clusters), that might be often hidden in the multidimensional representation of data samples. However, it is important to clarify that this dataset, contrarily to all the other ones used in this chapter, has significantly fewer features than

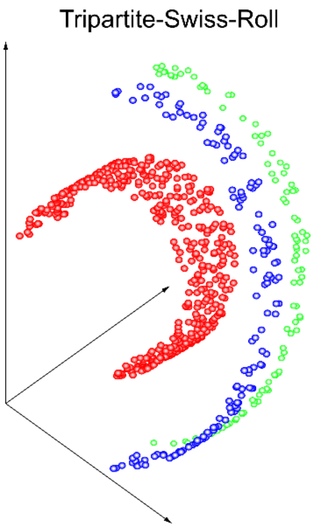

Figure 26. Tripartite-Swiss-Roll scatter plot evidencing the three nonlinear shaped clusters.

samples. Therefore it cannot be considered a multidimensional dataset. Yet, it is a very useful benchmark for nonlinear clustering.

The Tripartite-Swiss-Roll possesses three main features: (1) it has a clear nonlinear shape; (2) each cluster is clearly separated (not fuzzy) from the neighbour clusters; and (3) each cluster is dense.

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
MC-MCL dual	euc	-	-	1.00	1.00	1.00	1.00
MC-MCL hbr	euc	-	-	1.00	1.00	1.00	1.00
MC-MCL hdr	euc	-	-	1.00	1.00	1.00	1.00
isoMCL	euc		-	1.00	1.00	1.00	1.00
MC-MCL	euc	-	-	1.00	1.00	1.00	1.00
DBSCAN	euc		-	1.00	1.00	1.00	1.00
MCL	euc		-	1.00	1.00	1.00	1.00
cciMST	euc		-	1.00	1.00	1.00	1.00
Single linkage	euc		-	1.00	1.00	1.00	1.00
DPSP	euc		-	0.85	0.87	0.82	10.00
MC-AP	euc		-	0.64	0.47	0.58	11.00
Kmeans	euc		-	0.56	0.10	0.20	12.00
AP	euc		-	0.54	0.09	0.19	13.00

*Table 9. **Clustering performance in Tripartite-Swiss-Roll data**. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are nonlinear MCL variants with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.*

The performances of each algorithm are shown in Table 9. Many clustering algorithms are able to find the three clusters as indicated by the perfect performance segregations value of 1 in the three measures. These methods are MCL, isoMCL, all MC-MCL variants, DBSCAN, cciMST and Single linkage. As theoretical expected, the linear techniques AP and Kmeans cannot detect the nonlinear patterns and perform poorly in this dataset, obtaining the last positions. Surprisingly, DPSP and MC-AP, although outperforming the linear algorithms AP and Kmenas, are not able to successfully find the three clusters.

This synthetic dataset is the only case in the present study where, in the presence of a nonlinear clustering structure, classical MCL can achieve comparable performance to MC-MCL (Table 9). Indeed, in all the three real datasets previously analyzed, MCL was one of the worst algorithms between the 13 different types tested. These findings, on one side, suggest the utility to adopt synthetic data because yet on this example, linear clustering algorithms such as AP and K-means result, as theoretically expected, the worst. On the other side, the same results suggest that simple synthetic datasets with many samples and few dimensions, although they are an interesting and useful benchmark, might be too 'naïvely' designed. They might miss other crucial aspects of data nonlinearity that emerge in the case of curse of dimensionality[7]. Altogether, after this didactic example, we can conclude that it is important to expose the tested algorithms to different data scenarios in which nonlinearity emerges from different data sources.

In the case of parameter search for the nonlinear MCL variants (Figure 27), independently from the parameter to choose, all algorithms achieve the perfect segregation with values of one in all measures, except when isoMCL presents a multiple component graph (at low *k*).

[7] Course of dimensionality refers to when the number of features is substantially larger than the number of samples.

Till now, the tests were made in unsupervised recognition of nonlinear patterns that emerge from metagenomics, radar signal and synthetic backgrounds, but always the scenario of a few numbers of expected clusters was considered. It is now time to confront these algorithms on a more challenging benchmark, commonly applied in artificial intelligence for supervised and unsupervised tasks, to test their nonlinear pattern recognition performance.

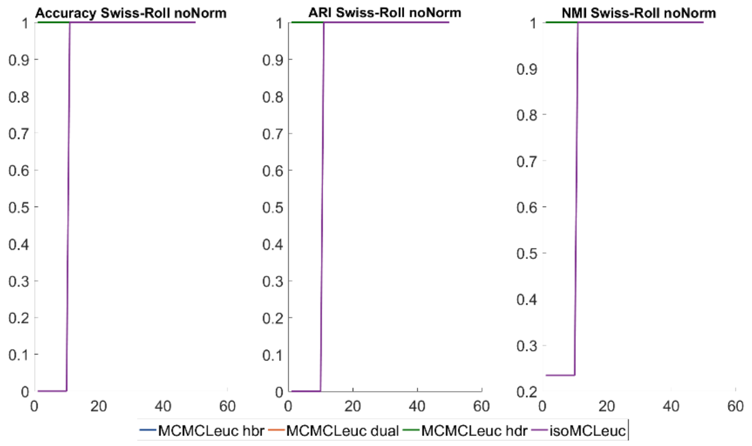

Figure 27. ***Parameter search for MC-MCL and isoMCL variants in Tripartite-Swiss-Roll dataset without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

MNIST

MNIST [80] is one of the most used datasets in machine learning. This is a large dataset that consists of 28x28 pixel images of handwritten digits. Every image can be thought of as a 784-dimensional array, where each value represents each pixel's intensity in a gray scale. The different sample groups are numbers between 0 and 9, for a total of 10 clusters (Figure 28). Since this is a very large dataset (70000 samples), three datasets were constructed from it. First, 300 samples were randomly selected for each digit, resulting in a sub-dataset with 3000 samples. Secondly, the MNIST test side (10000 samples) from the Kaggle competition was used. Finally, the full MNIST data was also used for testing the performances of the clustering algorithms. Therefore, three MNIST data employed are composed of a total of 3000, 10000 and 70000 samples, 784 features and 10 groups.

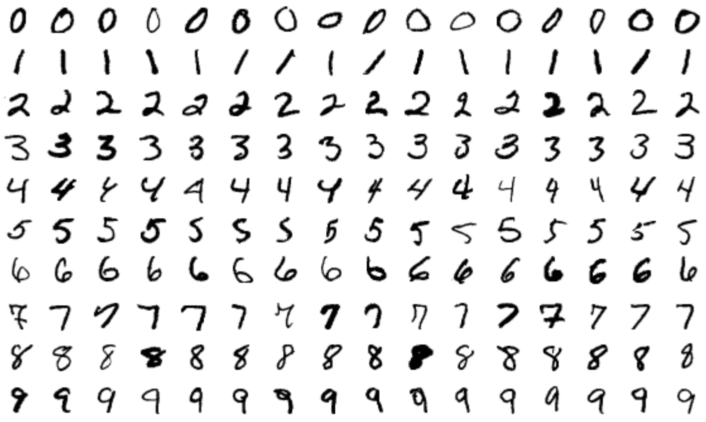

Figure 28. Sample images from the MNIST dataset. By Josef Steppan - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=64810040.

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
MC-MCL hdr	spea	LOG	-	0.82	0.74	0.81	1.67
MC-MCL dual	euc	-	LOG	0.85	0.73	0.79	1.67
isoMCL	euc		LOG	0.85	0.73	0.78	2.00
MC-MCL hbr	corr	LOG	LOG	0.79	0.69	0.78	3.67
MC-MCL	euc	-	LOG	0.75	0.62	0.70	5.33
MC-AP	euc		LOG	0.75	0.58	0.68	6.00
cciMST	corr		LOG	0.69	0.57	0.72	6.33
Kmeans	corr		LOG	0.57	0.40	0.53	8.33
AP	corr		LOG	0.56	0.39	0.51	9.33
MCL	corr		LOG	0.48	0.25	0.55	9.33
DPSP	euc		-	0.27	0.08	0.31	11.33
DBSCAN	corr		LOG	0.26	0.00	0.43	11.67
Single linkage	euc		-	0.10	0.00	0.01	12.67

Table 10. ***Clustering performance in MNIST 3000 data.*** *Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are nonlinear MCL variants with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.*

The evaluation on the first MNIST dataset with 3000 samples exhibits a great improvement of the MC-MCL and isoMCL compared to the original method, where the performances rose from 0.48, 0.25 and 0.55 to values greater than 0.80, 0.70 and 0.75 in accuracy, ARI and NMI respectively. Interestingly, and leaving aside isoMCL, all MC variants (including the MC-AP) perform better than the non-MC. From the non-MC-based algorithms, cciMST is the better performing, followed by Kmeans, AP and MCL. On the other hand, DPSP, DBSCAN and Single linkage proportioned lacklustre performances.

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
isoMCL	euc		LOG	0.91	0.82	0.84	1.67
MC-MCL dual	corr	-	-	0.86	0.81	0.86	2.00
MC-MCL hbr	corr	LOG	-	0.85	0.80	0.85	3.33
MC-MCL hdr	corr	-	-	0.85	0.79	0.84	4.00
HOE-CNN	cos		-	0.91	0.76	0.78	4.00
MC-MCL	euc	-	-	0.86	0.73	0.77	5.33
cciMST	corr		-	0.80	0.71	0.80	7.67
HOT	corr		-	0.82	0.65	0.69	8.33333
HOMO	euc		-	0.82	0.65	0.68	8.66667
HOE	cos		-	0.82	0.65	0.68	8.66667
HIT	cit		-	0.82	0.65	0.68	8.66667
MC-AP	corr		-	0.75	0.69	0.77	9.00
Kmeans	corr		LOG	0.57	0.42	0.55	13.33
MCL	euc		LOG	0.54	0.38	0.56	13.67
AP	corr		-	0.41	0.33	0.53	15.33
DPSP	spea		-	0.31	0.31	0.54	15.67
DBSCAN	euc		-	0.11	0.00	0.00	17.00
Single linkage	euc		-	0.11	0.00	0.00	17.00

Table 11. ***Clustering performance in MNIST test data.*** *Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance (Best dist) approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]; only in case of deep clustering algorithms: cosine distance [cos] and cityblock distance [cit]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are algorithms with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.*

From now on, starting from MNIST test and for the subsequent datasets to consider, deep clustering algorithms are included in the comparison tables as the state-of-the-art algorithms. The inclusion of these methods was not possible for smaller datasets due to the small size problem and complications concerning the code provided by the authors [34]. Deep networks, such as the deep autoencoder used in the methods from Peng et al. [34], usually need thousands of samples to successfully being trained in nontrivial scenarios (the algorithm is not trying to separate simply black from white images). Workarounds can be applied in a small-size data scenario, i.e. data augmentation[8]. However, for the sake of clustering methods comparison under 'real' circumstances, the deep clustering algorithms are applied starting from this point.

Results for MNIST test in Table 11 demonstrated that the new nonlinear MCL variants could outperform even 'complex' state-of-art deep-clustering algorithms. Taking a step back, in the study of Peng and colleagues, a deep clustering algorithm based on autoencoder and invariances sample assignments was presented (refer to chapter 1.2.5. 'Deep-based methods' for more details). The authors mentioned that their baseline algorithms (HOMO, HIT, HOT and HOE) could be further improved by changing the deep autoencoder architecture with other types of layers, i.e. by using convolutional (CNN) layers instead of fully connected ones; and further demonstrated the improvement in performance for the MNIST data (HOE-CNN). CNN layers are designed to follow vision processing from the visual cortex of living organisms, thus being aid for its application on image analysis [81]. Therefore, its use in the autoencoder and posterior performance improvement made sense for the study of Peng et al. On account of it, the outperforming values of the nonlinear MCL variants over the CNN deep clustering algorithm variant, aid for image datasets, is a nontrivial achievement accomplished purely by network geometry and navigability theory. The improvement of MC-MCL and isoMCL over the deep-clustering methods is completely conferred to network theory, considering that original MCL achieves low performances, with

[8] Data augmentation refers to the generation of new data samples, referred as to latent data, from known samples [97].

accuracy (0.54), ARI (0.38) and NMI (0.56) even under Kmeans values (0.57, 0.42 and 0.55 in accuracy, ARI and NMI respectively).

isoMCL and MC-MCL dual are the best performing methods (Table 11) followed by MC-MCL hbr, MC-MCL hdr, HOE-CNN and MC-MCL. Interestingly, cciMST achieves a better mean ranking than the deep-clustering methods with fully connected layers. On the other hand, MC-AP improves in performance compared to its linear version by an important extent. Finally, DPSP, DBSCAN and single linkage are the worst-performing methods.

The parameter search simulation for the MNIST test dataset can be found in Figure 29 without normalization application (The rest of the parameter search plots for MNIST datasets can be found in the Appendix section Figure A. 11-Figure A. 13, including the figure where isoMCL achieves its maximum values from Table 11). As appreciated,

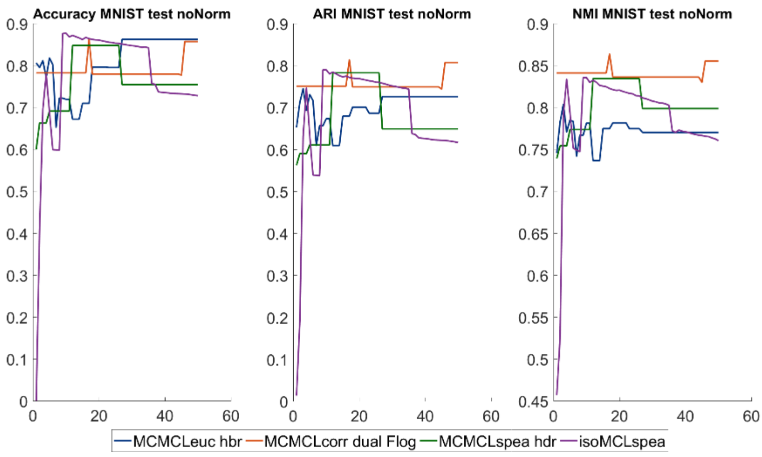

Figure 29. ***Parameter search for MC-MCL and isoMCL variants in MNIST test dataset without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

the maximum possible values of the different variances are close to each other. MC-MCL dual is the method with less performance fluctuation achieving less than 0.1 units of difference between the lowest and highest performances for the three measures, demonstrating to be a stabile algorithm. All other methods fluctuate more in performance, but generally with strong outcomes (values above 0.7 in most of the cases).

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
MC-MCL dual	corr	-	-	0.91	0.89	0.90	2.00
MC-MCL hbr	corr	LOG	-	0.92	0.89	0.89	2.00
isoMCL	euc		LOG	0.94	0.88	0.88	2.67
MC-MCL hdr	corr	-	-	0.90	0.87	0.89	3.67
HOE-CNN	cos		-	0.93	0.82	0.86	4.00
HOE	cos		-	0.87	0.74	0.76	6.67
cciMST	corr		-	0.81	0.78	0.83	7.33
HOMO	euc		-	0.86	0.72	0.74	7.67
HOT	corr		-	0.86	0.72	0.74	7.67
HIT	cit		-	0.86	0.72	0.74	7.67
MC-MCL	euc	-	-	0.60	0.58	0.74	10.33
MCL	euc		LOG	0.66	0.51	0.67	11.67
Kmeans	corr		LOG	0.56	0.40	0.53	13.00
DBSCAN	euc		-	0.11	0.00	0.00	14.00
Single linkage	euc		-	0.11	0.00	0.00	14.00

*Table 12. **Clustering performance in MNIST full data.** Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]; only in case of deep clustering algorithms: cosine distance [cos] and cityblock distance [cit]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are algorithms with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.*

In the case of the full MNIST dataset (Table 12), most of the algorithm's performances are improved, with the exception of

DBSCAN and Single linkage, which are not able to figure out the patterns inside it. Note that the algorithms AP, MCAP and DPSP do not appear here due to errors thrown by their respective codes (issues with the code's implementation). Once again, almost all nonlinear MCL variants outperform in a great fashion their linear counterpart MCL, and even the CNN-based deep clustering HOE-CNN, where accuracies are close to one another but the ARI and NMI performances are much stronger within the here presented algorithms. The exception comes with MC-MCL with the original kernel that slightly increased the performance of MCL and even decreased a bit in accuracy. Curiously the cciMST algorithm achieves as well a noticeable performance, and although it does not outperform HOE-CNN, it does compared to the deep-clusterings based on fully connected layers HOMO, HOT and HIT in the mean rank.

Due to the size of this dataset, and time constraints, there was no evaluation on parameter search for the MC-MCL multi-MST variants and isoMCL. The parameters selected for them were the same as for the best parameters found in the MNIST test dataset. Therefore, take into considerations that the performances of Table 12 might not be the final and even better performances can be obtained.

CIFAR

CIFAR [82], alike MNIST, is a widely employed dataset for artificial intelligence benchmarks composed of 'tiny' colour images of 32x32 pixels. They are a labelled subset of the '80 million tiny images' dataset collected by Krizhevsky, Nair and Hinton (and pulled offline during 2020 for 'teaching AI systems to use racist, misogynistic slurs'). From these 80 million tiny images, two datasets, namely CIFAR10 and CIFAR100 were extracted.

The CIFAR10 dataset consists of 60000 images and 10 classes with 6000 images each class. The classes encapsulate images of: airplanes, automobiles, birds, cats, deers, dogs, frogs, horses, ships and trucks

(Figure 30). Regularly, in AI benchmarks, this dataset is divided into a training set (50000 images) and a test set (10000 images). Here, the test batch was used for clustering analysis, which contains 1000 images for each of the ten classes. Note that the CIFAR10 images are here in grey scale.

On the other hand, CIFAR100 is a dataset that contains 100 classes, with 600 images each class, which can be categorized into 20 'super classes'. Here, just one superclass is utilized termed 'aquatic mammals', which naturally consists of aquatic mammals images from 5 different classes: beaver, dolphin, otter, seal and whale. As such, this dataset comprises a total of 3000 colour images, 600 for each of the five classes.

Both datasets are challenging for cluster algorithms, demonstrated by the low performances that the clustering methods achieve. In the case

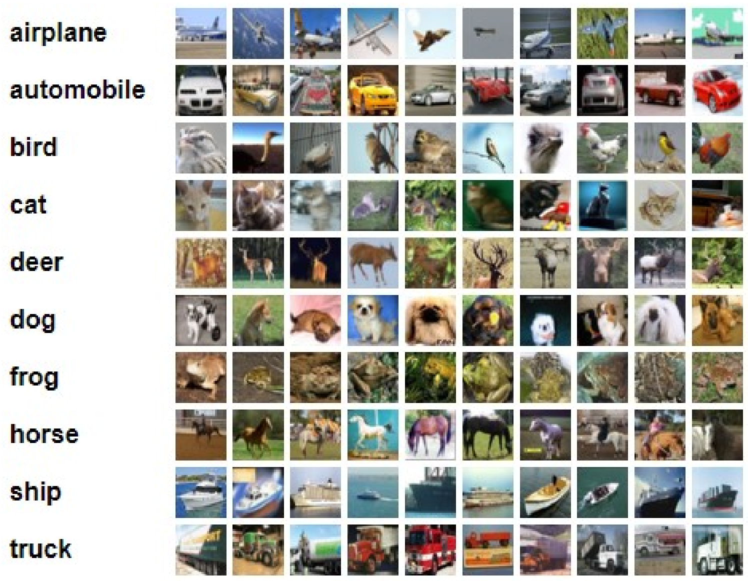

Figure 30. Subset of the CIFAR10 image dataset. Image extracted from Krizhevsky webpage: https://www.cs.toronto.edu/~kriz/cifar.html

of CIFAR100 (Table 13), the best performing methods are the deep clustering. Particularly, HOE obtains the highest values with 0.36, 0.12, and 0.14 in accuracy, ARI and NMI respectively. Despite the difficulty of this dataset, MC and isoMCL variants still improve their performance in comparison with MCL. Remarkably, AP obtains better performance than its nonlinear counterpart MC-AP, and together with Kmeans get the 6th and 7th position respectively out of 17 methods. From the MC-MCL variants, the hdr version is the closest to state-of-art performance. Oppositely, DBSCAN, MCL, DPSP, and Single linkage are the methods that present more troubles in assigning class memberships to samples, and achieve ARI values of 0 and NMI values close to 0.

