Pattern Recognition 45 (2012) 416–433

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

SpectralCAT: Categorical spectral clustering of numerical and nominal data

Gil David [a,*], Amir Averbuch [b]

[a] Department of Mathematics, Program in Applied Mathematics, Yale University, New Haven, CT 06510, USA
[b] School of Computer Science, Tel-Aviv University, Tel-Aviv 69978, Israel

ARTICLE INFO

Article history:
Received 11 October 2010
Received in revised form
2 July 2011
Accepted 3 July 2011
Available online 18 July 2011

Keywords:
Spectral clustering
Diffusion Maps
Categorical data clustering
Dimensionality reduction

ABSTRACT

Data clustering is a common technique for data analysis, which is used in many fields, including machine learning, data mining, customer segmentation, trend analysis, pattern recognition and image analysis. Although many clustering algorithms have been proposed, most of them deal with clustering of one data type (numerical or nominal) or with mix data type (numerical and nominal) and only few of them provide a generic method that clusters all types of data. It is required for most real-world applications data to handle both feature types and their mix. In this paper, we propose an automated technique, called *SpectralCAT*, for unsupervised clustering of high-dimensional data that contains numerical or nominal or mix of attributes. We suggest to automatically transform the high-dimensional input data into categorical values. This is done by discovering the optimal transformation according to the Calinski–Harabasz index for each feature and attribute in the dataset. Then, a method for spectral clustering via dimensionality reduction of the transformed data is applied. This is achieved by automatic non-linear transformations, which identify geometric patterns in the data, and find the connections among them while projecting them onto low-dimensional spaces. We compare our method to several clustering algorithms using 16 public datasets from different domains and types. The experiments demonstrate that our method outperforms in most cases these algorithms.

1. Introduction

Clustering means an unsupervised classification of observed data into different subsets (clusters) such that the objects in each subset are similar while objects in different subsets are dissimilar. Data clustering is a common technique for data analysis. It is used in many fields such as machine learning, data mining, customer segmentation, trend analysis, pattern recognition and image analysis. Although many clustering algorithms have been proposed, most of them were designed to deal with clustering of only specific type of data (numerical or nominal). Most of the methods, which were designed to process numerical data, are incapable to process nominal data and vice versa. This is due to the differences between the nature of different data types. For example, numerical data consists of numerical attributes whose values are represented by continuous variables. Finding similarity between numerical objects usually relies on a common distance measure such as Euclidean distance. The problem of clustering categorical data is different since it consists of data with nominal attributes whose values neither have a natural ordering nor a common scale. As a result, finding similarities between nominal objects by using common distance measures, which are used for processing numerical data, is not applicable here. On the other hand, common distance measures for categorical data, such as Hamming distance, are not applicable for numerical data. Moreover, real applications data have to handle mixed types of attributes such as numerical and nominal data that reside together. Several methods for clustering mix type of data (numerical and nominal) were proposed. Their strength is usually emerged when clustering mix data and not when clustering a single type of data. For example, *k*-prototypes algorithm [1] was designed to cluster mix data by integrating a numerical clustering algorithm (*k*-means [2]) and a categorical clustering algorithm (*k*-modes [3]). Its strength emerges when clustering the mixed data but when it clusters only one type of data it does not improve the performance of its building blocks algorithms. Only few methods provide generic algorithms that clusters all types of data (numerical or nominal or mix).

In this paper, we propose an algorithm that provides a generic solution for all data types. We present an automated technique that performs an unsupervised clustering of high-dimensional data with anywhere from four to thousands of dimensions, which contains either numerical or nominal or mix of numerical and nominal attributes. We suggest to automatically transform the high-dimensional input data into categorical values. This is done

* Corresponding author. Tel.: +1 203 432 4345.
E-mail addresses: gil.david@yale.edu (G. David), amir@math.tau.ac.il (A. Averbuch).

0031-3203/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2011.07.006

automatically by discovering the optimal transformation for each feature and attribute in the dataset according to the Calinski–Harabasz index [4]. Then, a spectral clustering via dimensionality reduction of the transformed data is applied to it, which is a non-linear transformation that identifies geometric patterns in the data and finds connections among them while projecting them onto low-dimensional spaces.