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
HOE	cos		-	0.36	0.12	0.14	1.666667
HOT	corr		-	0.36	0.11	0.14	2.333333
HOMO	euc		-	0.35	0.11	0.14	3.333333
HIT	cit		-	0.35	0.11	0.14	3.333333
MC-MCL hdr	corr	-	LOG	0.36	0.11	0.13	3.67
AP	spea		-	0.34	0.13	0.15	3.67
Kmeans	corr		-	0.35	0.12	0.13	4.00
MC-MCL hbr	corr	-	LOG	0.35	0.10	0.11	6.67
MC-MCL	corr	SQRT	LOG	0.35	0.08	0.10	8.00
MC-AP	spea		-	0.34	0.09	0.10	9.00
MC-MCL dual	euc	LOG	LOG	0.33	0.09	0.10	9.67
isoMCL	corr		-	0.30	0.06	0.07	12.00
cciMST	corr		-	0.22	0.00	0.02	13.00
DBSCAN	corr		LOG	0.20	0.00	0.02	13.67
MCL	euc		LOG	0.21	0.00	0.01	14.00
DPSP	euc		-	0.20	0.00	0.01	14.33
Single linkage	euc		-	0.20	0.00	0.00	15.00

Table 13. ***Clustering performance in CIFAR100 data.*** *Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previously mentioned measures) are reported for each clustering method together with*

the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]; only in case of deep clustering algorithms: cosine distance [cos] and cityblock distance [cit]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are algorithms with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because it is an initialization parameter that all methods require.

It seems that CIFAR10 is even more challenging than CIFAR100 by the general poor performances from all clustering algorithm. The main reason might be the increase in the number of samples to assign, from 5 in CIFAR100 to 10 in CIFAR10. As in CIFAR100, the top-performing methods are the deep clustering algorithms achieving a mean rank of 1 for the four of them, meaning equal performance across deep-clustering method independently from the distance used in the model (Table 14).

Methods	Best dist	Factor	Norm	Acc	ARI	NMI	Mean rank
HOE	cos		-	0.22	0.04	0.07	1.00
HOT	corr		-	0.22	0.04	0.07	1.00
HOMO	euc		-	0.22	0.04	0.07	1.00
HIT	cit		-	0.22	0.04	0.07	1.00
isoMCL	corr		LOG	0.18	0.04	0.07	4.33
Kmeans	euc		-	0.20	0.03	0.07	4.67
MC-MCL dual	euc	LOG	-	0.18	0.04	0.06	6.33
AP	euc		-	0.20	0.03	0.06	6.67
MCL	corr		LOG	0.17	0.04	0.06	7.00
MC-MCL hdr	euc	LOG	-	0.19	0.03	0.05	8.67
MC-MCL hbr	euc	LOG	-	0.19	0.03	0.05	8.67
MC-MCL	euc	LOG	-	0.19	0.03	0.05	8.67
MC-AP	euc		LOG	0.19	0.03	0.05	8.67
cciMST	corr		-	0.14	0.02	0.03	14.00
DBSCAN	euc		-	0.10	0.00	0.00	15.00
DPSP	euc		-	0.10	0.00	0.00	15.00
Single linkage	euc		-	0.10	0.00	0.00	15.00

Table 14. ***Clustering performance in CIFAR10 data.*** *Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), and mean rank (according to the previous mentioned measures) are reported for each clustering method together with the best distance approach (Pearson correlation [corr], Spearman correlation [spea] or Euclidean [euc]; only in case of deep clustering algorithms: cosine distance [cos] and cityblock distance [cit]), factor (for MC-MCL-variants) and normalization (Norm) applied. The methods are sorted by mean rank from the highest (top) to the lowest (bottom) performance. Methods in red are algorithms with one hyperparameter to optimize. Note that the parameter(s) to consider the number of clusters is/are not taken into account because is an initialization parameter that all methods require.*

Curiously, MCL achieves better ranking performance than its nonlinear variants except for MC-MCL dual and isoMCL. AP shows once again to perform better than its MC-AP in mean rank (Table 14). Kmeans performs competitively as well in this dataset achieving a 6th position, and DBSCAN, DPSP and single linkage have troubles trying to assign class memberships by interpreting the CIFAR10 dataset as one unique cluster, explaining the 0 values in ARI and NMI.

Leaving aside the values of mean ranking, the majority of algorithms perform really close to each other and with low performances, making this dataset the most difficult to cluster from all data up to this point. As MNIST, CIFAR consists of tiny images, although the patterns inside both data seem to differ greatly in simplicity. Black and white numbers may offer clearer patterns to be analyzed than objects/animal images. Moreover, while MNIST backgrounds are black, different colours (for CIFAR100) and shapes can be found in CIFAR backgrounds.

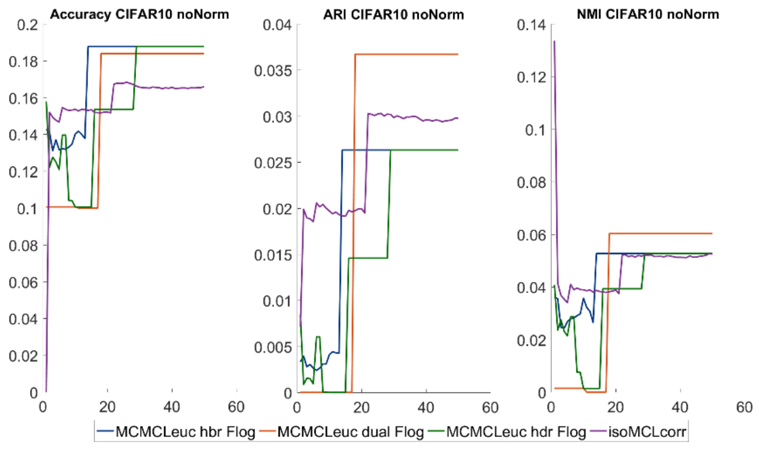

Figure 31. ***Parameter search for MC-MCL and isoMCL variants in CIFAR10 dataset without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

The parameter search simulation for the CIFAR10 test dataset can be found in Figure 31 without normalization application (The rest of the parameter search plots for CIFAR datasets can be found in the Appendix section Figure A. 14-Figure A. 16, including the plot where isoMCL achieves its maximum values from Table 14). A curiously different trend is observed in the CIFAR10 line plot (Figure 31), where all nonlinear MCL variants start improving their performances with higher parameter values (previously, there was a diverse performance line trend depending on the clustering variant and the dataset analyzed). Nevertheless, the performance values are maintained at a low level for all algorithm variants, and the threshold of 0.20, 0.05 and 0.10 is never reached for accuracy, ARI and NMI, respectively.

10.2. General clustering performance – an overview

This section provides the performance overview for all clustering algorithm across all datasets by a summary table of the NMI measure (Table 15), accuracy and ARI summary tables can be found in the appendix section (Table A. 1 & Table A. 2).

It is clearly appreciated that in general, the nonlinear MCL versions (red), namely MC-MCL (M2-M5) and isoMCL (M1), improve the performance of classical MCL (green)(M6) in all datasets and turns MCL into one of the best clustering methods for nonlinear data among the compared algorithms. In general, all nonlinear MCL variants perform similarly. However, MC-MCL with original MC kernel (M5) tends to perform lower, as exhibited in the Radar dataset with two clusters (D2), and MNIST datasets (D5, D8 and D9), this trend presents an exception on the Radar with three clusters (D3), where MC-MCL (M5) outperforms all the rest of the methods with an NMI value of 0.38. The use of the MC- or iso- kernel improve to a great degree the MCL performance outperforming even 'complicated' algorithms such as the deep clustering methods (M14 – M18) in the MNIST datasets (D8, D9), with one particular deep-clustering variant tailored for the analysis of MNIST data given by its autoencoder architecture (M18).

NMI	D1	D2	D3	D4	D5	D6	D7	D8	D9
M1	0.31	0.28	0.28	1.00	0.78	0.07	0.07	0.84	0.88
M2	0.31	0.32	0.30	1.00	0.79	0.10	0.06	0.86	0.90
M3	0.31	0.25	0.32	1.00	0.78	0.11	0.05	0.85	0.89
M4	0.37	0.28	0.32	1.00	0.81	0.13	0.05	0.84	0.89
M5	0.31	0.12	0.38	1.00	0.70	0.10	0.05	0.77	0.74
M6	0.21	0.06	0.05	1.00	0.55	0.01	0.06	0.56	0.67
M7	0.26	0.01	0.14	1.00	0.72	0.02	0.03	0.80	0.83
M8	0.24	0.09	0.22	0.58	0.68	0.10	0.05	0.77	-
M9	0.24	0.13	0.25	0.19	0.51	0.15	0.06	0.53	-

M10	0.21	0.03	0.03	0.82	0.31	0.01	0.00	0.54	-
M11	0.38	0.14	0.02	1.00	0.43	0.02	0.00	0.00	0.00
M12	0.26	0.13	0.15	0.20	0.53	0.13	0.07	0.55	0.53
M13	0.01	0.01	0.02	1.00	0.01	0.00	0.00	0.00	0.00
M14	-	-	-	-	-	0.14	0.07	0.69	0.74
M15	-	-	-	-	-	0.14	0.07	0.68	0.74
M16	-	-	-	-	-	0.14	0.07	0.68	0.76
M17	-	-	-	-	-	0.14	0.07	0.68	0.74
M18	-	-	-	-	-	-	-	0.78	0.86

*Table 15. **NMI performance summary of all clustering methods M across all datasets D.** Methods list: M1: isoMCL; M2:MC-MCL dual; M3: MC-MCL hbr; M4: MC-MCL hdr; M5: MC-MCL; M6: MCL; M7: cciMST; M8: MC-AP; M9: AP; M10: DPSP; M11: DBSCAN; M12: Kmeans; M13: Single linkage; M14: HOT; M15: HOMO; M16: HOE; M17: HIT; M18: HOE-CNN. Data list: D1: Gastric mucosa; D2: Radar 2C; D3: Radar 3C; D4: Tripartite-Swiss-Roll; D5 MNIST3000; D6: CIFAR100; D7: CIFAR10; D8: MNIST test; D9: MNIST full. Nonlinear MCL variants are marked with red, whereas the original MCL algorithm is marked in green.*

The algorithm cciMST (M7), based in the MST network as the MC-MCL variants, perform relatively good in several datasets with mean ranking usually after the here proposed nonlinear MCL methods, it even outperformed the deep-clustering methods (M14-M18) on the MNIST data (D8 and D9) (Table 15). However, this method evidenced some troubles for the Radar 2 clusters (D2) and CIFAR (D6, D7) data.

Furthermore, considering if MC can as well improve the performance of other clustering algorithms, such as AP (M9), with MC-AP (M8), the results do not display a clear improvement with the use of the MC kernel with the exception of the particular cases of the Tripartite-Swiss-Roll (D4) and the MNIST datasets (D5 and D8). Although, in the Tripartite-Swiss-Roll dataset, MC-AP was not able to find perfectly the three clusters (only 4 methods from 13 failed to do so). Notice that both AP and MC-AP also had troubles in computing clustering for the MNIST full data, this time regarding their code implementation; an issue that persisted with the method DPSP (M10).

In the case of the density-based clustering, DPSP (M10) and DBSCAN (M11) did not show outstanding performances, with the exception of DBSCAN in the gastric mucosa dataset (D1), where it achieves the highest NMI. Curiously, DBSCAN reduced its performances significantly in the MNIST datasets from the tiniest MNIST3000 (D5), to the biggest ones (D8 and D9), triggered by the nontrivial task of its two-parameter search under fuzzy clusters. A method that proved to be ineffective in its generic form is the recent proposed DPSP (M10), with low performances across all datasets. As aforementioned, it had troubles with the MNIST full (D9) cluster membership computation, and it was not able to retrieve the three clusters from the Tripartite-Swiss-Roll dataset (D4). This unsuccessful pattern under the analyzed datasets could be related to the fact that the density peaks were automatically selected by the algorithm, but with manually selected density peaks, the algorithm might perform better in the clustering task.

Kmeans (M12), on the other side, as theoretically expected, perform low in these nonlinear datasets. Nonetheless, it presents comparable NMI values to the best-performing methods for both CIFAR data (D6 and D7). These results, far from impressive, highlight the onerous patterns from the CIFAR datasets, where a possible explanation to the poor clustering performances - and different from MNIST - are the complex shapes of objects looked from different perspectives and distances, added to the different shapes and colour tones from the backgrounds. One class image, i.e. dogs, can vary tremendously noise-wise in its background, if the image was taken on a beach, mountains or at home; patterns very different from each other and difficult to discern with unsupervised tasks.

For the hierarchical clustering method, Single linkage (M13), all NMI performances were close to 0, proving the inability of this algorithm for these dataset types, with the exception of the Tripartite-Swiss-Roll (D4), where it was able to perfectly assign all class memberships, achieving a 1 in NMI performance.

Regarding the last type of algorithms, the state-of-art methods using deep-clustering (M14-M17) evidenced strong performances in CIFAR (D6 and D7), having the highest performances (Table 15), yet achieving this podium with low values. With respect to MNIST (D8 and D9), the same methods presented strong performances, outperformed by its deep-clustering cousin with the convolutional variant (M18). However, these techniques performed lower than the nonlinear MCL methods.

Lastly, addressing the time complexity of each MC-MCL variant, most of them are similar and need the generation of one or two networks (MST or proximity-based) that later are unified for the kernel construction. An exception is MC-MCL dual, which could unify many dual MSTs, depending on the input parameter value (number of dual MSTs to construct). In this work, 50 was the maximum value tried, and by analyzing the results on different datasets, a general trend was observed. Usually, rather a low input value was needed to obtain already high performances, as displayed in Figure 32 (All other dataset plots can be found in the appendix Figure A. 17-Figure A. 28). Meaning

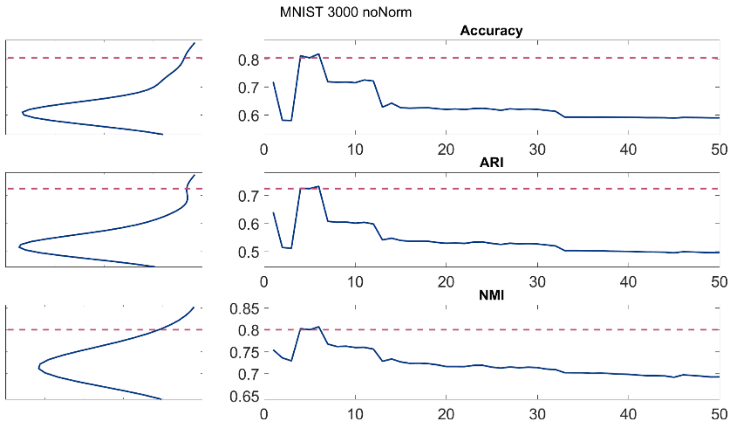

Figure 32. ***Illustration of the peak performance zone on MNIST3000.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

that it is regularly enough to use a few dual MSTs for the kernel computation, which already obtain high significant performances when comparing with a more (computationally) expensive kernel. A summary of parameter analysis, including all nonlinear MCL variants, can be found in Figure A. 29.

10.3. Advantages and limitations of MC-MCL and isoMCL

The nine considered datasets represent a benchmark with ground-truth annotation, including generally adopted data (CIFAR and MNIST) to test algorithms for clustering tasks. These are considered to be enough complete and diverse to adequately investigate the performance in nonlinear problems of each method suggested. In fact, nonlinear MCL should offer better results than pure MCL, because the similarity kernel is derived from dissimilarity distances that approximate a network geometry, either by use of the MST, or the proximity graph (refer to chapters 7.3. 'Minimum curvilinear Markov clustering multi-MST variants' and 7.4. 'Isomap-inspired Markov clustering' for more information). This theoretical expectation is confirmed on all datasets, where the five nonlinear MCL variants achieve a greater mean rank than the kernel-less MCL, corroborating the rationale on how to design similarity kernels that favour the stochastic simulation procedure of MCL.

MC-MCL and isoMCL demonstrated clear superior performance not only over MCL, but overall the here presented clustering algorithms, achieving always leading mean rank positions in these complex nonlinear scenarios.

Notwithstanding the big improvements reached by these MCL variants, reduced performances could be appreciated in the CIFAR datasets. Despite close to the state-of-art performance, the engineering network geometry and navigability improvement for the MCL random walk was not enough to catch the complex patterns behind the CIFAR datasets. As above-mentioned in chapter 10.2. ‘General clustering performance – an overview’, CIFARs objects plains, and different backgrounds (acting like noise) make it difficult for MCL to catch certain patterns from the object to be clustered, and such patterns do not simply emerge through the kernels here constructed (by means of MSTs or proximity networks). Although this seems to be a limitation tangent to all clustering methods, the issue does not persist in the supervised scenario, where several algorithms achieve close to perfect classification in CIFAR10 and CIFAR100 [83]–[89]. Therefore, more strategies regarding these limitations could pinpoint to a general improvement in clustering performance not only in CIFAR, but also in many real datasets.

Part IV. CONCLUSION

In conclusion, this dissertation introduces a rationale on how to design similarity measures for MCL, which try to approximate the hidden geometry of the manifold that generates the data network topology, either in the framework of community detection or clustering tasks. For the community detection scenario, since the hidden geometry of many real complex networks is hyperbolic and tree-like [90], [91], its congruous approximation can favour the stochastic simulation procedure of MCL. The empirical and numerical results provided in this work support the rationale, and the derived similarity measures EBC, RA, and ER seem to boost MCL both in real and synthetic networks.

On the other hand, following the idea that many real complex data follow a tree-like structure, its approximation, inspired by network geometry and navigability, and derived from multidimensional datasets through the MC and Isomap inspired network-based similarity kernels, is supported by the important boost given to MCL in many nonlinear and real multidimensional data scenarios. Such improvements could outperform, by an important gap, even 'complicated' deep-clustering-based algorithms in AI benchmark datasets like MNIST, where the architecture of the deep algorithm (autoencoder) is tailored exclusively for that type of data. On the contrary, the here proposed nonlinear kernels can work in a comprehensive list of datatypes, and no need for changes is required to be directly applied in different datasets.

In the case of CIFAR, the proposed methods could perform close to state-of-art methods, although a collective clustering-wise issue is appreciated, where all performance values, through all evaluation measurements, are rather small. A possible solution to this, inspired by the deep-clustering methods, could be to embed the data through a certain technique, prior to clustering membership assignments (like an autoencoder or a dimensionality reduction algorithm). The new data coordinates could hypothetically lead to prior segregation between classes, that clustering algorithms could easily catch, as already demonstrated in the study of Peng et al. [34]. Under this circumstance, a deep learning approach could enhance greatly the performances of the here proposed methods, with the drawback of designing different deep architectures depending on the dataset to be analyzed, adding the additional 'stress' of data augmentation in the case of small-size datasets, if necessary.