We provide a framework that is based upon transforming the data to common nominal scales and then applying diffusion processes for finding meaningful geometric descriptions in these datasets which can be uniform and heterogenous together. We show that the eigenfunctions of Markov matrices are used to construct diffusion maps that generate efficient representations of complex geometric structures that enable to perform efficient data mining tasks. The associated family of diffusion distances, obtained by iterating a Markov matrix, defines multiscale (coarse graining) geometries that prove to be useful for data mining methodologies and for learning of large uniform, heterogeneous and distributed datasets. We extend the diffusion process methodology to handle categorical data and we propose a method to construct an adaptive kernel. The proposed framework relates the spectral properties of Markov processes to their geometric counterparts and it unifies ideas arising in a variety of contexts such as machine learning, spectral graph theory and kernel methods. It enables to reduce the dimensionality of the data that is embedded into low-dimensional subspaces where all the requested and sought after information lie. This reduction does not violate the integrity and the coherency of the embedded data.

The proposed algorithm has two sequential steps:

1. *Data transformation by categorization:* Conversion of the high-dimensional data to common categorical scales. This is done automatically by discovering the optimal transformation for each feature and attribute in the dataset. Nominal features are already in categorical scales and therefore are not transformed.
2. *Spectral clustering of the categorized data:* Study and analysis of the behavior of the dataset by projecting it onto a low-dimensional space. This step clusters the data.

We compare our method with several clustering algorithms by using 16 public datasets from different domains and types. The experiments demonstrate that our method outperforms in most case these algorithms. The experiments show that SpectralCAT is generic and fits different data types from various domains including high-dimensional data.

The paper is organized as follows. Section 2 describes related clustering algorithms. Section 3 describes a method for data transformation by categorization. It describes how to transform data into categorical scales. The method for spectral clustering of the transformed data is described in Section 4. Section 5 provides a detailed description of the SpectralCAT algorithm for clustering numerical data or nominal data or mixed data and a description of SpectralCAT++ that extends it. Section 6 compares between our method and other clustering algorithms.

2. Related work

Many unsupervised clustering algorithms have been proposed over the years. Most of them cluster numerical data, where the data consists of numerical attributes whose values are represented by continuous variables. Finding similarity between numerical objects usually relies on common distance measures such as Euclidean, Manhattan, Minkowski and Mahalanobis distances to name some. A comprehensive survey of various clustering algorithms is given in [5].

k-means [2] is one of the most commonly used clustering algorithm. It was designed to cluster numerical data in which each cluster has a center called the mean. *k*-means finds the centers that minimize the sum of squared distances from each data point toits closest center. The *k*-means algorithm is classified as either partitioner or non-hierarchical clustering method. Finding an exact solution to the *k*-means problem for an arbitrary input is NP-hard. But the *k*-means algorithm finds quickly an approximated solution that can be arbitrarily bad with respect to the objective function in comparison to optimal clustering. The performance of *k*-means is highly dependent on the initialization of the centers. Furthermore, it does not perform effectively on high-dimensional data.

k-means++ [6] algorithm chooses the initial values for the *k*-means clustering. It was proposed as an approximation algorithm for the NP-hard *k*-means problem. *k*-means++ specifies a procedure to initialize the cluster centers before proceeding with the standard *k*-means optimization iterations. With the *k*-means++ initialization, the algorithm is guaranteed to find a solution that is $O(\log k)$ competitive to the optimal *k*-means solution.

The LDA-Km algorithm [7] combines linear discriminant analysis (LDA) and *k*-means clustering [2] to adaptively select the most discriminative subspace. It uses *k*-means clustering to generate class labels and uses LDA to perform subspace selection. The clustering process is integrated with the subspace selection process and the data is then simultaneously clustered while feature subspaces are selected.

The localized diffusion folders (LDF) methodology [8] performs hierarchical clustering and classification of high-dimensional datasets. The diffusion folders are multi-level data partitioning into local neighborhoods that are generated by several random selections of data points and folders in a diffusion graph and by defining local diffusion distances between them. This multi-level partitioning defines an improved localized geometry of the data and a localized Markov transition matrix that is used for the next time step in the diffusion process. The result of this clustering method is a bottom-up hierarchical clustering of the data while each level in the hierarchy contains localized diffusion folders of folders from the lower levels. This methodology preserves the local neighborhood of each point while eliminating noisy connections between distinct points and areas in the graph.

An agglomerative hierarchical algorithm is proposed in BIRCH [9]. It is used for clustering large numerical datasets in Euclidean spaces. BIRCH performs well when clusters have identical sizes and their shapes are either convex or spherical. However, it is affected by the input order of the data and it may not perform well when clusters have either different sizes or non-spherical shapes.

CURE [10] is another method that clusters numerical datasets using hierarchical agglomerative algorithm. CURE can identify non-spherical shapes in large databases that have different sizes. It uses a combination of random sampling and partitioning in order to process large databases. Therefore, it is affected by the random sampler performance.

A density-based clustering algorithm is proposed in DBSCAN [11]. This method is used to discover arbitrarily shaped clusters. DBSCAN is sensitive to its parameters, which in turn, are difficult to determine. Furthermore, DBSCAN does not perform any pre-clustering and it is executed directly on the entire database. As a result, DBSCAN can incur substantial I/O costs in processing large databases.

DIANA [12] is a divisive hierarchical algorithm that applies to all datasets that can be clustered by hierarchical agglomerative algorithms. Since the algorithm uses the largest dissimilarity between two objects in a cluster such as the diameter of the cluster, it is sensitive to outliers.

Several clustering algorithms for categorical data have been proposed in recent years. The *k*-modes algorithm [3] emerged from the *k*-means algorithm. It was designed to cluster categorical datasets. The main idea of the *k*-modes algorithm is to specify the number of clusters and then to select *k* initial modes, followed by allocating every object to its nearest mode. The algorithm minimizes the dissimilarity of the objects in a cluster with respect to its mode.

The inter-attribute and intra-attribute summaries of a database are constructed in CACTUS [13]. Then, a graph, called the similarity graph, is defined according to these summaries. Finally, the clusters are found with respect to these graphs.

COOLCAT [14] clusters categorical attributes using an incremental partition clustering algorithm to minimize the expected entropy. First, it finds a suitable set of clusters from a sample from the entire dataset. Then, it assigns the remaining records to a suitable cluster. The major drawback of this algorithm is that the order of the processing data points has a definite impact on the clustering quality.

ROCK [15] is a hierarchical agglomerative clustering algorithm that employs links to merged clusters. A link between two categorical objects is defined as the number of common neighbors. ROCK uses a similarity measure between links to measure the similarity between two data points and between two clusters.

STIRR [16] is an iterative method that is based on non-linear dynamic systems from multiple instances of weighted hypergraphs (known as basins). Each attribute value is represented by a weighted vertex. Two vertices are connected when the attribute values, which they represent, co-occur at least once in the dataset. The weights are propagated in each hypergraph until the configuration of the weights in the main basin converges to a fixed point.

More recently, some kernel methods for clustering were proposed. A clustering method, which uses support vector machine, is given in [17]. The data points are mapped by a Gaussian kernel to a high-dimensional feature space, where it searches the minimal enclosing sphere. When this sphere is mapped back to data space, it can be separated into several components where each encloses a separate cluster.

A method for unsupervised partitioning of a data sample, which estimates the possible number of inherent clusters that generate the data, is described in [18]. It exploits the notion that performing a non-linear data transformation into some high dimensional feature space increases the probability for linear separability between the patterns within the transformed space. Therefore, it simplifies the associated data structure. It shows that the eigenvectors of a kernel matrix, which define an implicit mapping, provide means to estimate the number of clusters inherent within the data. A computational iterative procedure is presented for the subsequent feature space that partitions the data.

A kernel clustering scheme, which is based on *k*-means for large datasets, is proposed in [19]. It introduces a clustering scheme which changes the clustering order from a sequence of samples to a sequence of kernels. It employs a disk-based strategy to control the data.

Another kernel clustering scheme, which is based on *k*-means, is proposed in [20]. It uses a kernel function, which is based on Hamming distance, to embed categorical data in a constructed feature space where the clustering takes place.

In recent years, spectral clustering has become a popular clustering technique. It is simple to implement, can be solved efficiently and very often outperforms traditional clustering algorithms. A comprehensive survey of spectral clustering methods is given in [21]. Spectral clustering refers to a class of techniques which rely on the eigenstructure of a matrix to partition points into disjoint clusters, according to their components in the top few singular vectors of the matrix, with points in the same cluster having high similarity and points in different clusters having low similarity [22].

A spectral clustering method, which is based on *k* singular vectors, is presented in [23]. Given a matrix *A*, where the rows of *A* are points in a high-dimensional space, then the subspace defined by the top *k* right singular vectors of *A* is the rank-*k* subspace that best approximates *A*. This spectral algorithm projects all the points onto this subspace. Each singular vector then defines a cluster. To obtain a clustering, each projected point is mapped into a cluster defined by the singular vector that is closest in angle to it.