Regarding the nonlinear MCL variants for the multidimensional datasets, remarkably, the methods that better perform were the MC-multi-MSTs and Isomap-inspired kernels. This improvement over the original MC kernel could be explained by the increase in the local connectivity of the MST to a point where it can alleviate paths with high traffic (central nodes connecting the network), but avoiding 'multiple possible paths' from one point to another (many edges). Indeed, the parameter values that obtain the highest performances in the case of the MC-MCL dual and isoMCL variants are, in general, rather low.

To summarize, network geometry was already shown to facilitate greedy routing [61], [92], and affinity propagation [61], and to the best of the collected knowledge in this dissertation, this is the first time that the strategy is applied to better guide random-walk (stochastic flow) based simulations. Therefore, the results displayed in this dissertation provide further confirmation that network geometry can be adopted to make information flow processes more efficient and therefore pave the way for the generalized understanding of the impact of network geometry on algorithms based on network navigability.

APPENDIX

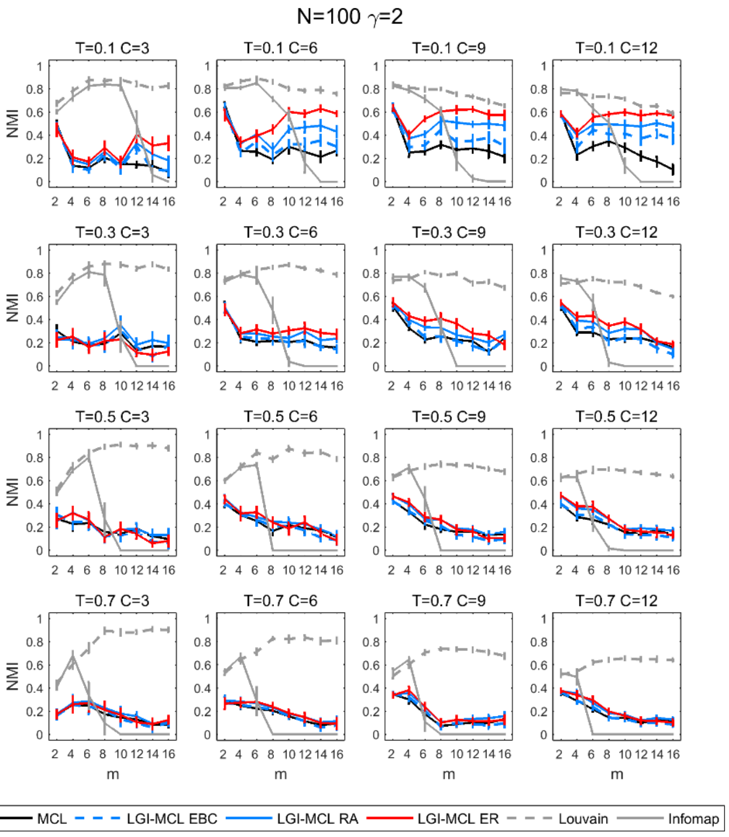

Figure A. 1. ***Community detection on nPSO networks: N = 100 and γ = 2.*** *Synthetic networks have been generated using the nPSO model with parameters N = 100 (network size) γ = 2 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information*

(NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.

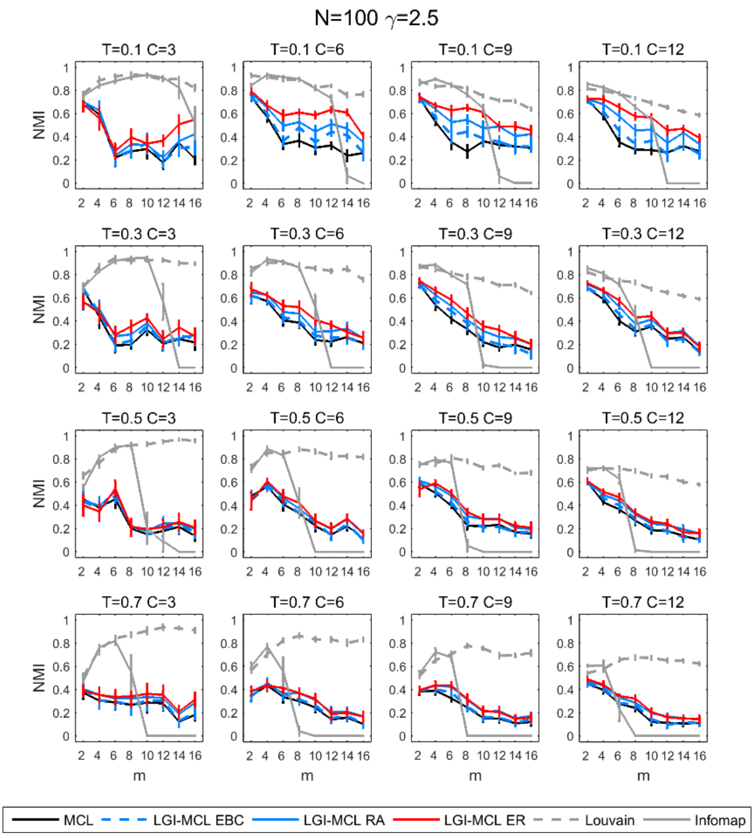

Figure A. 2. ***Community detection on nPSO networks: N = 100 and γ = 2.5.*** *Synthetic networks have been generated using the nPSO model with parameters N = 100 (network size) γ = 2.5 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

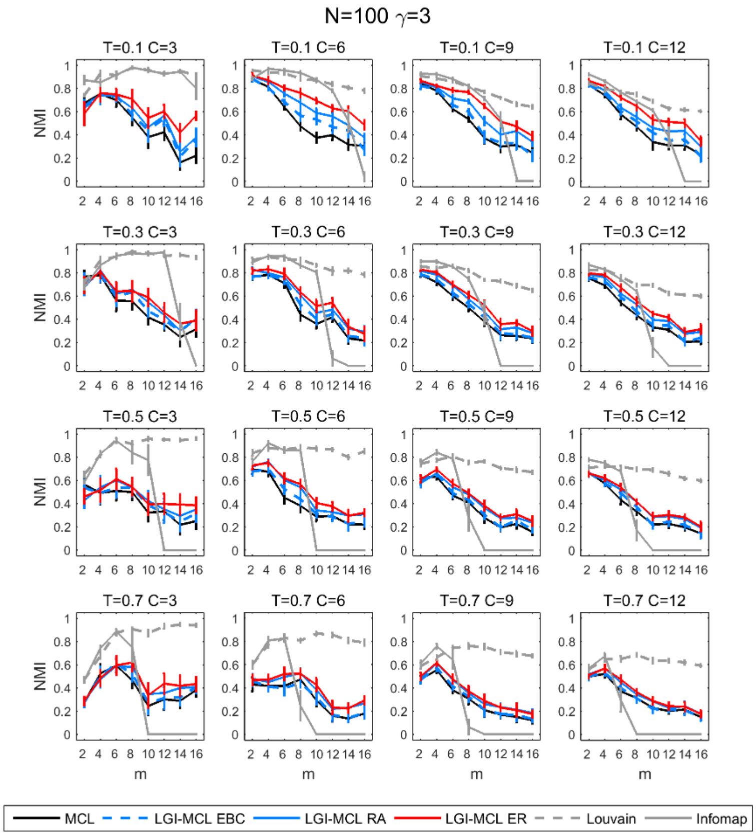

Figure A. 3. ***Community detection on nPSO networks: N = 100 and γ = 3****. Synthetic networks have been generated using the nPSO model with parameters N = 100 (network size) γ = 3 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

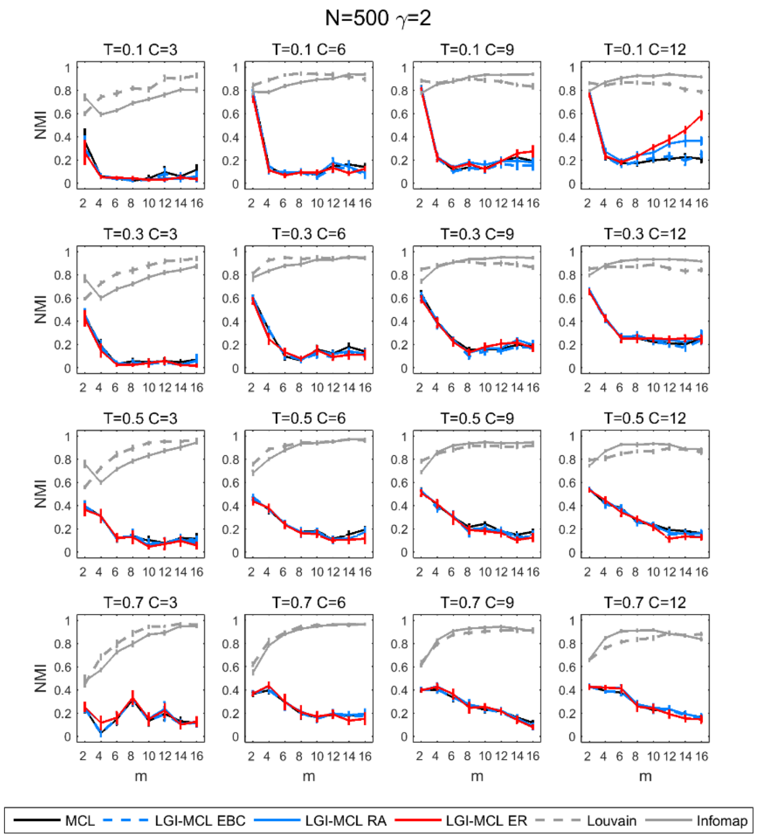

Figure A. 4. ***Community detection on nPSO networks: N = 500 and γ = 2****. Synthetic networks have been generated using the nPSO model with parameters N = 500 (network size) γ = 2 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

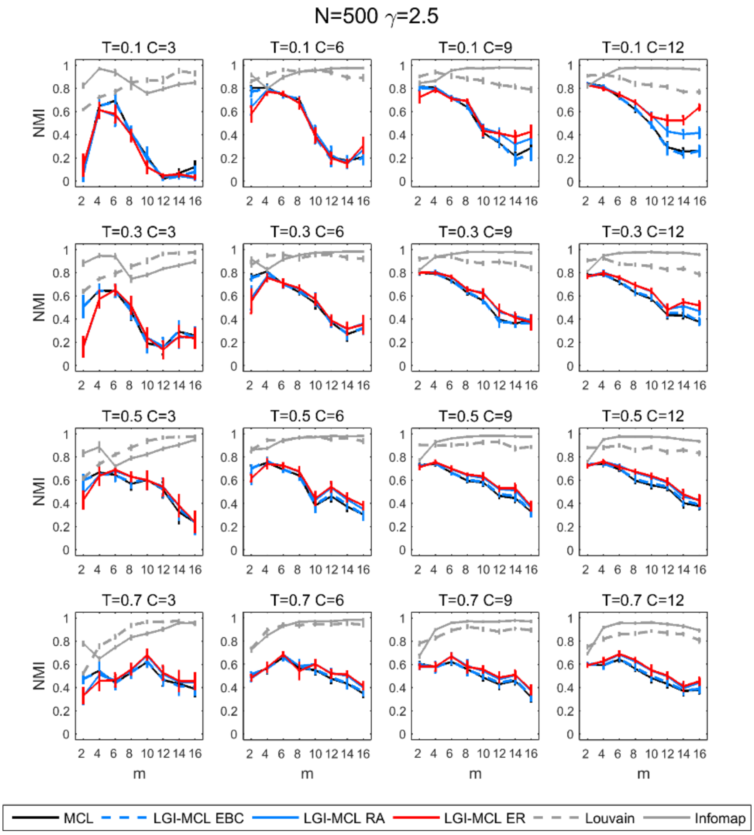

Figure A. 5. ***Community detection on nPSO networks: N = 500 and γ = 2.5.*** *Synthetic networks have been generated using the nPSO model with parameters N = 500 (network size) γ = 2.5 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

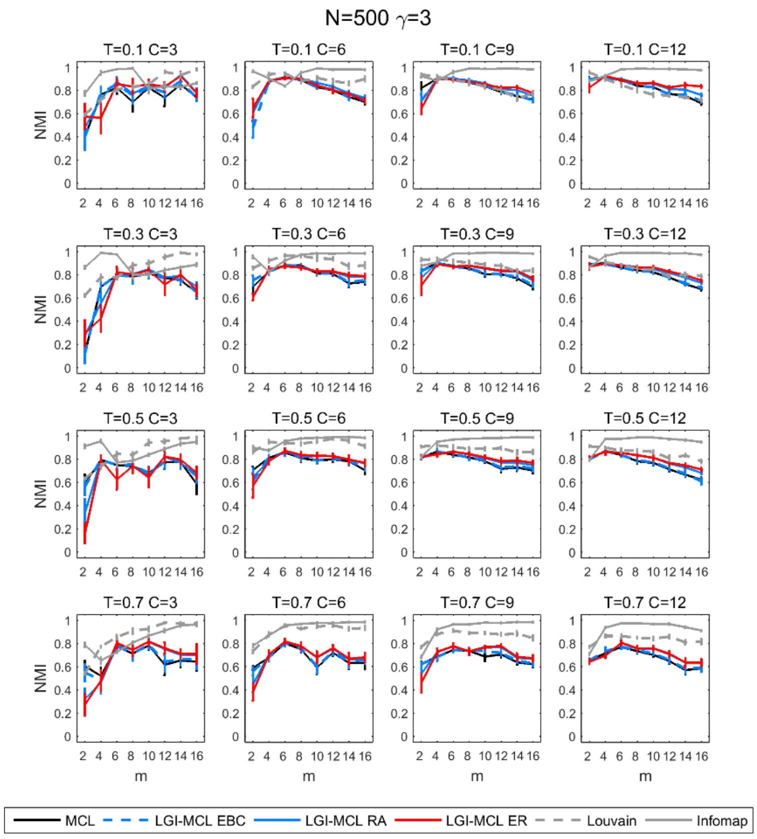

Figure A. 6. ***Community detection on nPSO networks: N = 500 and γ = 3***. *Synthetic networks have been generated using the nPSO model with parameters N = 500 (network size) γ = 3 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

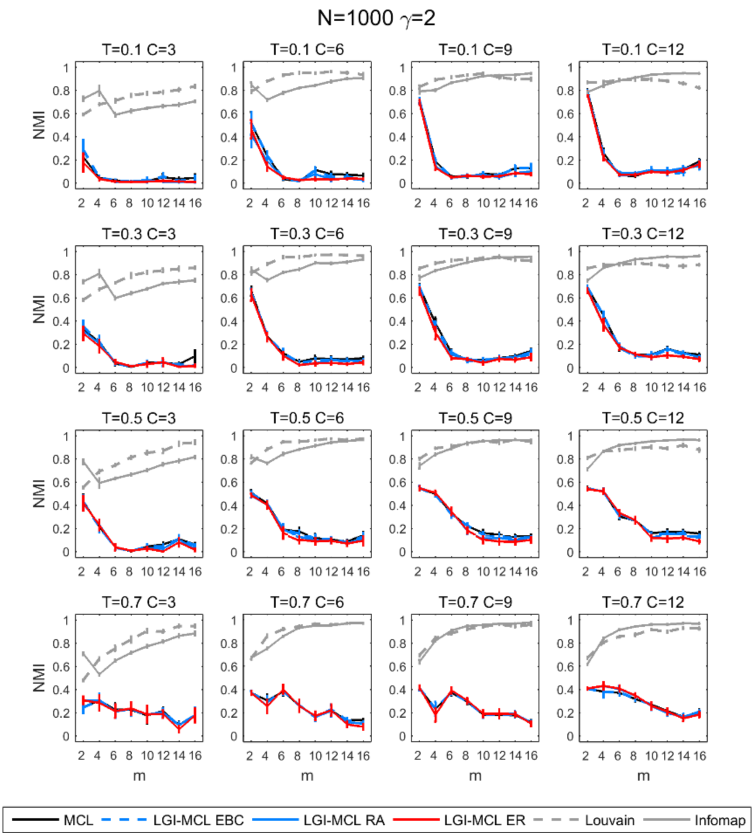

Figure A. 7. ***Community detection on nPSO networks: N = 1000 and γ = 2.*** *Synthetic networks have been generated using the nPSO model with parameters N = 1000 (network size) γ = 2 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

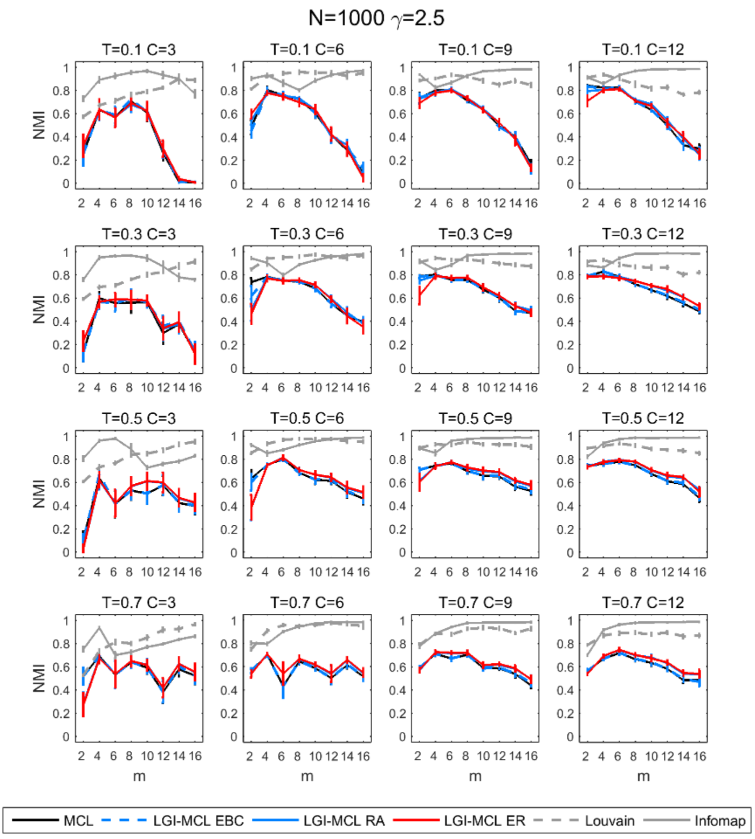

Figure A. 8. ***Community detection on nPSO networks: N = 1000 and γ = 2.5.*** *Synthetic networks have been generated using the nPSO model with parameters N = 1000 (network size) γ = 2.5 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

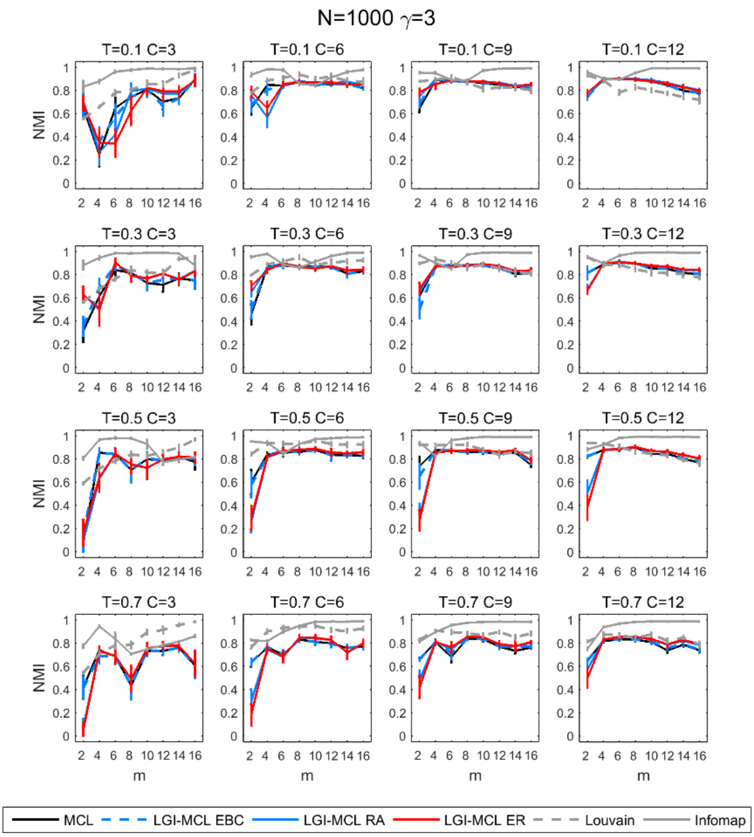

Figure A. 9. ***Community detection on nPSO networks: N = 1000 and γ = 3.*** *Synthetic networks have been generated using the nPSO model with parameters N = 1000 (network size) γ = 3 (power-law degree distribution exponent), m = [2, 4, 6, 8, 10, 12, 14, 16] (half of average degree), T = [0.1, 0.3, 0.5, 0.7] (temperature, inversely related to the clustering coefficient) and C = [3, 6, 9, 12] (number of communities). For each combination of parameters, 10 networks have been generated. For each network the community detection methods have been executed and the communities detected have been compared to the annotated ones computing the Normalized Mutual Information (NMI). The plots report for each parameter combination the mean NMI and standard error over the random repetitions.*