A method for using the first non-trivial eigenvector to approximate the optimal normalized cut of a graph was presented in [24]. This method constructs a similarity graph of the data points, computes its unnormalized Laplacian and then computes the first *k* generalized eigenvectors. The rows of these eigenvectors are clustered into *k* clusters via *k*-means.

A similar spectral clustering method is presented in [25]. This method constructs a similarity graph of the data points, computes its normalized Laplacian and then computes the first *k* eigenvectors. The rows of these eigenvectors are normalized to have unit length and then they are clustered into *k* clusters via *k*-means.

Similar to *k*-modes algorithm, mean shift offers a nonparametric technique for the analysis of a complex multimodal feature space for numerical data. Mean shift is a simple iterative procedure that shifts each data point to the average of data points in its neighborhood [26–28]. Adaptive mean shift clustering method [29] is an unsupervised kernel density estimation method that automatically selects bandwidth of every sample point. Ref. [26] suggested the density gradient-based approach as a method for cluster analysis. It is derived from kernel density gradient estimator and it selects a kernel bandwidth that guarantee asymptotic unbiasedness of the estimate.

The *k*-prototypes algorithm [1] integrates the *k*-means [2] and the *k*-modes [3] algorithms by defining a combined dissimilarity measure to enable clustering of mixed numerical and categorical attributes.

Gower [30] introduced a similarity index that measures the similarity between two samples whose attributes are numerical or categorical or mixed type of data. Applying the *k*-means algorithm to the Gower similarity index enables clustering of mixed attributes.

Many of the methods do not provide a generic algorithm for clustering all types of data (numerical or nominal or mix). Some of them were designed to deal with only one type of data. Some were designed to deal only with mix data but not with a single type. Moreover, some of them fail to handle high-dimensional data. Our proposed method is generic and it is designed to deal with all types of data (numerical or nominal or mix) that are also high-dimensional.

3. Data transformation by categorization

Categorical data contains data with nominal attributes whose values neither have a natural ordering nor an inherent order. The variables of categorical data are measured by nominal scales. Numerical data consists of data with numerical attributes whose values are represented by continuous variables. The variables of numerical data are measured by an interval scales or by a ratio of scales. Many algorithms, which have been developed by the machine learning community, focus on learning in nominal feature spaces. Many real-world classification tasks deal with continuous features. These algorithms could not be applied to these classification tasks unless the continuous features are first discretized. Continuous variable discretization has received significant attention in the machine learning community [31]. Other reasons for variable discretization, aside from the algorithmic requirements, include speed increase of induction algorithms [32] and viewing General Logic Diagrams [33] of the induced classifier. Data discretization is part of any data mining and it has particular importance especially in mining of numerical data. It is used to reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals also called quantization. Interval labels can then be used to replace actual data values. Replacing numerical values of a continuous attribute by a small number of interval labels thereby reduces and simplifies the original data and make it workable. This leads to a concise, easy to use, knowledge-level representation of mining results. There are many methods for data discretization such as entropy-based discretization, cluster analysis, binning, histogram analysis and discretization by intuitive partitioning. Entropy is one of the most commonly used. Entropy-based discretization is a supervised splitting technique. It explores class distribution information in its calculation and determination of split-points. To discretize a numerical attribute A, the method selects the value of A that has the minimum entropy as a split-point, and recursively partitions the resulting intervals to arrive at a hierarchical discretization. Cluster analysis is also a popular data discretization method. A clustering method can be applied to discretize a numerical attribute A by partitioning the values of A into clusters. Clustering takes the distribution of A into consideration, as well as the closeness of data points. Therefore, it is able to produce high-quality discretization results. These discretization techniques transform each attribute independently by changing the original data and the relationships between the original attributes, but this preprocessing step contributes toward the success of the mining process [34]. There is always a trade off between transforming the original data as a preprocessing step and mining the original data. There are advantages and disadvantages for each approach. In order to understand the effect of the categorization step in our algorithm, we compare in Section 6 the results obtained with and without this preprocessing step.

This section describes how to categorize numerical data by transforming into categorical data. Our method for data transformation by automatic categorization normalizes and discretizes the original data. It is similar to discretization by cluster analysis that was mentioned above. It defines a smart way to analyze the clusters and discretizes each attribute. It provides a measure to evaluate the performance of the discretization and selects the optimal clusters to be discretized.