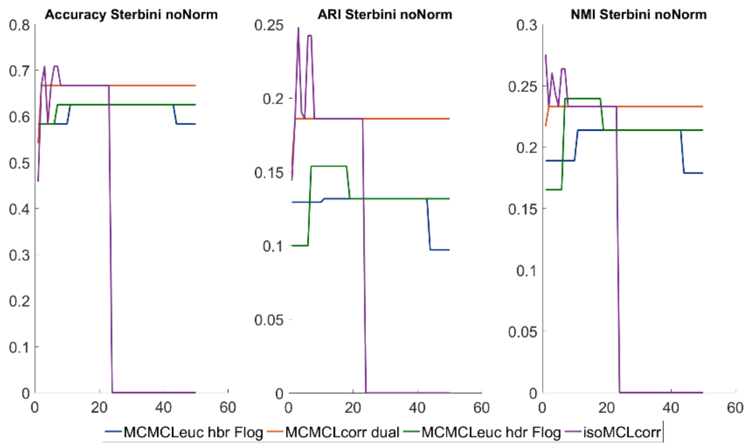

Figure A. 10. ***Parameter search for MC-MCL and isoMCL variants in gastric mucosa dataset without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

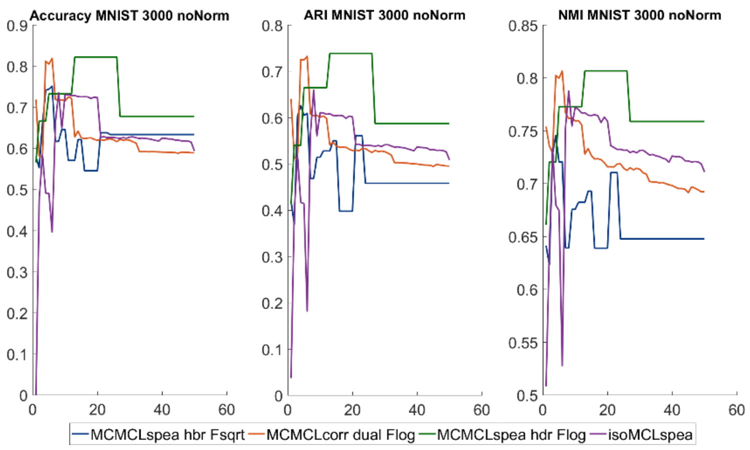

Figure A. 11. ***Parameter search for MC-MCL and isoMCL variants in MNIST 3000 dataset without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

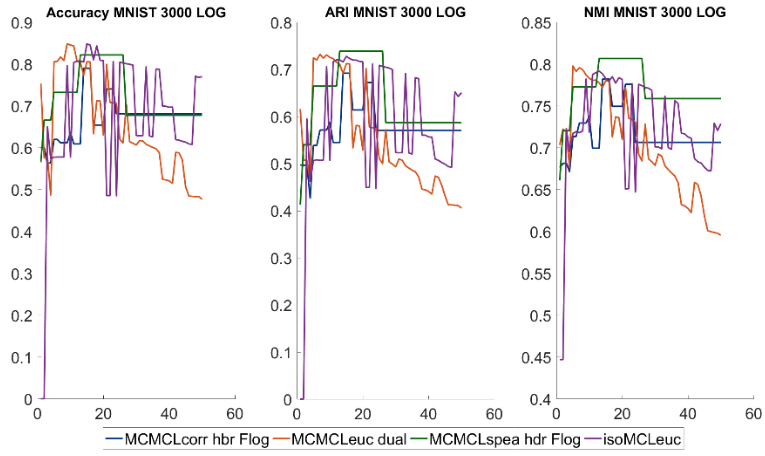

Figure A. 12. ***Parameter search for MC-MCL and isoMCL variants in MNIST 3000 dataset with LOG normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

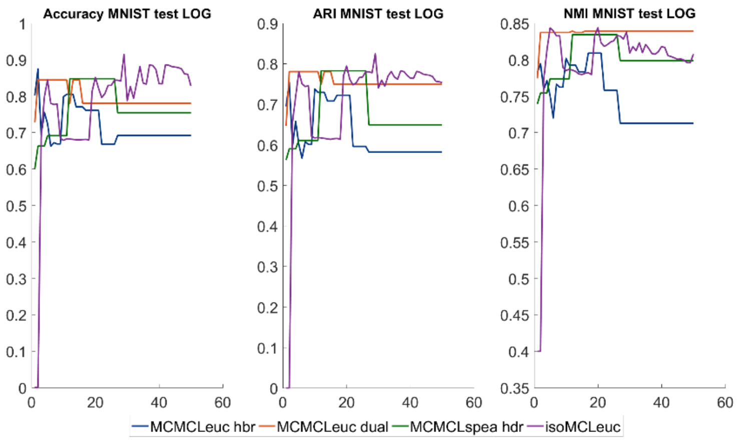

Figure A. 13. ***Parameter search for MC-MCL and isoMCL variants in MNIST test dataset with LOG normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

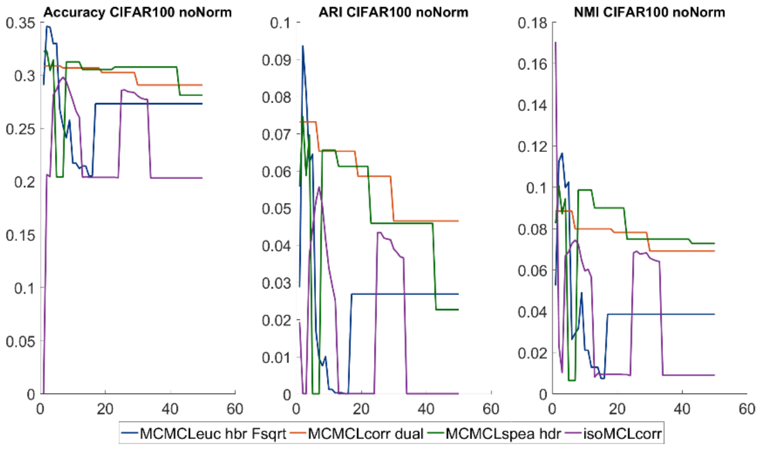

Figure A. 14. ***Parameter search for MC-MCL and isoMCL variants in CIFAR100 dataset without normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

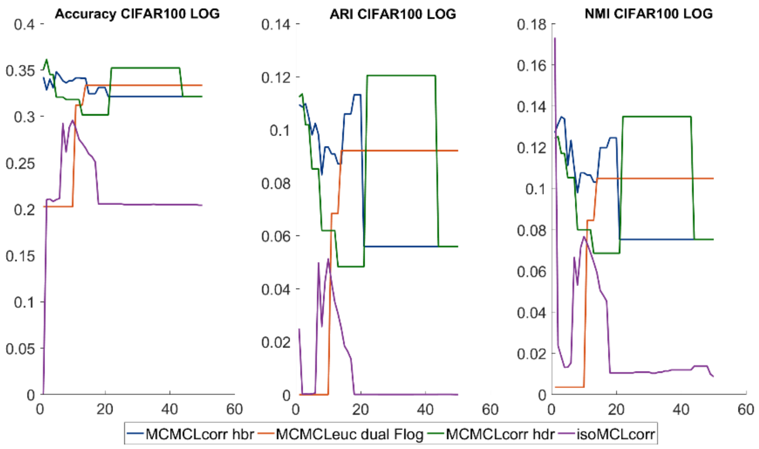

Figure A. 15. ***Parameter search for MC-MCL and isoMCL variants in CIFAR100 dataset with LOG normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

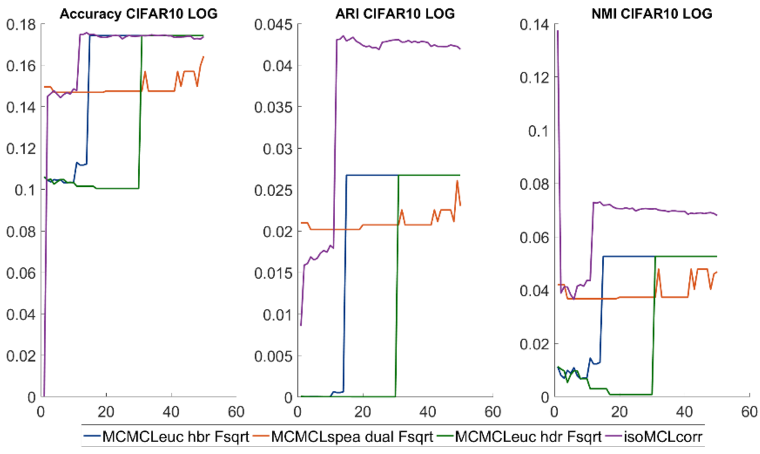

Figure A. 16. ***Parameter search for MC-MCL and isoMCL variants in CIFAR10 dataset with LOG normalization.*** *Accuracy, ARI and NMI performances. The x axis (from 1 to 50) represents different units depending on the MCL variant: for MC-MCL hbr (blue) and hdr (green), is the percentile of high betweenness centrality/degree nodes to be removed for the computation of the second MST. For MC-MCL dual (orange) it corresponds to the number of dual MSTs to construct and ensemble with the original MST. For isoMCL (purple) it consists in the k value to create the proximity graph.*

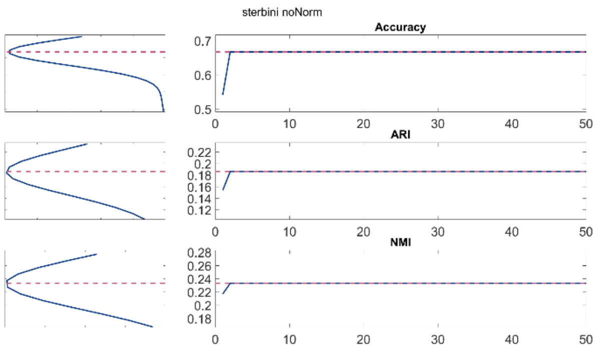

Figure A. 17. ***Illustration of the peak performance zone on gastric mucosa dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

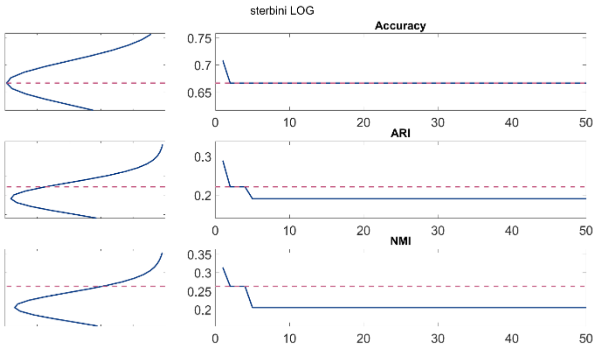

Figure A. 18. ***Illustration of the peak performance zone on gastric mucosa dataset with LOG normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

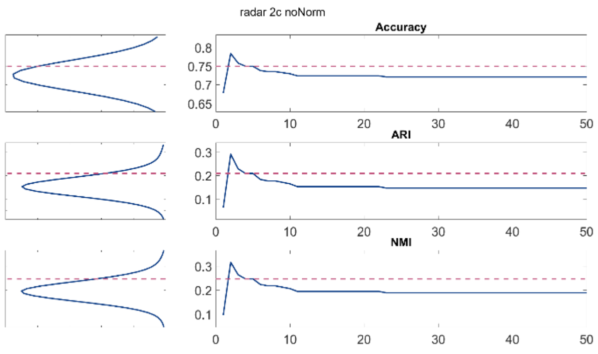

Figure A. 19. ***Illustration of the peak performance zone on radar (two clusters) dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

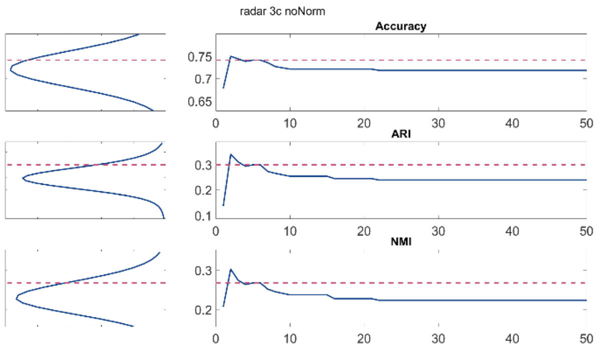

Figure A. 20. ***Illustration of the peak performance zone on radar (three clusters) dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

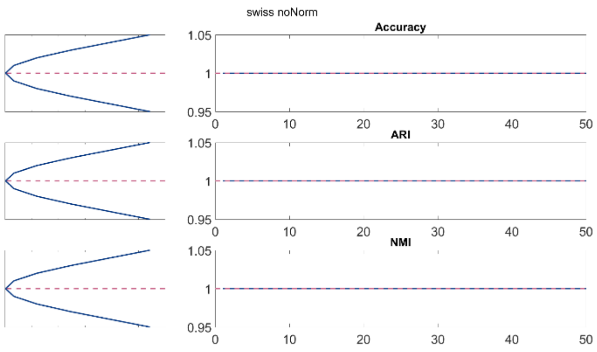

Figure A. 21. ***Illustration of the peak performance zone on tripartite-swiss-roll dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

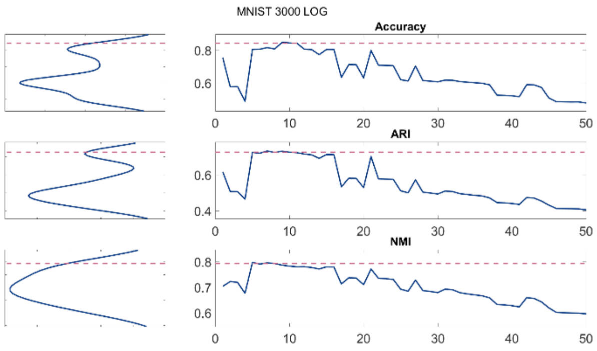

Figure A. 22. ***Illustration of the peak performance zone on MNIST 3000 dataset with LOG normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

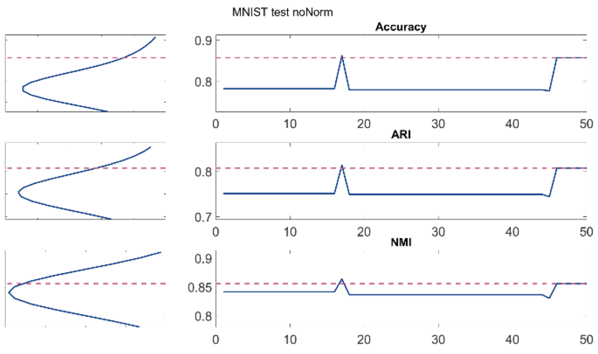

Figure A. 23. ***Illustration of the peak performance zone on MNIST test dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

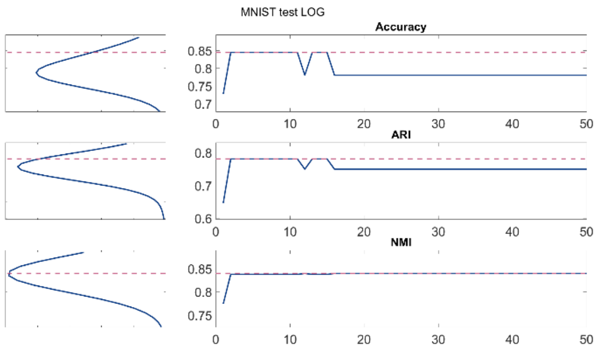

Figure A. 24. ***Illustration of the peak performance zone on MNIST test dataset with LOG normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

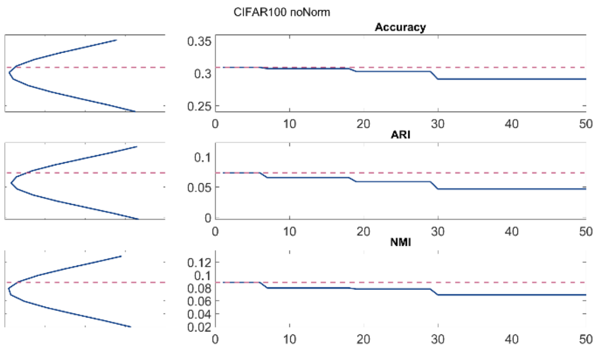

Figure A. 25. ***Illustration of the peak performance zone on CIFAR100 dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

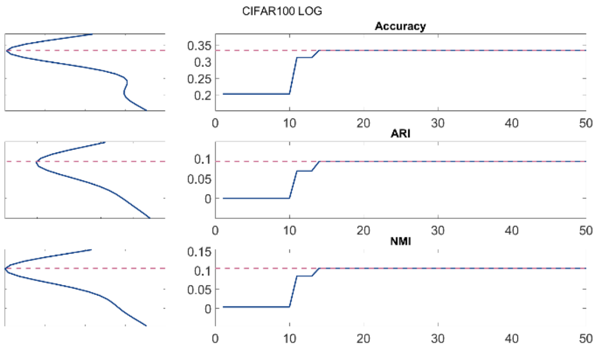

Figure A. 26. ***Illustration of the peak performance zone on CIFAR100 dataset with LOG normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

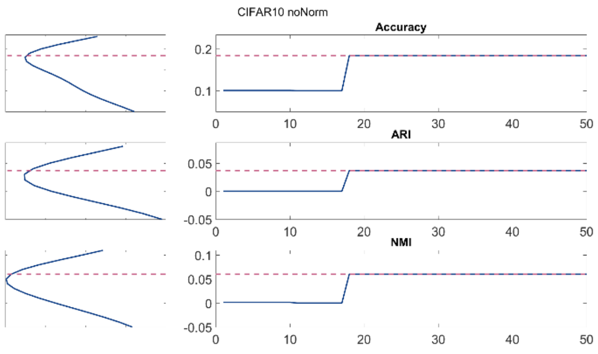

Figure A. 27. ***Illustration of the peak performance zone on CIFAR10 dataset without normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

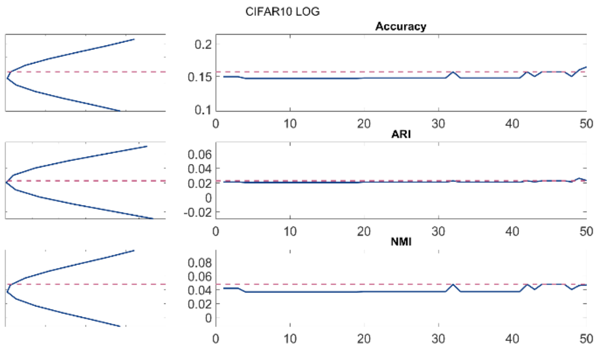

Figure A. 28. ***Illustration of the peak performance zone on CIFAR10 dataset with LOG normalization.*** *Accuracy, ARI and NMI performances from different input parameter values for the method MC-MCL dual (right). Distribution plot from the different performances achieved through the different evaluation measures (left). The magenta line represents the 95 highest performance percentile and is marked both in the distribution and performance plots.*

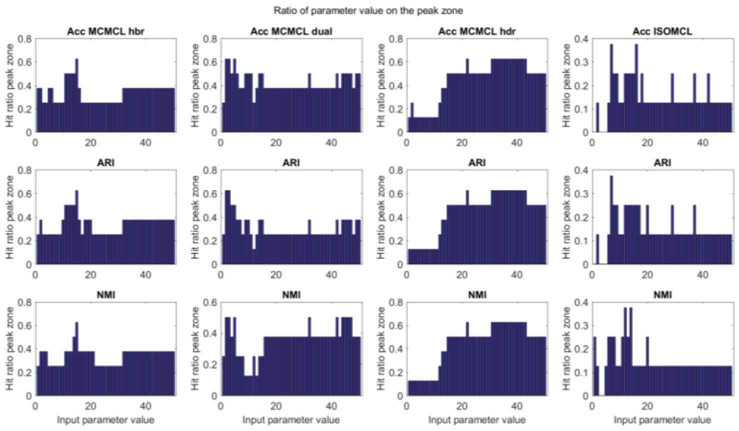

Figure A. 29. ***Peak performance zone hit ratio for each parameter value for MC-MCL and isoMCL variants across all datasets.*** *For each measure, Accuracy (Acc), ARI and NMI, the bars display the hit ratio of each parameter value in which its performance across the different data was on the so-called peak performance zone (highest performances according to a 95 percentile). The first subplot columns shows the hit frequence for the MC-MCL-hbr variant, the second of MC-MCL dual variant, the third for the MC-MCL hdr variant and the last column for isoMCL variant.*

Acc	D1	D2	D3	D4	D5	D6	D7	D8	D9
M1	0.75	0.77	0.75	1.00	0.85	0.30	0.18	0.91	0.94
M2	0.71	0.78	0.75	1.00	0.85	0.33	0.18	0.86	0.91
M3	0.71	0.80	0.74	1.00	0.79	0.35	0.19	0.85	0.92
M4	0.75	0.77	0.74	1.00	0.82	0.36	0.19	0.85	0.90
M5	0.71	0.71	0.74	1.00	0.75	0.35	0.19	0.86	0.60
M6	0.67	0.60	0.42	1.00	0.48	0.21	0.17	0.54	0.66
M7	0.71	0.64	0.62	1.00	0.69	0.22	0.14	0.80	0.81
M8	0.67	0.69	0.56	0.64	0.75	0.34	0.19	0.75	-
M9	0.67	0.71	0.66	0.54	0.56	0.34	0.20	0.41	-
M10	0.54	0.65	0.65	0.85	0.27	0.20	0.10	0.31	-
M11	0.58	0.68	0.64	1.00	0.26	0.20	0.10	0.11	0.11
M12	0.67	0.71	0.62	0.56	0.57	0.35	0.20	0.57	0.56
M13	0.50	0.64	0.64	1.00	0.10	0.20	0.10	0.11	0.11
M14	-	-	-	-	-	0.36	0.22	0.82	0.86
M15	-	-	-	-	-	0.35	0.22	0.82	0.86
M16	-	-	-	-	-	0.36	0.22	0.82	0.87
M17	-	-	-	-	-	0.35	0.22	0.82	0.86
M18	-	-	-	-	-	-	-	0.91	0.93

Table A. 1. ***Accuracy (acc) performance summary of all clustering methods M across all datasets D.*** *Methods list: M1: isoMCL; M2:MC-MCL dual; M3: MC-MCL hbr; M4: MC-MCL hdr; M5: MC-MCL; M6: MCL; M7: cciMST; M8: MC-AP; M9: AP; M10: DPSP; M11: DBSCAN; M12: Kmeans; M13: Single linkage; M14: HOT; M15: HOMO; M16: HOE; M17: HIT; M18: HOE-CNN. Data list: D1: Gastric mucosa; D2: Radar 2C; D3: Radar 3C; D4: Tripartite-Swiss-Roll; D5 MNIST3000; D6: CIFAR100; D7: CIFAR10; D8: MNIST test; D9: MNIST full. Nonlinear MCL variants are marked with red, whereas the original MCL algorithm is marked in green.*

ARI	D1	D2	D3	D4	D5	D6	D7	D8	D9
M1	0.33	0.25	0.32	1.00	0.73	0.06	0.04	0.82	0.88
M2	0.29	0.29	0.34	1.00	0.73	0.09	0.04	0.81	0.89
M3	0.29	0.35	0.34	1.00	0.69	0.10	0.03	0.80	0.89
M4	0.36	0.25	0.35	1.00	0.74	0.11	0.03	0.79	0.87
M5	0.29	0.17	0.27	1.00	0.62	0.08	0.03	0.73	0.58
M6	0.19	0.04	0.03	1.00	0.25	0.00	0.04	0.38	0.51
M7	0.24	0.00	0.15	1.00	0.57	0.00	0.02	0.71	0.78
M8	0.20	0.14	0.08	0.47	0.58	0.09	0.03	0.69	-
M9	0.20	0.17	0.23	0.09	0.39	0.13	0.03	0.33	-
M10	0.15	0.02	0.02	0.87	0.08	0.00	0.00	0.31	-
M11	0.28	0.10	0.01	1.00	0.00	0.00	0.00	0.00	0.00
M12	0.20	0.18	0.16	0.10	0.40	0.12	0.03	0.42	0.40
M13	0.01	0.00	0.01	1.00	0.00	0.00	0.00	0.00	0.00
M14	-	-	-	-	-	0.11	0.04	0.65	0.72
M15	-	-	-	-	-	0.11	0.04	0.65	0.72
M16	-	-	-	-	-	0.12	0.04	0.65	0.74
M17	-	-	-	-	-	0.11	0.04	0.65	0.72
M18	-	-	-	-	-	-	-	0.76	0.82

Table A. 2. ***ARI performance summary of all clustering methods M across all datasets D.*** *Methods list: M1: isoMCL; M2:MC-MCL dual; M3: MC-MCL hbr; M4: MC-MCL hdr; M5: MC-MCL; M6: MCL; M7: cciMST; M8: MC-AP; M9: AP; M10: DPSP; M11: DBSCAN; M12: Kmeans; M13: Single linkage; M14: HOT; M15: HOMO; M16: HOE; M17: HIT; M18: HOE-CNN. Data list: D1: Gastric mucosa; D2: Radar 2C; D3: Radar 3C; D4: Tripartite-Swiss-Roll; D5 MNIST3000; D6: CIFAR100; D7: CIFAR10; D8: MNIST test; D9: MNIST full. Nonlinear MCL variants are marked with red, whereas the original MCL algorithm is marked in green.*

Gastric Mucosa	Acc	ARI	NMI	Norm
MCMCLeuc_hdr_Fsqrt (1)	0.75	0.36	0.37	LOG
MCMCLeuc_hdr_Flog (1)	0.75	0.36	0.37	LOG
isoMCLcorr (1)	0.75	0.33	0.31	LOG
MCMCLspea_hdr_Flog (1)	0.75	0.33	0.31	LOG
MCMCLcorr_hbr (1)	0.71	0.29	0.31	LOG
MCMCLcorr_hbr_Flog (1)	0.71	0.29	0.31	LOG
MCMCLcorr_dual (1)	0.71	0.29	0.31	LOG
MCMCLcorr_dual_Flog (1)	0.71	0.29	0.31	LOG
MCMCLcorr_hdr (1)	0.71	0.29	0.31	LOG
MCMCLcorr_hdr_Flog (1)	0.71	0.29	0.31	LOG
MC-MCLl corr	0.71	0.29	0.31	LOG
MC-MCL corr	0.71	0.29	0.31	LOG
MC-MCLs corr	0.71	0.26	0.31	LOG
isoMCLcorr (1)	0.71	0.25	0.26	-
MCMCLspea_hdr_Flog (1)	0.71	0.25	0.26	noNorm
cciMSTcorr	0.71	0.24	0.26	-
Kmeans corr	0.67	0.20	0.26	LOG
MCAP corr	0.67	0.20	0.24	LOG
AP corr	0.67	0.20	0.24	LOG
cciMSTeuc	0.67	0.20	0.24	LOG
cciMSTcorr	0.67	0.20	0.24	LOG
cciMSTeuc	0.67	0.22	0.23	-
DBSCAN corr	0.58	0.28	0.38	-
MCL corr	0.67	0.19	0.23	-
MCMCLcorr_dual (1)	0.67	0.19	0.23	noNorm
MCMCLcorr_dual_Flog (1)	0.67	0.19	0.23	noNorm
MCMCLeuc_dual_Fsqrt (1)	0.58	0.23	0.28	noNorm
MCL corr	0.67	0.19	0.21	LOG
MC-MCLs corr	0.67	0.18	0.23	-
MCMCLeuc_hdr_Flog (1)	0.63	0.15	0.24	noNorm
MCMCLeuc_hbr (1)	0.63	0.14	0.27	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.63	0.14	0.27	LOG

MCMCLeuc_hbr_Flog (1)	0.63	0.14	0.27	LOG
MCAP eucl	0.63	0.15	0.22	-
MC-MCLl corr	0.54	0.21	0.30	-
isoMCLeuc (1)	0.63	0.15	0.20	LOG
MCMCLeuc_dual_Fsqrt (1)	0.63	0.15	0.20	LOG
MCMCLeuc_dual_Flog (1)	0.63	0.15	0.20	LOG
MCMCLspea_hdr_Flog (1)	0.63	0.15	0.20	LOG
MC-MCLs eucl	0.63	0.15	0.20	LOG
MC-MCLl eucl	0.63	0.15	0.20	LOG
MCAP eucl	0.63	0.15	0.20	LOG
MCMCLeuc_hbr_Flog (1)	0.63	0.13	0.21	noNorm
MCMCLeuc_hdr (1)	0.63	0.13	0.21	LOG
MCAP corr	0.63	0.14	0.21	-
MCMCLspea_hdr (1)	0.63	0.15	0.14	noNorm
MCMCLspea_hdr_Fsqrt (1)	0.63	0.15	0.14	noNorm
MCMCLspea_hdr_Flog (1)	0.63	0.15	0.14	noNorm
MCMCLspea_hdr (1)	0.63	0.15	0.14	LOG
MCMCLspea_hdr_Fsqrt (1)	0.63	0.15	0.14	LOG
MCMCLspea_hdr_Flog (1)	0.63	0.15	0.14	LOG
MC-MCL corr	0.54	0.15	0.22	-
DPSPeuc	0.54	0.15	0.21	-
DPSPcorr	0.54	0.15	0.21	-
MCMCLeuc_dual (1)	0.63	0.12	0.14	LOG
AP corr	0.00	0.19	0.24	-
MC-MCLs eucl	0.54	0.18	0.18	-
MCMCLeuc_hbr_Fsqrt (1)	0.54	0.18	0.18	noNorm
MCMCLeuc_hdr_Fsqrt (1)	0.54	0.18	0.18	noNorm
Kmeans corr	0.58	0.15	0.17	-
MCMCLcorr_hbr (1)	0.58	0.10	0.18	noNorm
MCMCLcorr_hbr_Flog (1)	0.58	0.10	0.18	noNorm
MCMCLeuc_dual_Flog (1)	0.58	0.10	0.18	noNorm
MCMCLcorr_hdr (1)	0.58	0.10	0.18	noNorm
MCMCLcorr_hdr_Flog (1)	0.58	0.10	0.18	noNorm

MCMCLspea_hbr (1)	0.58	0.10	0.12	noNorm
MCMCLspea_hbr_Flog (1)	0.58	0.10	0.12	noNorm
MCMCLspea_dual (1)	0.58	0.10	0.12	noNorm
MCMCLspea_dual_Flog (1)	0.58	0.10	0.12	noNorm
MCMCLspea_hbr (1)	0.58	0.10	0.12	LOG
MCMCLspea_hbr_Flog (1)	0.58	0.10	0.12	LOG
MCMCLspea_dual (1)	0.58	0.10	0.12	LOG
MCMCLspea_dual_Flog (1)	0.58	0.10	0.12	LOG
MC-MCLl eucl	0.58	0.10	0.18	-
MC-MCL eucl	0.54	0.12	0.09	LOG
cciMSTspear	0.54	0.05	0.12	-
cciMSTspear	0.54	0.05	0.12	LOG
MCMCLspea_hbr_Fsqrt (1)	0.54	0.05	0.12	noNorm
MCMCLspea_dual_Fsqrt (1)	0.54	0.05	0.12	noNorm
MCMCLspea_hbr_Fsqrt (1)	0.54	0.05	0.12	LOG
MCMCLspea_dual_Fsqrt (1)	0.54	0.05	0.12	LOG
isoMCLspear (1)	0.54	0.06	0.06	-
isoMCLspear (1)	0.54	0.06	0.06	LOG
MCMCLspea_hdr_Flog (1)	0.54	0.06	0.06	noNorm
MCMCLspea_hdr_Flog (1)	0.54	0.06	0.06	LOG
DBSCAN corr	0.54	0.02	0.06	LOG
Single linkage	0.50	0.01	0.01	LOG
Single linkage	0.50	0.01	0.01	-
DBSCAN eucl	0.50	0.01	0.01	-
AP eucl	0.50	0.01	0.01	-
DBSCAN eucl	0.50	0.01	0.01	LOG
Kmeans eucl	0.42	0.01	0.06	LOG
isoMCLeuc (1)	0.50	0.01	-0.02	-
MCMCLeuc_hbr (1)	0.50	0.01	-0.02	noNorm
MCMCLeuc_dual (1)	0.50	0.01	-0.02	noNorm
MCMCLeuc_hdr (1)	0.50	0.01	-0.02	noNorm
MCMCLspea_hdr_Flog (1)	0.50	0.01	-0.02	noNorm
MCL eucl	0.00	0.00	0.00	LOG

AP eucl	0.00	0.00	0.00	LOG
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.00	noNorm
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.00	noNorm
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.00	noNorm
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.00	LOG
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.00	LOG
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.00	LOG
MC-MCL eucl	0.00	0.01	-0.02	-
MCL eucl	0.00	0.01	-0.02	-
Kmeans eucl	0.46	-0.03	-0.04	-
DPSPspear	0.42	-0.06	-0.05	-
DPSPspear	0.42	-0.06	-0.05	LOG
DPSPeuc	0.42	-0.06	-0.05	LOG
DPSPcorr	0.42	-0.06	-0.05	LOG

Table A. 3. ***Clustering performance in Gastric mucosa microbiome data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

Radar (two clusters)	Acc	ARI	NMI	Norm
MCMCLspea_dual_Flog (1)	0.78	0.29	0.32	-
MCMCLspea_hbr_Flog (1)	0.77	0.27	0.30	-
MCMCLspea_dual (1)	0.77	0.27	0.30	-
MCMCLeuc_hbr_Fsqrt (1)	0.80	0.35	0.25	-
MCMCLcorr_dual_Flog (1)	0.77	0.26	0.29	-
isoMCLspear (1)	0.77	0.25	0.28	-
MCMCLspea_hdr_Flog (1)	0.77	0.25	0.28	-
isoMCLcorr (1)	0.76	0.24	0.27	-
MCMCLspea_hdr_Flog (1)	0.76	0.24	0.27	-
MCMCLspea_hbr (1)	0.75	0.22	0.26	-
MCMCLspea_hdr_Flog (1)	0.75	0.22	0.26	-
MCMCLcorr_hdr_Flog (1)	0.75	0.22	0.25	-
MCMCLeuc_hdr_Flog (1)	0.75	0.25	0.16	-
MCMCLcorr_dual (1)	0.75	0.20	0.24	-
MCMCLeuc_hbr_Flog (1)	0.74	0.23	0.15	-
MCMCLcorr_hdr (1)	0.73	0.17	0.21	-
MCMCLspea_dual_Fsqrt (1)	0.73	0.16	0.20	-
MCMCLspea_hdr (1)	0.72	0.15	0.19	-
Kmeans eucl	0.71	0.18	0.13	-
AP eucl	0.71	0.17	0.13	-
MCMCLeuc_dual_Flog (1)	0.71	0.17	0.13	-
MC-MCLl eucl	0.71	0.17	0.12	-
MCMCLspea_hbr_Fsqrt (1)	0.70	0.11	0.15	-
MCMCLcorr_hbr (1)	0.70	0.09	0.13	-
MCMCLcorr_hbr_Flog (1)	0.70	0.09	0.13	-
isoMCLeuc (1)	0.70	0.15	0.11	-
MCMCLspea_hdr_Flog (1)	0.70	0.15	0.11	-
DBSCAN corr	0.68	0.10	0.14	-
MCMCLeuc_dual_Fsqrt (1)	0.69	0.14	0.11	-
MC-MCLs corr	0.69	0.09	0.13	-
MCAP eucl	0.69	0.14	0.09	-

MCMCLspea_hdr_Fsqrt (1)	0.69	0.08	0.12	-
MC-MCLl corr	0.69	0.08	0.12	-
MCMCLeuc_dual (1)	0.68	0.13	0.08	-
MC-MCL corr	0.68	0.07	0.11	-
MCMCLeuc_hdr_Fsqrt (1)	0.65	0.08	0.09	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.20	-
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.20	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.20	-
MCMCLeuc_hbr (1)	0.66	0.03	0.05	-
MCMCLeuc_hdr (1)	0.66	0.02	0.04	-
cciMSTspear	0.62	0.06	0.06	-
DPSPcorr	0.65	0.02	0.03	-
MCL eucl	0.60	0.04	0.06	-
DPSPspear	0.65	0.01	0.02	-
DPSPeuc	0.65	0.01	0.02	-
cciMSTeuc	0.64	0.00	0.01	-
Single linkage	0.64	0.00	0.01	-
DBSCAN eucl	0.64	0.00	0.01	-
Kmeans corr	0.59	0.02	0.01	-
AP corr	0.57	0.01	0.00	-
cciMSTcorr	0.52	-0.02	0.03	-
MCAP corr	0.52	-0.03	0.02	-
MC-MCLs eucl	0.00	0.00	0.00	-
MC-MCL eucl	0.00	0.00	0.00	-
MCL corr	0.00	0.00	0.00	-

Table A. 4. ***Clustering performance in radar (two clusters) data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