Let $X=\{x_1,\dots,x_m\}$ be a set of n-dimensional numerical points in $\mathbb{R}^n$ where $x_i=\{x_i^1,\dots,x_i^n\}$, $i=1,\dots,m$ and x_i is a row vector. In order to transform X from numerical scales into categorical scales, each component, which describes one feature from a single scale only, is transformed. Formally, let $X^l=\{x_1^l,\dots,x_m^l\}$ be a single feature in X. X^l is transformed into the categorical values $\hat{X}^l=\{\hat{x}_1^l,\dots,\hat{x}_m^l\}$, $l=1,\dots,n$. This way, each point $x_i\in X$, $i=1,\dots,m$ is transformed into $\hat{x}_i=\{\hat{x}_i^1,\dots,\hat{x}_i^n\}$ and X is transformed into $\hat{X}=\{\hat{x}_1,\dots,\hat{x}_m\}$. At the end of this process, each component $l=1,\dots,n$, from the transformed data $\hat{X}$ has its own set of categories.

As described above, categorization of numerical data is obtained by categorization of each feature. The categorization process of one feature, which is represented by a numerical variable, requires to know in advance the optimal number of categories in the transformed data. Since every numerical variable has it own numerical scale and its own natural order, it also has its own number of categories. Discovery of the known number of clusters is critical for the success of the categorization process. Since we have no information on this optimal number of categories for each numerical variable, then, it has to be automatically discovered on-the-fly. Section 3.1 describes an automatic categorization process that reveals the optimal number of categories.

3.1. Automatic categorization and discovery of the optimal number of categories

An automatic categorization process of numerical data of one feature takes place by the application of a clustering method to the data while computing the validity index (defined below) of the clusters. This process is repeated (iterated) using increasing number of categories in each iteration. This process is terminated when the first local maxima is found. The number of categories, which reaches the best validity index (the local maxima), is chosen as the optimal number of categories.

In Section 3, we defined X^l as a single dimension in X, where $l=1,\dots,n$. Since the process of categorization is applicable to any number of dimensions, we can generalize the definition of X^l to be q-dimensional (features) in X. In this case, l corresponds to q-tuple features, where each feature is from the range $1,\dots,n$. This enables to transform several features into a single categorical feature. Let f_k be a clustering function that associates each $x_i^l\in X^l$, $i=1,\dots,m$, to one of the k clusters in C^l, where $C^l=\{c_1^l,\dots,c_k^l\}$ and $c_j^l=\{x_i^l|f_k(x_i^l)\in c_j^l, i=1,\dots,m, c_j^l\in C^l\}$. k is unknown at this point.

The total sum of squares of X^l is defined as $S^l=\sum_{i=1}^m (x_i^l-\overline{x}^l)(x_i^l-\overline{x}^l)^T$, where $\overline{x}^l=(1/m)\sum_{i=1}^m x_i^l$. The within-cluster sum of squares, with k clusters, is defined as $S_w^l(k)=\sum_{j=1}^k \sum_{x^l\in c_j^l}(x^l-\mu_j^l)(x^l-\mu_j^l)^T$, where $\mu_j^l=(1/|c_j^l|)\sum_{x^l\in c_j^l}x^l$. $S_w^l(k)$ denotes the sum of deviations from the centers of their associated clusters of all the points in the data. A good clustering method should achieve a small $S_w^l(k)$.

The between-cluster sum of squares, with k clusters, is defined as $S_b^l(k)=\sum_{j=1}^k |c_j^l|(\mu_j^l-\overline{x}^l)(\mu_j^l-\overline{x}^l)^T$. $S_b^l(k)$ denotes the sum of the weighted distances from the data center of all the centers of the k clusters. A good clustering method should achieve a large $S_b^l(k)$.

It is easily seen (Appendix B, point 1) that the total sum of squares (S^l) equals to the sum of the between-cluster sum of squares ($S_b^l(k)$) and the within-cluster sum of squares ($S_w^l(k)$). Therefore,

$$S^l=S_w^l(k)+S_b^l(k). \tag{1}$$

In order to measure the validity index $S_{k,m}^l$ of the clustering, we use the Calinski–Harabasz index [4]:

$$S_{k,m}^l=\frac{(m-k)S_b^l(k)}{(k-1)S_w^l(k)}. \tag{2}$$

The validity index maximizes $S_{k,m}^l$ on k clusters. $S_{k,m}^l$ denotes the variance ratio criterion.