Radar (three clusters)	Acc	ARI	NMI	Norm
MCMCLcorr_dual_Flog (1)	0.74	0.36	0.33	-
MCMCLspea_dual (1)	0.75	0.34	0.30	-
MCMCLspea_hdr_Flog (1)	0.74	0.35	0.32	-
MCMCLspea_hbr (1)	0.74	0.34	0.32	-
isoMCLspear (1)	0.75	0.32	0.28	-
MCMCLspea_dual_Flog (1)	0.75	0.32	0.29	-
MCMCLspea_hdr_Flog (1)	0.75	0.32	0.28	-
MCMCLspea_hbr_Flog (1)	0.74	0.33	0.31	-
MC-MCLs corr	0.74	0.27	0.38	-
MCMCLcorr_hdr_Flog (1)	0.73	0.27	0.33	-
MCMCLeuc_hbr_Fsqrt (1)	0.75	0.32	0.25	-
MCMCLspea_hdr (1)	0.73	0.31	0.30	-
MCMCLcorr_hbr_Flog (1)	0.72	0.24	0.35	-
isoMCLcorr (1)	0.72	0.28	0.28	-
MCMCLspea_hdr_Flog (1)	0.72	0.28	0.28	-
MCMCLcorr_hbr (1)	0.72	0.22	0.34	-
MC-MCLl corr	0.70	0.20	0.35	-
MCMCLcorr_hdr (1)	0.70	0.26	0.28	-
MCMCLcorr_dual (1)	0.70	0.26	0.28	-
MC-MCL corr	0.70	0.18	0.32	-
isoMCLeuc (1)	0.66	0.24	0.24	-
MCMCLeuc_dual (1)	0.66	0.24	0.24	-
MCMCLspea_hdr_Flog (1)	0.66	0.24	0.24	-
MCMCLeuc_hdr_Flog (1)	0.70	0.24	0.18	-
AP eucl	0.66	0.23	0.25	-
MCMCLeuc_hbr_Flog (1)	0.69	0.21	0.17	-
MCMCLeuc_hdr (1)	0.67	0.21	0.17	-
MCMCLspea_hbr_Fsqrt (1)	0.66	0.08	0.14	-
MCMCLeuc_dual_Flog (1)	0.66	0.15	0.13	-
Kmeans eucl	0.62	0.16	0.15	-
MCMCLeuc_hbr (1)	0.66	0.05	0.07	-

MCAP corr	0.56	0.08	0.22	-
cciMSTeuc	0.62	0.15	0.14	-
cciMSTcorr	0.52	0.06	0.20	-
MCMCLspea_dual_Fsqrt (1)	0.59	0.07	0.13	-
MC-MCLl eucl	0.00	0.15	0.13	-
MCMCLspea_hdr_Fsqrt (1)	0.58	0.06	0.11	-
DPSPcorr	0.65	0.02	0.03	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.24	-
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.24	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.24	-
cciMSTspear	0.57	0.04	0.07	-
DPSPspear	0.64	0.01	0.02	-
DPSPeuc	0.64	0.01	0.02	-
Single linkage	0.64	0.01	0.02	-
DBSCAN eucl	0.64	0.01	0.02	-
DBSCAN corr	0.64	0.01	0.02	-
MCAP eucl	0.52	0.04	0.10	-
Kmeans corr	0.52	0.05	0.03	-
AP corr	0.54	0.04	0.02	-
MCL eucl	0.42	0.03	0.05	-
MCMCLeuc_dual_Fsqrt (1)	0.64	0.00	0.00	-
MCMCLeuc_hdr_Fsqrt (1)	0.64	0.00	0.00	-
MC-MCLs eucl	0.00	0.00	0.00	-
MC-MCL eucl	0.00	0.00	0.00	-
MCL corr	0.00	0.00	0.00	-

Table A. 5. ***Clustering performance in radar (three clusters) data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

Tripartite-Swiss-Roll	Acc	ARI	NMI	Norm
Single linkage euc	1.00	1.00	1.00	-
MC-MCLl eucl	1.00	1.00	1.00	-
MC-MCL eucl	1.00	1.00	1.00	-
MCL eucl	1.00	1.00	1.00	-
isoMCLeuc (1)	1.00	1.00	1.00	-
DBSCAN eucl	1.00	1.00	1.00	-
cciMSTeuc	1.00	1.00	1.00	-
MCMCLeuc_hbr (1)	1.00	1.00	1.00	-
MCMCLeuc_hbr_Fsqrt (1)	1.00	1.00	1.00	-
MCMCLeuc_hbr_Flog (1)	1.00	1.00	1.00	-
MCMCLeuc_dual (1)	1.00	1.00	1.00	-
MCMCLeuc_dual_Fsqrt (1)	1.00	1.00	1.00	-
MCMCLeuc_dual_Flog (1)	1.00	1.00	1.00	-
MCMCLeuc_hdr (1)	1.00	1.00	1.00	-
MCMCLeuc_hdr_Flog (1)	1.00	1.00	1.00	-
MCMCLspea_hdr_Flog (1)	1.00	1.00	1.00	-
DPSPeuc	0.85	0.87	0.82	-
MCMCLeuc_hdr_Fsqrt (1)	0.78	0.43	0.62	-
MCAP eucl	0.64	0.47	0.58	-
DPSPspear	0.63	0.22	0.24	-
MC-MCLs eucl	0.00	0.47	0.63	-
cciMSTspear	0.54	0.13	0.17	-
MCMCLspea_hbr_Fsqrt (1)	0.54	0.13	0.17	-
MCMCLspea_dual_Fsqrt (1)	0.54	0.13	0.17	-
Kmeans corr	0.52	0.13	0.18	-
Kmeans eucl	0.56	0.10	0.20	-
isoMCLspear (1)	0.43	0.11	0.20	-
MCMCLspea_hbr (1)	0.43	0.11	0.20	-
MCMCLspea_dual (1)	0.43	0.11	0.20	-
MCMCLspea_hdr_Flog (1)	0.43	0.11	0.20	-
AP corr	0.00	0.13	0.23	-

AP eucl	0.54	0.09	0.19	-
isoMCLcorr (1)	0.66	0.04	0.04	-
MCMCLcorr_hbr (1)	0.66	0.04	0.04	-
MCMCLcorr_hbr_Flog (1)	0.66	0.04	0.04	-
MCMCLcorr_dual (1)	0.66	0.04	0.04	-
MCMCLcorr_dual_Flog (1)	0.66	0.04	0.04	-
MCMCLcorr_hdr (1)	0.66	0.04	0.04	-
MCMCLcorr_hdr_Flog (1)	0.66	0.04	0.04	-
MCMCLspea_hdr_Flog (1)	0.66	0.04	0.04	-
MC-MCLl corr	0.66	0.04	0.04	-
MC-MCL corr	0.66	0.04	0.04	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.23	-
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.23	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.23	-
MCMCLspea_hbr_Flog (1)	0.59	0.00	0.05	-
MCMCLspea_dual_Flog (1)	0.59	0.00	0.05	-
cciMSTcorr	0.46	0.00	0.04	-
MC-MCLs corr	0.46	0.00	0.04	-
MCL corr	0.46	0.00	0.04	-
MCAP corr	0.43	-0.01	0.12	-
DPSPcorr	0.43	-0.01	0.12	-
DBSCAN corr	0.00	0.00	0.03	-
MCMCLspea_hdr (1)	0.00	0.00	0.00	-
MCMCLspea_hdr_Fsqrt (1)	0.00	0.00	0.00	-
MCMCLspea_hdr_Flog (1)	0.00	0.00	0.00	-

Table A. 6. ***Clustering performance in tripartite-swiss-roll data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

MNIST 3000	Acc	ARI	NMI	Norm
MCMCLspea_hdr_Flog (1)	0.82	0.74	0.81	-
MCMCLspea_hdr_Flog (1)	0.82	0.74	0.81	LOG
MCMCLcorr_dual_Flog (1)	0.82	0.73	0.81	-
MCMCLeuc_dual (1)	0.85	0.73	0.79	LOG
MCMCLeuc_dual_Fsqrt (1)	0.81	0.73	0.79	LOG
isoMCLeuc (1)	0.85	0.73	0.78	LOG
MCMCLeuc_dual_Flog (1)	0.84	0.72	0.78	LOG
MCMCLcorr_hdr (1)	0.75	0.70	0.80	-
MCMCLcorr_hdr_Flog (1)	0.80	0.70	0.78	LOG
MCMCLeuc_dual (1)	0.80	0.71	0.78	-
MCMCLcorr_hdr_Flog (1)	0.75	0.70	0.79	-
MCMCLcorr_hbr_Flog (1)	0.79	0.69	0.78	LOG
MCMCLeuc_dual_Flog (1)	0.79	0.69	0.77	-
MCMCLspea_hdr (1)	0.74	0.67	0.80	-
MCMCLspea_hdr (1)	0.74	0.67	0.80	LOG
MCMCLeuc_dual_Fsqrt (1)	0.79	0.69	0.77	-
MCMCLspea_dual_Flog (1)	0.74	0.67	0.80	-
MCMCLspea_dual_Flog (1)	0.74	0.67	0.80	LOG
MCMCLspea_hbr_Flog (1)	0.74	0.68	0.78	-
MCMCLspea_hbr_Flog (1)	0.74	0.68	0.78	LOG
isoMCLspear (1)	0.74	0.66	0.79	-
isoMCLspear (1)	0.74	0.66	0.79	LOG
MCMCLspea_hdr_Flog (1)	0.74	0.66	0.79	-
MCMCLcorr_hbr_Flog (1)	0.74	0.67	0.77	-
MCMCLeuc_hdr (1)	0.78	0.67	0.75	LOG
MCMCLcorr_hbr (1)	0.74	0.67	0.77	-
MCMCLspea_dual_Fsqrt (1)	0.73	0.65	0.78	-
MCMCLspea_dual_Fsqrt (1)	0.73	0.65	0.78	LOG
MCMCLeuc_hdr_Fsqrt (1)	0.76	0.64	0.73	LOG
MCMCLcorr_hbr (1)	0.74	0.64	0.76	LOG
MCMCLcorr_dual_Flog (1)	0.73	0.61	0.78	LOG

isoMCLcorr (1)	0.73	0.61	0.77	LOG
isoMCLcorr (1)	0.73	0.61	0.77	-
MCMCLspea_hdr_Flog (1)	0.73	0.61	0.77	-
MC-MCL eucl	0.75	0.62	0.70	LOG
MCMCLeuc_hbr (1)	0.75	0.62	0.70	LOG
MCMCLcorr_hdr (1)	0.72	0.64	0.76	LOG
MCMCLspea_hbr_Fsqrt (1)	0.75	0.61	0.72	-
MCMCLspea_hbr_Fsqrt (1)	0.75	0.61	0.72	LOG
MCMCLcorr_dual (1)	0.73	0.61	0.77	LOG
MC-MCLl corr	0.72	0.64	0.75	-
MCMCLspea_hdr_Fsqrt (1)	0.73	0.64	0.74	-
MCMCLspea_hdr_Fsqrt (1)	0.73	0.64	0.74	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.75	0.61	0.70	LOG
MC-MCLs eucl	0.75	0.61	0.70	LOG
MCMCLcorr_dual	0.71	0.60	0.76	-
MCAP corr	0.71	0.61	0.73	-
MCMCLspea_hbr	0.70	0.59	0.77	-
MCMCLspea_hbr	0.70	0.59	0.77	LOG
MCMCLeuc_hbr	0.73	0.61	0.71	-
MCAP eucl	0.75	0.58	0.68	LOG
cciMSTcorr	0.66	0.63	0.74	-
MCMCLspea_dual	0.67	0.57	0.76	-
MCMCLspea_dual	0.67	0.57	0.76	LOG
MC-MCL corr	0.66	0.60	0.73	-
MCMCLeuc_hdr	0.71	0.60	0.69	-
cciMSTcorr	0.69	0.57	0.72	LOG
MC-MCLl corr	0.68	0.57	0.71	LOG
MCAP corr	0.68	0.56	0.70	LOG
isoMCLeuc (1)	0.68	0.54	0.70	-
MCMCLspea_hdr_Flog (1)	0.68	0.54	0.70	-
MC-MCLs corr	0.62	0.57	0.71	LOG
MCMCLeuc_hbr_Flog (1)	0.71	0.53	0.66	LOG
MC-MCLs corr	0.67	0.53	0.68	-

MC-MCLl eucl	0.66	0.55	0.66	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.67	0.52	0.66	-
MCMCLeuc_hdr_Flog (1)	0.66	0.51	0.65	LOG
MCAP eucl	0.65	0.48	0.63	-
MC-MCL corr	0.57	0.51	0.68	LOG
cciMSTeuc	0.64	0.48	0.62	-
MCMCLeuc_hdr_Fsqrt (1)	0.63	0.48	0.62	-
cciMSTspear	0.57	0.45	0.68	-
cciMSTspear	0.57	0.45	0.68	LOG
MCMCLeuc_hdr_Flog (1)	0.64	0.45	0.59	-
MCMCLeuc_hbr_Flog (1)	0.62	0.44	0.59	-
MC-MCL eucl	0.58	0.45	0.58	-
Kmeans corr	0.57	0.40	0.53	LOG
cciMSTeuc	0.52	0.38	0.59	LOG
Kmeans corr	0.55	0.40	0.53	-
MC-MCLs eucl	0.52	0.40	0.55	-
Kmeans eucl	0.55	0.40	0.52	LOG
AP corr	0.56	0.39	0.51	LOG
AP eucl	0.56	0.37	0.47	LOG
Kmeans eucl	0.52	0.35	0.50	-
MCL corr	0.48	0.25	0.55	LOG
MC-MCLl eucl	0.46	0.35	0.51	-
AP corr	0.51	0.33	0.48	-
MCL corr	0.41	0.18	0.46	-
AP eucl	0.48	0.25	0.38	-
MCL eucl	0.35	0.12	0.42	LOG
DPSPeuc	0.27	0.08	0.31	-
DPSPspear	0.20	0.09	0.30	-
DPSPspear	0.20	0.09	0.30	LOG
MCL eucl	0.26	0.06	0.30	-
DBSCAN corr	0.26	0.00	0.43	LOG
DPSPcorr	0.18	0.03	0.17	LOG
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.45	-

MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.45	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.45	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.45	LOG
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.45	LOG
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.45	LOG
DPSPcorr	0.16	0.02	0.17	-
DPSPeuc	0.10	0.00	0.01	LOG
DBSCAN corr	0.10	0.00	0.02	-
Single linkage euc	0.10	0.00	0.01	-
DBSCAN eucl	0.10	0.00	0.01	-
Single linkage euc	0.10	0.00	0.01	LOG
DBSCAN eucl	0.10	0.00	0.01	LOG

Table A. 7. ***Clustering performance in MNIST 3000 data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

MNIST test	Acc	ARI	NMI	Norm
isoMCLeuc (1)	0.91	0.82	0.84	LOG
MCMCLcorr_dual_Flog (1)	0.86	0.81	0.86	-
isoMCLcorr (1)	0.88	0.80	0.84	LOG
MCMCLcorr_dual (1)	0.86	0.81	0.86	-
MCMCLcorr_hbr_Flog (1)	0.85	0.80	0.85	-
isoMCLspear (1)	0.88	0.79	0.84	-
isoMCLspear (1)	0.88	0.79	0.84	LOG
isoMCLcorr (1)	0.87	0.80	0.84	-
MCMCLcorr_hdr (1)	0.85	0.79	0.84	-
MCMCLspea_hdr (1)	0.85	0.78	0.83	-
MCMCLspea_hdr (1)	0.85	0.78	0.83	LOG
MCMCLeuc_dual (1)	0.84	0.78	0.84	LOG
isoMCLeuc (1)	0.85	0.78	0.83	-
MCMCLcorr_hdr_Flog (1)	0.84	0.78	0.83	-
MCMCLcorr_hbr_Flog (1)	0.85	0.78	0.83	LOG
MCMCLspea_dual_Flog (1)	0.83	0.77	0.83	-
MCMCLspea_dual_Flog (1)	0.83	0.77	0.83	LOG
HOE-CNN (1)	0.91	0.76	0.78	-
MCMCLeuc_hbr (1)	0.87	0.75	0.79	LOG
MCMCLcorr_dual_Flog (1)	0.78	0.75	0.84	LOG
MCMCLspea_hbr_Flog (1)	0.83	0.74	0.82	-
MCMCLspea_hbr_Flog (1)	0.83	0.74	0.82	LOG
MCMCLcorr_dual (1)	0.78	0.75	0.83	LOG
MCMCLeuc_hdr (1)	0.82	0.75	0.80	-
MCMCLeuc_dual_Flog (1)	0.81	0.74	0.82	-
MCMCLeuc_hbr (1)	0.86	0.73	0.77	-
MCMCL_eucl	0.86	0.73	0.77	-
MCMCLspea_dual (1)	0.78	0.74	0.83	-
MCMCLspea_dual (1)	0.78	0.74	0.83	LOG
MCMCLeuc_dual (1)	0.78	0.74	0.83	-
MCMCLcorr_hbr (1)	0.77	0.74	0.83	-
MCMCLspea_hdr_Flog (1)	0.81	0.72	0.81	-

MCMCLspea_hdr_Flog (1)	0.81	0.72	0.81	LOG
MCMCLcorr_hdr_Flog (1)	0.78	0.74	0.82	LOG
MCMCLcorr_hdr (1)	0.78	0.73	0.81	LOG
MCMCLcorr_hbr (1)	0.77	0.74	0.82	LOG
MCMCLeuc_dual_Fsqrt (1)	0.80	0.72	0.80	-
cciMSTcorr	0.80	0.71	0.80	-
MCMCLeuc_hdr (1)	0.80	0.72	0.79	LOG
MCMCLeuc_dual_Fsqrt (1)	0.80	0.71	0.79	LOG
MCMCLspea_hbr_Fsqrt (1)	0.82	0.69	0.75	-
MCMCLspea_hbr_Fsqrt (1)	0.82	0.69	0.75	LOG
MCMCLeuc_hdr_Fsqrt (1)	0.79	0.70	0.78	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.81	0.68	0.75	LOG
MCMCLspea_dual_Fsqrt (1)	0.76	0.67	0.81	-
MCMCLspea_hbr (1)	0.76	0.66	0.81	-
MCMCLspea_hbr (1)	0.76	0.66	0.81	LOG
MCMCLeuc_dual_Flog (1)	0.76	0.69	0.79	LOG
MCMCLspea_hdr_Fsqrt (1)	0.79	0.67	0.74	-
MCMCLspea_hdr_Fsqrt (1)	0.79	0.67	0.74	LOG
MCAP corr	0.75	0.69	0.77	-
HOT (1)	0.82	0.65	0.69	-
HOMO (1)	0.82	0.65	0.68	-
HOE (1)	0.82	0.65	0.68	-
HIT (1)	0.82	0.65	0.68	-
MCMCLspea_dual_Fsqrt (1)	0.70	0.65	0.81	LOG
MCMCLeuc_hdr_Flog (1)	0.72	0.66	0.75	LOG
MCMCLl_corr	0.72	0.62	0.77	LOG
MCMCLeuc_hbr_Flog (1)	0.75	0.63	0.73	LOG
MCMCL_corr	0.74	0.63	0.74	-
MCAP eucl	0.73	0.63	0.73	LOG
cciMSTcorr	0.70	0.61	0.73	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.73	0.60	0.72	-
MCMCLeuc_hdr_Flog (1)	0.70	0.62	0.72	-
MCMCL_corr	0.66	0.58	0.75	LOG

MCAP corr	0.66	0.62	0.73	LOG
MCMCLeuc_hbr_Flog (1)	0.72	0.59	0.71	-
MCMCLs_corr	0.68	0.57	0.72	-
MCAP spea	0.65	0.60	0.71	-
MCAP spea	0.65	0.60	0.71	LOG
MCAP eucl	0.70	0.59	0.70	-
MCMCL_eucl	0.69	0.58	0.71	LOG
cciMSTeuc	0.65	0.55	0.73	LOG
MCMCLeuc_hdr_Fsqrt (1)	0.71	0.56	0.69	-
cciMSTeuc	0.71	0.51	0.66	-
MCMCLs_corr	0.66	0.53	0.68	LOG
cciMSTspear	0.60	0.52	0.68	-
cciMSTspear	0.60	0.52	0.68	LOG
MCMCLl_eucl	0.63	0.52	0.66	LOG
MCMCLspea	0.54	0.50	0.70	-
MCMCLspea	0.54	0.50	0.70	LOG
MCMCLs_eucl	0.60	0.44	0.66	LOG
MCMCLl_corr	0.55	0.45	0.66	-
Kmeans corr	0.57	0.42	0.55	LOG
MCMCLl_eucl	0.54	0.38	0.58	-
MCL_corr	0.52	0.42	0.62	-
Kmeans corr	0.55	0.41	0.53	-
MCL_eucl	0.54	0.38	0.56	LOG
Kmeans eucl	0.54	0.39	0.51	-
Kmeans eucl	0.53	0.39	0.52	LOG
MCMCLs_eucl	0.50	0.31	0.57	-
AP corr	0.41	0.33	0.53	-
AP spea	0.41	0.33	0.52	-
AP spea	0.41	0.33	0.52	LOG
AP corr	0.38	0.33	0.54	LOG
DPSPspear	0.31	0.31	0.54	-
DPSPspear	0.31	0.31	0.54	LOG
DPSPcorr	0.32	0.29	0.54	LOG