The optimal number of categories, denoted by k_{best}^l, is obtained by the application of the clustering method f_k to X^l while calculating the corresponding validity index $S_{k,m}^l$ for X^l, $k=2,3,\dots$. We define k_{best}^l to be the smallest k for which the smoothed $S_{k,m}^l$ has a local maximum. We analyze the smoothed $S_{k,m}^l$ that does not have oscillatory peaks (noise) as it is in the source $S_{k,m}^l$. The smoothness is achieved by the application of a 1D moving average filter, with a span of 5, to $S_{k,m}^l$. The filter is a direct form II transposed [35] implementation of the standard difference equation. When the first local maxima is found then this process is terminated.

Fig. 1 shows an example of the application of this process to one of the features of the Segmentation dataset [36]. In this example, f_k, $k=2,\dots,100$ was applied and the corresponding validity indexes were calculated. The first local maxima was found at $k=9$ and therefore $k_{best}=9$.

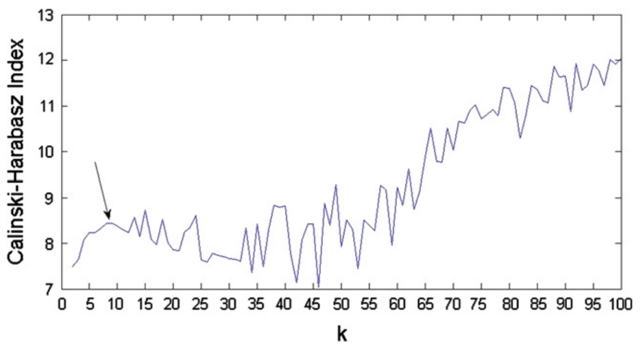

Fig. 1. The validity indexes for $k=2,\dots,100$. The first local maxima is marked by the arrow.

The justification of the validity index is given in Appendix A. Therefore, the application of the clustering method $f_{k^l_{best}}$ to X^l associates each $x^l_i \in X^l$, $i=1,\dots,m$, to one of the clusters $c^l_j \in C^l$, $j=1,\dots,k^l_{best}$, and the corresponding categorical value of x^l_i is determined as j. This automatic categorization process reveals the optimal number of categories in X^l and transforms the numerical data into categorical data. This process is repeated for each feature $l, l=1,\dots,n$ in X.

At the end of this process, the data is transformed into common categorical scales. Since in most cases the dataset contains high-dimensional data points, it is required to study and analyze the dataset by projecting it onto a low-dimensional subspaces where all the requested and sought after information lie to reveal global geometric information. Section 4 deals with the dimensionality reduction step. In Section 6, we show the effect of data clustering after the transformation step only (without the dimensionality reduction step). This comparison will emphasize the importance of the second step in our proposed method.