MCL_corr	0.43	0.20	0.50	LOG
AP eucl	0.41	0.26	0.44	-
AP eucl	0.38	0.31	0.49	LOG
DPSPeuc	0.34	0.17	0.46	-
DPSPcorr	0.31	0.11	0.40	-
MCL_eucl	0.30	0.09	0.35	-
Single linkage euc	0.11	0.00	0.00	LOG
DBSCAN eucl	0.11	0.00	0.00	LOG
DBSCAN corr	0.11	0.00	0.00	LOG
DBSCAN corr	0.11	0.00	0.00	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.40	-
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.40	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.40	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.40	LOG
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.40	LOG
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.40	LOG
DPSPeuc	0.11	0.00	0.00	LOG
Single linkage euc	0.11	0.00	0.00	-
DBSCAN eucl	0.11	0.00	0.00	-

Table A. 8. ***Clustering performance in MNIST test data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

MNIST full	Acc	ARI	NMI	Norm
MCMCLcorr_dual (1)	0.91	0.89	0.90	-
MCMCL_corr_hbr_Flog (1)	0.92	0.89	0.89	-
isoMCLeuc (1)	0.94	0.88	0.88	LOG
HOE-CNN (1)	0.93	0.82	0.86	-
MCMCLcorr_hdr (1)	0.90	0.87	0.89	-
HOE (1)	0.87	0.74	0.76	-
cciMSTcorr	0.81	0.78	0.83	-
HOMO (1)	0.86	0.72	0.74	-
HOT (1)	0.86	0.72	0.74	-
HIT (1)	0.86	0.72	0.74	-
cciMSTspear	0.80	0.69	0.77	-
cciMSTspear	0.80	0.69	0.77	LOG
cciMSTcorr	0.67	0.61	0.75	LOG
MCMCLeuc	0.60	0.58	0.74	-
MCLeuc	0.66	0.51	0.67	LOG
cciMSTeuc	0.62	0.44	0.68	LOG
Kmeans corr	0.56	0.40	0.53	LOG
cciMSTeuc	0.56	0.37	0.65	-
Kmeans eucl	0.55	0.39	0.51	-
Kmeans eucl	0.55	0.38	0.50	LOG
Kmeans corr	0.51	0.37	0.50	-
DBSCAN corr	0.11	0.00	0.00	-
DBSCAN eucl	0.11	0.00	0.00	LOG
Single linkage	0.11	0.00	0.00	LOG
DBSCAN corr	0.11	0.00	0.00	LOG
DBSCAN eucl	0.11	0.00	0.00	-
Single linkage	0.11	0.00	0.00	-

*Table A. 9. **Clustering performance in MNIST full data**. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find* *the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

CIFAR100	Acc	ARI	NMI	Norm
HOE (1)	0.36	0.12	0.14	-
HOT (1)	0.36	0.11	0.14	-
HOMO (1)	0.35	0.11	0.14	-
HIT (1)	0.35	0.11	0.14	-
AP spea	0.34	0.13	0.15	LOG
AP spea	0.34	0.13	0.15	-
MCMCLcorr_hdr (1)	0.36	0.11	0.13	LOG
Kmeans corr	0.35	0.12	0.13	-
MCMCLcorr_hdr_Flog (1)	0.35	0.11	0.12	LOG
Kmeans corr	0.34	0.11	0.13	LOG
MCMCLcorr_hbr (1)	0.35	0.10	0.11	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.35	0.09	0.11	-
MCMCLcorr_hbr_Flog (1)	0.34	0.10	0.12	LOG
Kmeans eucl	0.33	0.09	0.14	LOG
MCAP spea	0.34	0.09	0.10	LOG
MCAP spea	0.34	0.09	0.10	-
MCMCLcorr_hbr (1)	0.33	0.09	0.11	-
MCMCLs_corr	0.35	0.08	0.10	LOG
MCMCLeuc_dual_Flog (1)	0.33	0.09	0.10	LOG
AP corr	0.32	0.11	0.13	LOG
AP corr	0.31	0.11	0.13	-
MCAP corr	0.32	0.10	0.11	LOG
MCMCLspea_hbr_Flog (1)	0.33	0.08	0.10	-
MCMCLspea_hbr_Flog (1)	0.33	0.08	0.10	LOG
Kmeans eucl	0.33	0.08	0.11	-
MCMCLspea_hbr (1)	0.33	0.08	0.10	-
MCMCLspea_hbr (1)	0.33	0.08	0.10	LOG
MCMCLcorr_hbr_Flog (1)	0.32	0.08	0.10	-
MCMCLcorr_dual_Flog (1)	0.32	0.08	0.09	LOG
MCMCLspea_hdr (1)	0.32	0.07	0.10	-

MCMCLspea_hdr (1)	0.32	0.07	0.10	LOG
MCMCLs_corr	0.32	0.08	0.10	-
MCAP corr	0.32	0.08	0.09	-
MCMCLeuc_hbr_Flog (1)	0.32	0.06	0.09	-
MCMCLl_eucl	0.31	0.08	0.09	LOG
MCMCLeuc_hdr_Flog (1)	0.31	0.08	0.09	LOG
MCMCLeuc_hbr_Flog (1)	0.31	0.08	0.09	LOG
MCMCLeuc_hdr_Flog (1)	0.32	0.06	0.09	-
MCMCLspea_hbr_Fsqrt (1)	0.32	0.07	0.09	-
MCMCLspea_hbr_Fsqrt (1)	0.32	0.07	0.09	LOG
AP eucl	0.32	0.05	0.10	LOG
MCMCLcorr_hdr (1)	0.31	0.07	0.09	-
MCMCLspea_hdr_Flog (1)	0.32	0.06	0.08	-
MCMCLspea_hdr_Flog (1)	0.32	0.06	0.08	LOG
MCMCLcorr_hdr_Flog (1)	0.32	0.06	0.09	-
MCMCL_corr	0.32	0.06	0.08	LOG
MCMCLcorr_dual	0.31	0.07	0.09	-
MCMCLl_corr	0.32	0.06	0.07	LOG
MCAP eucl	0.32	0.05	0.07	-
MCMCLeuc_hdr_Fsqrt (1)	0.31	0.05	0.09	-
AP eucl	0.32	0.05	0.09	-
MCMCLspea_dual (1)	0.30	0.05	0.08	-
MCMCLspea_dual (1)	0.30	0.05	0.08	LOG
MCMCLcorr_dual_Flog (1)	0.30	0.06	0.07	-
isoMCLcorr (1)	0.30	0.06	0.07	-
isoMCLcorr (1)	0.30	0.05	0.08	LOG
MCMCLl_eucl	0.31	0.04	0.07	-
MCMCLspea_dual_Flog (1)	0.29	0.05	0.07	-
isoMCLspear (1)	0.29	0.05	0.07	-
isoMCLspear (1)	0.29	0.05	0.07	LOG
MCMCLspea_hdr_Fsqrt (1)	0.29	0.04	0.08	-
MCMCLspea_hdr_Fsqrt (1)	0.29	0.04	0.08	LOG
MCAP eucl	0.30	0.05	0.06	LOG

MCMCLcorr_dual	0.28	0.04	0.07	LOG
MCMCLs_eucl	0.29	0.04	0.05	LOG
MCMCLeuc_hdr_Fsqrt (1)	0.29	0.04	0.05	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.29	0.04	0.05	LOG
MCMCLl_corr	0.29	0.04	0.07	-
MCMCL_corr	0.27	0.03	0.06	-
MCMCLeuc_dual_Fsqrt (1)	0.28	0.02	0.06	-
MCMCLspea_dual_Flog (1)	0.27	0.02	0.06	LOG
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.33	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.33	LOG
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.33	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.33	LOG
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.33	-
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.33	LOG
cciMSTeuc	0.27	0.02	0.03	-
isoMCLeuc (1)	0.24	0.00	0.04	LOG
MCMCLs_eucl	0.26	0.01	0.04	-
MCMCLeuc_dual (1)	0.23	0.00	0.04	LOG
MCMCLeuc_hbr (1)	0.23	0.00	0.04	LOG
cciMSTcorr	0.22	0.00	0.03	LOG
cciMSTcorr	0.22	0.00	0.02	-
cciMSTspear	0.22	0.00	0.02	-
cciMSTspear	0.22	0.00	0.02	LOG
MCMCLeuc_hdr (1)	0.22	0.00	0.03	-
MCMCLeuc_hbr (1)	0.22	0.00	0.03	-
MCMCL_eucl	0.22	0.00	0.03	-
isoMCLeuc (1)	0.22	0.00	0.02	-
MCMCLeuc_dual_Fsqrt (1)	0.21	0.00	0.02	LOG
MCMCLeuc_dual_Flog (1)	0.21	0.00	0.01	-
MCMCLspea_dual_Fsqrt (1)	0.21	0.00	0.01	-
MCMCLspea_dual_Fsqrt (1)	0.21	0.00	0.01	LOG
MCL_eucl	0.21	0.00	0.01	LOG
MCMCLeuc_hdr (1)	0.21	0.00	0.01	LOG

MCMCL_eucl	0.21	0.00	0.01	LOG
MCL_eucl	0.20	0.00	0.01	-
MCMCLeuc_dual (1)	0.21	0.00	0.01	-
MCL_corr	0.20	0.00	0.01	-
DPSPeuc	0.20	0.00	0.01	-
MCL_corr	0.20	0.00	0.01	LOG
DBSCAN corr	0.20	0.00	0.02	LOG
DPSPcorr	0.20	0.00	0.00	-
cciMSTeuc	0.20	0.00	0.00	LOG
DBSCAN corr	0.19	0.00	0.03	-
DPSPcorr	0.20	0.00	0.00	LOG
DPSPspear	0.20	0.00	0.00	-
DPSPspear	0.20	0.00	0.00	LOG
Single linkage euc	0.20	0.00	0.00	LOG
Single linkage spea	0.20	0.00	0.00	LOG
Single linkage spea	0.20	0.00	0.00	-
DBSCAN eucl	0.20	0.00	0.00	LOG
DBSCAN spea	0.20	0.00	0.00	LOG
DBSCAN spea	0.20	0.00	0.00	-
Single linkage corr	0.20	0.00	0.00	LOG
Single linkage euc	0.20	0.00	0.00	-
Single linkage corr	0.20	0.00	0.00	-
DBSCAN eucl	0.20	0.00	0.00	-
DPSPeuc	0.20	0.00	0.00	LOG

*Table A. 10. **Clustering performance in CIFAR100 data**. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

CIFAR10	Acc	ARI	NMI	Norm
HOT (1)	0.22	0.04	0.07	-
HOMO (1)	0.22	0.04	0.07	-
HOE (1)	0.22	0.04	0.07	-
HIT (1)	0.22	0.04	0.07	-
Kmeans corr	0.20	0.03	0.06	LOG
isoMCLcorr (1)	0.18	0.04	0.07	LOG
Kmeans corr	0.20	0.03	0.06	-
Kmeans eucl	0.20	0.03	0.06	LOG
Kmeans eucl	0.20	0.03	0.07	-
AP eucl	0.20	0.03	0.06	-
MCMCLeuc_dual_Flog (1)	0.18	0.04	0.06	-
AP eucl	0.19	0.03	0.06	LOG
MCMCLeuc_dual_Fsqrt (1)	0.18	0.03	0.06	-
MCL_corr	0.17	0.04	0.06	LOG
MCAP eucl	0.19	0.03	0.05	LOG
MCMCLeuc_hbr_Fsqrt (1)	0.18	0.03	0.05	-
MCMCLeuc_hdr_Fsqrt (1)	0.18	0.03	0.05	-
MCMCLs_eucl	0.18	0.03	0.05	-
MCMCLl_eucl	0.19	0.03	0.05	-
MCMCLeuc_hbr_Flog (1)	0.19	0.03	0.05	-
MCMCLeuc_hdr_Flog (1)	0.19	0.03	0.05	-
isoMCLcorr (1)	0.17	0.03	0.05	-
MCMCLeuc_hbr_Fsqrt (1)	0.17	0.03	0.05	LOG
MCMCLeuc_hdr_Fsqrt (1)	0.17	0.03	0.05	LOG
MCAP eucl	0.17	0.03	0.05	-
MCAP corr	0.18	0.02	0.06	LOG
MCMCLcorr_hbr (1)	0.18	0.03	0.05	-
MCMCLcorr_hdr (1)	0.18	0.03	0.05	-
MCMCL_corr	0.18	0.03	0.05	-

MCAP corr	0.18	0.02	0.05	-
MCMCLs_corr	0.18	0.02	0.05	-
MCMCLs_corr	0.17	0.03	0.05	LOG
MCMCLspea_hdr_Flog (1)	0.16	0.02	0.05	-
MCMCLspea_hdr_Flog (1)	0.16	0.02	0.05	LOG
MCMCLspea_dual_Fsqrt (1)	0.16	0.02	0.05	LOG
MCMCLspea_hdr (1)	0.16	0.03	0.05	-
MCMCLspea_hdr (1)	0.16	0.03	0.05	LOG
MCMCLspea_dual_Flog (1)	0.16	0.02	0.04	LOG
MCMCLcorr_hbr_Flog (1)	0.16	0.02	0.05	LOG
MCMCLspea_dual_Flog (1)	0.16	0.02	0.04	-
MCMCLspea_dual_Fsqrt (1)	0.16	0.02	0.05	-
MCMCLcorr_hdr (1)	0.16	0.02	0.05	LOG
AP corr	0.00	0.03	0.06	-
MCMCLcorr_hdr_Flog (1)	0.16	0.02	0.05	LOG
AP corr	0.00	0.03	0.06	LOG
MCMCLl_corr	0.16	0.02	0.05	LOG
isoMCLeuc (1)	0.16	0.01	0.06	-
isoMCLspear (1)	0.16	0.02	0.04	-
isoMCLspear (1)	0.16	0.02	0.04	LOG
MCMCLspea_dual (1)	0.16	0.02	0.04	-
MCMCLspea_dual (1)	0.16	0.02	0.04	LOG
MCMCLs_eucl	0.15	0.02	0.05	LOG
MCMCLcorr_hbr_Flog (1)	0.16	0.02	0.04	-
MCMCLcorr_hdr_Flog (1)	0.16	0.02	0.04	-
MCMCLspea_hbr_Fsqrt (1)	0.16	0.02	0.04	-
MCMCLspea_hbr_Fsqrt (1)	0.16	0.02	0.04	LOG
MCMCLl_corr	0.16	0.02	0.04	-
MCMCLcorr_dual_Flog (1)	0.15	0.02	0.04	-
MCMCL_corr	0.15	0.02	0.04	LOG
MCMCLcorr_hbr (1)	0.15	0.02	0.04	LOG
MCMCLspea_hdr_Fsqrt (1)	0.16	0.02	0.04	-
MCMCLspea_hdr_Fsqrt (1)	0.16	0.02	0.04	LOG

MCMCLcorr_dual (1)	0.15	0.02	0.04	-
MCMCLspea_hbr_Flog (1)	0.16	0.02	0.04	-
MCMCLspea_hbr_Flog (1)	0.16	0.02	0.04	LOG
MCMCLcorr_dual (1)	0.15	0.02	0.04	LOG
MCMCLcorr_dual_Flog (1)	0.15	0.02	0.04	LOG
MCMCLspea_hbr (1)	0.15	0.02	0.04	-
MCMCLspea_hbr (1)	0.15	0.02	0.04	LOG
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.40	-
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.40	-
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.40	-
MCMCLcorr_hbr_Fsqrt (1)	0.00	0.00	0.40	LOG
MCMCLcorr_dual_Fsqrt (1)	0.00	0.00	0.40	LOG
MCMCLcorr_hdr_Fsqrt (1)	0.00	0.00	0.40	LOG
MCMCLeuc	0.14	0.01	0.04	-
MCMCLeuc_hbr (1)	0.14	0.01	0.04	-
MCMCLeuc_hdr (1)	0.14	0.01	0.04	-
MCL_corr	0.15	0.02	0.03	-
MCMCL_eucl	0.14	0.01	0.04	-
cciMSTcorr	0.14	0.02	0.03	-
MCMCLspea	0.14	0.01	0.03	-
cciMSTspear	0.14	0.01	0.03	-
cciMSTspear	0.14	0.01	0.03	LOG
cciMSTcorr	0.14	0.01	0.03	LOG
MCMCLeuc_hbr_Flog (1)	0.13	0.01	0.03	LOG
MCMCLeuc_hdr_Flog (1)	0.13	0.01	0.03	LOG
MCMCLl_eucl	0.13	0.01	0.03	LOG
MCMCLeuc_dual_Flog (1)	0.13	0.00	0.03	LOG
MCMCLeuc_dual_Fsqrt (1)	0.12	0.00	0.02	LOG
MCMCLeuc_dual (1)	0.12	0.00	0.03	-
MCMCLeuc_hbr (1)	0.10	0.00	0.01	LOG
isoMCLeuc (1)	0.11	0.00	0.01	LOG
MCMCLeuc_hdr (1)	0.10	0.00	0.01	LOG
MCMCLeuc_dual (1)	0.10	0.00	0.01	LOG

MCMCL_eucl	0.10	0.00	0.01	LOG
MCL_eucl	0.10	0.00	0.00	LOG
DPSPeuc	0.10	0.00	0.00	LOG
DPSPcorr	0.10	0.00	0.00	LOG
DPSPeuc	0.10	0.00	0.00	-
DBSCAN corr	0.06	0.00	0.02	LOG
DPSPcorr	0.10	0.00	0.00	-
Single linkage euc	0.10	0.00	0.00	-
DBSCAN eucl	0.10	0.00	0.00	-
DPSPspear	0.10	0.00	0.00	-
DPSPspear	0.10	0.00	0.00	LOG
Single linkage euc	0.10	0.00	0.00	LOG
DBSCAN eucl	0.10	0.00	0.00	LOG
cciMSTeuc	0.10	0.00	0.00	-
MCL_eucl	0.10	0.00	0.00	-
cciMSTeuc	0.10	0.00	0.00	LOG
DBSCAN corr	0.00	0.00	0.00	-

Table A. 11. ***Clustering performance in CIFAR10 data****. Accuracy (Acc), Adjusted Rand Index (ARI), Normalized Mutual Information (NMI), are reported for each clustering method and variant. The distances, specified alongside the methods name and factor (if applied) corresponds to Pearson correlation (corr), Spearman correlation (spea) or Euclidean (euc), the factors (for MC-MCL-variants) are specified as Fsqrt for square root or Flog for logarithm. The last column display the normalization (Norm). The value in parenthesis besides the method's name represents the number of hyperparameters for the clustering method, without taking into consideration the parameters needed to find the number of clusters. Results are ordered according to rank performance (rank value not shown) from the highest (top) to the lowest (bottom).*

BIBLIOGRAPHY

[1] R. Xu and D. WunschII, “Survey of Clustering Algorithms,” *IEEE Trans. Neural Networks*, vol. 16, no. 3, pp. 645–678, May 2005, doi: 10.1109/TNN.2005.845141.

[2] C. Fraley and A. E. Raftery, “Model-Based Clustering, Discriminant Analysis, and Density Estimation,” *J. Am. Stat. Assoc.*, vol. 97, no. 458, pp. 611–631, Jun. 2002, doi: 10.1198/016214502760047131.

[3] M. Halkidi, Y. Batistakis, and M. Vazirgiannis, “On Clustering Validation Techniques,” *J. Intell. Inf. Syst.*, vol. 17, no. 2/3, pp. 107–145, 2001, doi: 10.1023/A:1012801612483.

[4] A. K. Jain, “Data clustering: 50 years beyond K-means,” *Pattern Recognit. Lett.*, vol. 31, no. 8, pp. 651–666, Jun. 2010, doi: 10.1016/J.PATREC.2009.09.011.

[5] G. B. Coleman and H. C. Andrews, “Image segmentation by clustering,” *Proc. IEEE*, vol. 67, no. 5, pp. 773–785, 1979, doi: 10.1109/PROC.1979.11327.

[6] M. Omran, A. P. Engelbrecht, and A. Salman, “Particle swarm optimization method for image clustering,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 19, no. 3, pp. 297–321, 2005, doi: 10.1142/S0218001405004083.

[7] H. P. Ng, S. H. Ong, K. W. C. Foong, P. S. Goh, and W. L. Nowinski, “Medical image segmentation using k-means clustering and improved watershed algorithm,” *Proc. IEEE Southwest Symp. Image Anal. Interpret.*, vol. 2006, pp. 61–65, 2006, doi: 10.1109/ssiai.2006.1633722.

[8] A. Baraldi and P. Blonda, “A survey of fuzzy clustering algorithms for pattern recognition - Part I,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 29, no. 6, pp. 778–785, 1999, doi: 10.1109/3477.809032.

[9] D. Horn and A. Gottlieb, “Algorithm for Data Clustering in Pattern Recognition Problems Based on Quantum Mechanics,” *Phys. Rev. Lett.*, vol. 88, no. 1, p. 4, 2002, doi: 10.1103/PhysRevLett.88.018702.

[10] H. He and Y. Tan, “Automatic pattern recognition of ECG signals using entropy-based adaptive dimensionality reduction and clustering,” *Appl. Soft Comput. J.*, vol. 55, pp. 238–252, 2017, doi: 10.1016/j.asoc.2017.02.001.

[11] R. J. Calantone and C. A. Benedetto, “Clustering product launches by price and launch strategy,” *J. Bus. Ind. Mark.*, vol. 22, no. 1, pp. 4–19, 2007, doi: 10.1108/08858620710722789.

[12] G. Arimond and A. Elfessi, “A clustering method for categorical data in tourism market segmentation research,” *J. Travel Res.*, vol. 39, no. 4, pp. 391–397, 2001, doi: 10.1177/004728750103900405.

[13] S. Ciucci *et al.*, “Enlightening discriminative network functional modules behind Principal Component Analysis separation in differential-omic science studies,” no. October 2016, pp. 1–24, 2017, doi: 10.1038/srep43946.

[14] A. K. Jain, M. N. Murty, and P. J. Flynn, “Data Clustering : A Review,” vol. 31, no. 3, pp. 264–323, 1999, doi: 10.1007/978-3-319-09156-3.

[15] H. Steinhaus, “Sur la division del corps matériels en parties,” *Bull. l’Académie Pol. del Sci. - Cl. III*, vol. IV, no. 12, pp. 801–804, 1956.

[16] J. MacQueen, *Some methods for classification and analysis of multivariate observations*. 1967.

[17] F. Nielsen, “Hierarchical Clustering,” in *Handbook of Cluster Analysis*, no. February, 2016, pp. 195–211.

[18] R. Sibson, “SLINK: An optimally efficient algorithm for the single-link cluster method,” *Comput. J.*, vol. 16, no. 1, pp. 30–34, 1973, doi: 10.1093/comjnl/16.1.30.

[19] L. L. McQuitty, “Elementary Linkage Analysis for Isolating Orthogonal and Oblique Types and Typal Relevancies,” *Educ. Psychol. Meas.*, vol. 17, no. 2, pp. 207–229, Jul. 1957, doi: 10.1177/001316445701700204.

[20] H. K. Seifoddini, “Single linkage versus average linkage clustering in machine cells formation applications,” *Comput. Ind. Eng.*, vol. 16, no. 3, pp. 419–426, 1989, doi: 10.1016/0360-8352(89)90160-5.

[21] J. Sander, “Density-Based Clustering,” in *Encyclopedia of Machine Learning*, Springer Science+Business Media LLC 2011, 2011.

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

[23] A. Karami and R. Johansson, "Choosing DBSCAN Parameters Automatically using Differential Evolution," *Int. J. Comput. Appl.*, vol. 91, no. 7, pp. 1–11, 2014, doi: 10.5120/15890-5059.

[24] K. Khan, S. U. Rehman, K. Aziz, S. Fong, S. Sarasvady, and A. Vishwa, "DBSCAN: Past, present and future," 2014, doi: 10.1109/ICADIWT.2014.6814687.

[25] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science (80-.).*, vol. 344, no. 6191, pp. 1492–1496, 2014, doi: 10.1126/science.1242072.

[26] D. U. Pizzagalli, S. F. Gonzalez, and R. Krause, "A trainable clustering algorithm based on shortest paths from density peaks," *Sci. Adv.*, vol. 5, no. 10, pp. 1–11, 2019, doi: 10.1126/sciadv.aax3770.

[27] B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," *Science (80-.).*, vol. 315, no. 5814, pp. 972–976, 2007, doi: 10.1126/science.1136800.

[28] M. J. Brusco, D. Steinley, J. Stevens, and J. D. Cradit, "Affinity propagation: An exemplar-based tool for clustering in psychological research," *Br. J. Math. Stat. Psychol.*, vol. 72, no. 1, pp. 155–182, 2019, doi: 10.1111/bmsp.12136.

[29] O. Grygorash, Z. Yan, and Z. Jorgensen, "Minimum spanning tree based clustering algorithms," *Proc. - Int. Conf. Tools with Artif. Intell. ICTAI*, pp. 73–81, 2006, doi: 10.1109/ICTAI.2006.83.

[30] C. T. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Trans. Comput.*, 1971, doi: 10.1109/T-C.1971.223083.

[31] X. Lv, Y. Ma, X. He, H. Huang, and J. Yang, "CciMST: A Clustering Algorithm Based on Minimum Spanning Tree and Cluster Centers," *Math. Probl. Eng.*, vol. 2018, 2018, doi: 10.1155/2018/8451796.

[32] S. van Dongen, "Graph clustering by flow simulation," *Graph Stimul. by flow Clust.*, 2000, doi: 10.1016/j.cosrev.2007.05.001.

[33] M. A. Kramer, "Nonlinear principal component analysis using

autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, 1991, doi: 10.1002/aic.690370209.

[34] X. Peng, H. Zhu, J. Feng, C. Shen, H. Zhang, and J. T. Zhou, "Deep Clustering with Sample-Assignment Invariance Prior," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 11, pp. 4857–4868, 2020, doi: 10.1109/TNNLS.2019.2958324.

[35] M. Girvan and M. E. J. Newman, "Community structure in social and biological networks," *PNAS*, vol. 99, no. 12, pp. 7821–7826, 2002, doi: 10.1073/pnas.122653799.

[36] A. Lancichinetti, J. Saramäki, M. Kivelä, and S. Fortunato, "Characterizing the community structure of complex networks," *PLoS One*, 2010, doi: 10.1371/journal.pone.0011976.

[37] M. Rosvall and C. T. Bergstrom, "Multilevel compression of random walks on networks reveals hierarchical organization in large integrated systems," *PLoS One*, vol. 6, no. 4, p. e18209, 2011, doi: 10.1371/journal.pone.0018209.

[38] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *J. Stat. Mech Theory Exp.*, vol. 2008, no. 10, p. 10008, 2008, doi: 10.1088/1742-5468/2008/10/P10008.

[39] A. Lancichinetti and S. Fortunato, "Community detection algorithms: A comparative analysis," *Phys. Rev. E*, vol. 80, no. 5, p. 056117, 2009, doi: 10.1103/PhysRevE.80.056117.

[40] Z. Yang, R. Algesheimer, and C. J. Tessone, "A Comparative Analysis of Community Detection Algorithms on Artificial Networks," *Sci. Rep.*, vol. 6, p. 30750, 2016, doi: 10.1038/srep30750.

[41] G. K. Orman and V. Labatut, "A Comparison of Community Detection Algorithms on Artificial Networks," in *Discovery science*, 2009, pp. 242–256.

[42] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998, doi: 10.1038/30918.

[43] D. A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes," *Proc. IRE*, 1952, doi: 10.1109/JRPROC.1952.273898.

[44] M. Rosvall and C. T. Bergstrom, "Maps of random walks on complex

networks reveal community structure," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 105, no. 4, pp. 1118–1123, 2008, doi: 10.1073/pnas.0706851105.

[45] L. van der Maaten, E. Postma, and J. van den Herik, "Dimensionality Reduction: A Comparative Review," *J. Mach. Learn. Res.*, vol. 10, no. February, pp. 1–41, 2009, doi: 10.1080/13506280444000102.

[46] I. T. Jolliffe, "Principal Component Analysis," *Springer Ser. Stat.*, vol. 98, p. 487, 2002, doi: 10.1007/b98835.

[47] Ringnér, "What is principal component analysis?," *Nat. Biotechnol.*, vol. 26, no. 3, pp. 303–304, Mar. 2008, doi: doi:10.1038/nbt0308-303.

[48] R. N. Shepard, "The analysis of proximities: Multidimensional scaling with an unknown distance function. I.," *Psychometrika*, vol. 27, no. 2, pp. 125–140, Jun. 1962, doi: 10.1007/BF02289630.

[49] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, Mar. 1964, doi: 10.1007/BF02289565.

[50] E. W. Beals, "Bray-curtis ordination: An effective strategy for analysis of multivariate ecological data," in *Advances in Ecological Research*, vol. 14, no. C, 1984, pp. 1–55.

[51] C. Durán *et al.*, "Nonlinear machine learning pattern recognition and bacteria-metabolite multilayer network analysis of perturbed gastric microbiome," *Nat. Commun.*, no. 12, p. 1926, 2021, doi: 10.1038/s41467-021-22135-x.

[52] C. V. Cannistraci, T. Ravasi, F. M. Montevecchi, T. Ideker, and M. Alessio, "Nonlinear dimension reduction and clustering by minimum curvilinearity unfold neuropathic pain and tissue embryological classes," *Bioinformatics*, vol. 26, no. 18, pp. 531–539, 2010, doi: 10.1093/bioinformatics/btq376.

[53] C. V. Cannistraci, G. Alanis-Lobato, and T. Ravasi, "Minimum curvilinearity to enhance topological prediction of protein interactions by network embedding," *Bioinformatics*, vol. 29, no. 13, pp. 199–209, 2013, doi: 10.1093/bioinformatics/btt208.

[54] M. Boguñá, D. Krioukov, and K. C. Claffy, "Navigability of complex networks," *Nat. Phys.*, vol. 5, no. 1, pp. 74–80, 2008, doi: 10.1038/nphys1130.

[55] J. Kleinberg, "Small-world phenomena and the dynamics of

information," *Adv. Neural Inf. Process. Syst.*, pp. 1–14, 2002, doi: 10.7551/mitpress/1120.003.0060.

[56] F. Papadopoulos, M. Kitsak, M. A. Serrano, M. Boguñá, and D. Krioukov, "Popularity versus similarity in growing networks," *Nature*, vol. 489, no. 7417, pp. 537–540, 2012, doi: 10.1038/nature11459.

[57] C. V. Cannistraci, T. Ravasi, F. M. Montevecchi, T. Ideker, and M. Alessio, "Nonlinear dimension reduction and clustering by Minimum Curvilinearity unfold neuropathic pain and tissue embryological classes," *Bioinformatics*, vol. 26, pp. i531–i539, 2010.

[58] A. Muscoloni and C. V. Cannistraci, "Navigability evaluation of complex networks by greedy routing efficiency," vol. 116, no. 5, pp. 1468–1469, 2019, doi: 10.1073/pnas.1817880116.

[59] A. Muscoloni and C. V. Cannistraci, "Minimum curvilinear automata with similarity attachment for network embedding and link prediction in the hyperbolic space," *ArXiv:1802.01183*, 2018.

[60] A. Muscoloni, J. M. Thomas, S. Ciucci, G. Bianconi, and C. V. Cannistraci, "Machine learning meets complex networks via coalescent embedding in the hyperbolic space," *Nat. Commun.*, vol. 8, no. 1, p. 1615, 2017.

[61] C. V. Cannistraci and A. Muscoloni, "Latent Geometry Inspired Graph Dissimilarities Enhance Affinity Propagation Community Detection in Complex Networks," *ArXiv: 1804.04566*, vol. 20:063022, 2018, [Online]. Available: http://arxiv.org/abs/1804.04566.

[62] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction.," *Science*, vol. 290, pp. 2319–23, 2000, doi: 10.1126/science.290.5500.2319.

[63] N. X. Vinh, J. Epps, and J. Bailey, "Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance," *J. Mach. Learn. Res.*, vol. 11, pp. 2837–2854, Dec. 2010.

[64] L. Danon, A. Diaz-Guilera, J. Duch, and A. Arenas, "Comparing community structure identification," *J. Stat. Mech. Theory Exp.*, vol. P09008, pp. 1–10, 2005.

[65] L. Hubert and P. Arabie, "Comparing partitions," *J. Classif.*, vol. 2, no. 1, pp. 193–218, Dec. 1985, doi: 10.1007/BF01908075.

[66] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” *J. Am. Stat. Assoc.*, vol. 66, no. 336, pp. 846–850, 1971, doi: 10.1080/01621459.1971.10482356.

[67] M. Á. Serrano, D. Krioukov, and M. Boguñá, “Self-similarity of complex networks and hidden metric spaces,” *Phys. Rev. Lett.*, vol. 100, no. 7, pp. 1–4, 2008, doi: 10.1103/PhysRevLett.100.078701.

[68] D. Krioukov, F. Papadopoulos, M. Kitsak, A. Vahdat, and M. Boguñá, “Hyperbolic geometry of complex networks,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 82, no. 3, p. 036106, 2010, doi: 10.1103/PhysRevE.82.036106.

[69] M. Jalili and M. Perc, “Information cascades in complex networks,” *J. Complex Networks*, vol. 5, no. 5, pp. 665–693, 2017, doi: 10.1093/comnet/cnx019.

[70] A. Clauset, C. Rohilla Shalizi, and M. E. J. Newman, “Power-Law Distributions in Empirical Data,” *SIAM Rev.*, vol. 51, no. 4, pp. 661–703, 2009, doi: 10.1214/13-AOAS710.

[71] W. W. Zachary, “An Information Flow Model for Conflict and Fission in Small Groups,” *J. Anthropol. Res.*, vol. 33, no. 4, pp. 452–473, 1977, doi: 10.2307/3629752.

[72] R. Cross and A. Parker, *The Hidden Power of Social Networks*. Harvard Business School Press, 2004.

[73] L. A. Adamic and N. Glance, “The Political Blogosphere and the 2004 U.S. Election: Divided They Blog,” *LinkKDD 2005*, pp. 36–43, 2005.

[74] A. Muscoloni and C. V. Cannistraci, “A nonuniform popularity-similarity optimization (nPSO) model to efficiently generate realistic complex networks with communities,” *New J. Phys.*, vol. 20, p. 052002, 2018, doi: https://doi.org/10.1088/1367-2630/aac06f.

[75] A. Muscoloni and C. V. Cannistraci, “Leveraging the nonuniform PSO network model as a benchmark for performance evaluation in community detection and link prediction,” *New J. Phys.*, vol. 20, no. 063022, p. 063022, 2018.

[76] F. Paroni Sterbini *et al.*, “Effects of Proton Pump Inhibitors on the Gastric Mucosa-Associated Microbiota in Dyspeptic Patients.,” *Appl. Environ. Microbiol.*, vol. 82, no. 22, pp. 6633–6644, Nov. 2016, doi: 10.1128/AEM.01437-16.

[77] Z. Li *et al.*, “Effect of long-term proton pump inhibitor administration

on gastric mucosal atrophy: A meta-analysis," *Saudi Journal of Gastroenterology*. 2017, doi: 10.4103/sjg.SJG_573_16.

[78] S. C. Nasser, M. Slim, J. G. Nassif, and S. M. Nasser, "Influence of proton pump inhibitors on gastritis diagnosis and pathologic gastric changes," *World J. Gastroenterol.*, vol. 21, no. 15, pp. 4599–4606, 2015, doi: 10.3748/wjg.v21.i15.4599.

[79] V. G. Sigillito, S. P. Wing, L. V Hutton, and K. B. Baker, "CLASSIFICATION OF RADAR RETURNS FROM THE," vol. 10, no. 3, 1989.

[80] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998, doi: 10.1109/5.726791.

[81] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[82] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," 2009.

[83] A. Kolesnikov *et al.*, "Big Transfer (BiT): General Visual Representation Learning," 2020, pp. 491–507.

[84] Y. Huang *et al.*, "GPipe: Efficient training of giant neural networks using pipeline parallelism," *Adv. Neural Inf. Process. Syst.*, vol. 32, no. NeurIPS, 2019.

[85] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[86] M. Wistuba, A. Rawat, and T. Pedapati, "A survey on neural architecture search," *arXiv*, 2019.

[87] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "Autoaugment: Learning augmentation strategies from data," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2019-June, no. Section 3, pp. 113–123, 2019, doi: 10.1109/CVPR.2019.00020.

[88] N. Nayman, A. Noy, T. Ridnik, I. Friedman, R. Jin, and L. Zelnik-Manor, "XNAS: Neural architecture search with expert advice," *arXiv*, 2019.

[89] N. H. Phong and B. Ribeiro, “Rethinking Recurrent Neural Networks and other improvements for image classification,” *arXiv*, pp. 1–16, 2020.

[90] D. Hric, R. K. Darst, and S. Fortunato, “Community detection in networks: Structural communities versus ground truth,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 90, no. 6, 2014, doi: 10.1103/PhysRevE.90.062805.

[91] A. Muscoloni and C. V. Cannistraci, “Minimum curvilinear automata with similarity attachment for network embedding and link prediction in the hyperbolic space,” Feb. 2018.

[92] S. Athanassopoulos, C. Kaklamanis, I. Laftsidis, and E. Papaioannou, “An Experimental Study of Greedy Routing Algorithms,” *Proc. Int. Conf. High Perform. Comput. Simul.*, pp. 150–156, 2010, doi: 10.1109/HPCS.2010.5547143.

[93] M. E. J. Newman and M. Girvan, “Finding and evaluating community structure in networks,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 69, no. 2 2, pp. 1–15, 2004, doi: 10.1103/PhysRevE.69.026113.

[94] K. Fukunaga, *Introduction to Statistical Pattern Recognition*. Academic Press, 2013.

[95] A. Muscoloni and C. V. Cannistraci, “Rich-clubness test: How to determine whether a complex network has or doesn’t have a rich-club?,” *arXiv*, 2017.

[96] D. Krioukov, F. Papadopoulos, A. Vahdat, and M. Boguñá, “Curvature and temperature of complex networks,” *Phys. Rev. E - Stat. Nonlinear, Soft Matter Phys.*, vol. 80, no. 3, 2009, doi: 10.1103/PhysRevE.80.035101.

[97] M. A. Tanner and W. H. Wong, “The calculation of posterior distributions by data augmentation,” *J. Am. Stat. Assoc.*, vol. 82, no. 398, pp. 528–540, 1987, doi: 10.1080/01621459.1987.10478458.