4. Spectral clustering of categorized data

Section 3 described a method for an automatic data categorization. This section describes how to cluster the transformed high-dimensional data via spectral clustering. The diffusion maps framework [37,38] and its inherent diffusion distances provide a method for finding meaningful geometric structures in datasets. In most cases, the dataset contains high-dimensional data points in $\mathbb{R}^n$. The diffusion maps construct coordinates that parameterize the dataset and the diffusion distance provides a local preserving metric for this data. A non-linear dimensionality reduction, which reveals global geometric information, is constructed by local overlapping structures. We use the diffusion maps in order to reduce the dimensionality of the data, and then we cluster the data in the embedding. Let $X=\{x_1,\dots,x_m\}$ be a set of points in $\mathbb{R}^n$. The non-negative symmetric kernel $W_\epsilon \triangleq w_\epsilon(x_i,x_j)$ is constructed on the data. It measures the pairwise similarity between the points. $w_\epsilon(x_i,x_j)=e^{-\|x_i-x_j\|^2/\epsilon}$, which uses the Euclidean distance measure, was chosen for W_ϵ. However, other distance measures such as the cosine and Mahalanobis distances can also be used. The non-negativity property of W_ϵ allows to normalize it into a Markov transition matrix P where the states of the corresponding Markov process are the data points. This enables to analyze X as a random walk. The construction of P follows the classical construction of the *normalized graph Laplacian* [39]. Formally, $P=\{p(x_i,x_j)\}_{i,j=1,\dots,m}$ is constructed as $p(x_i,x_j)=w_\epsilon(x_i,x_j)/d(x_i)$, where $d(x_i)=\int_X w_\epsilon(x_i,x_j)\,d\mu(x_j)$ is the degree of x_i. P is a Markov matrix since the sum of each row in P is 1 and $p(x_i,x_j)\geq 0$. Thus, $p(x_i,x_j)$ can be viewed as the probability to move from x_i to x_j in *one* time step. By raising this quantity to a power t (advance in time), this influence is propagated to nodes in the neighborhood of x_i and x_j and the result is the probability for this move in t time steps. We denote this probability by $p_t(x_i,x_j)$. These probabilities measure the connectivity among the points within the graph. The parameter t controls the scale of the neighborhood in addition to the scale control provided by ϵ. Construction of $\tilde{p}(x_i,x_j)=(\sqrt{d(x_i)}/\sqrt{d(x_j)})p(x_i,x_j)$, which is a symmetric and positive semi-definite kernel, leads to the following eigen-decomposition $\tilde{p}(x_i,x_j)=\sum_{k\geq 0}^{m}\lambda_k v_k(x_i)v_k(x_j)$. A similar eigen-decomposition is obtained from $\tilde{p}_t(x_i,x_j)=\sum_{k\geq 0}^{m}\lambda_k^t v_k(x_i)v_k(x_j)$ after advancing t times on the graph. Here $\tilde{p}_t$ is the transition probability from x_i to x_j in t time steps. A fast decay of $\{\lambda_k\}$ is achieved by an appropriate choice of ϵ. Thus, only a few terms are required in the sum above to achieve a sufficient good accuracy. The family of diffusion maps Φ_t is given by $\Phi_t(x)=(\lambda_0^t v_0(x),\lambda_1^t v_1(x),\dots)^T$. Φ_t is the low-dimensional embedding of the data into an Euclidean space and it provides coordinates for the set X. Hence, we use Φ_t in order to cluster the data in the embedding space.

As described above, a common choice for W_ϵ is $w_\epsilon(x_i,x_j)=e^{-\|x_i-x_j\|^2/\epsilon}$. Since an appropriate choice of ϵ is critical in the construction of the Gaussian kernel, we describe in Section 4.1 a method that constructs an adaptive Gaussian kernel with an appropriate choice of ϵ. Moreover, the Euclidean distance metric, which is used in the construction of the Gaussian kernel, is applicable for numerical data but not for categorical data. In Section 4.2, we describe how to adapt the Gaussian kernel to our transformed (categorical) data.

4.1. Construction of an adaptive Gaussian kernel

In this section, we propose a method for the construction of an adaptive Gaussian kernel. Let $X=\{x_1,\dots,x_m\}$ be a set of points in $\mathbb{R}^n$. Let $w_\epsilon(x_i,x_j)=e^{-\|x_i-x_j\|^2/\epsilon}$ be the imposed Gaussian kernel. For each point $x_i \in X$, this Gaussian kernel pushes away from x_i all the points that are already far away from x_i. On the other hand, it pulls toward x_i all the points that are already close to x_i. This push-pull process is controlled by ϵ. Since ϵ is fixed for all the entries in W, it produces a coarse scaling control. This scale control is obviously not optimal for all the entries in the Gaussian kernel since it does not take into account the local geometry of each data point in the graph. Fig. 2 shows an example of data points in R^3. In this example, $\epsilon=0.02$ was chosen as the scale control. The orange/red points in this image are the neighbors of the green points according to the constructed Gaussian kernel (using the fixed scale control). These are the points that were pulled toward the green points.

In the left image, we see that although the green point is in a very dense area, it has only few neighbors. In this case, we are interested in pulling more points toward the green point. This is achieved by selecting a larger ϵ (that provides a wider radius) since it will include more points in the neighborhood of the green point. In the right image, we see that although the green point is an outlier, it has relatively many neighbors where some of them are in areas that are well separated from the green point. In this case, we are interested in pushing more points away from the green point. This is achieved by selecting a smaller ϵ (that provides a narrower radius) since it will include less points in the neighborhood of the green point.

Therefore, a selection of an adaptive scale control is necessary in order to construct a more accurate Gaussian kernel that will express the local geometry of each data point in the graph.

In this section, we propose a method that constructs a two-phase adaptive Gaussian kernel. The key idea in the construction of the adaptive Gaussian kernel is to determine automatically an

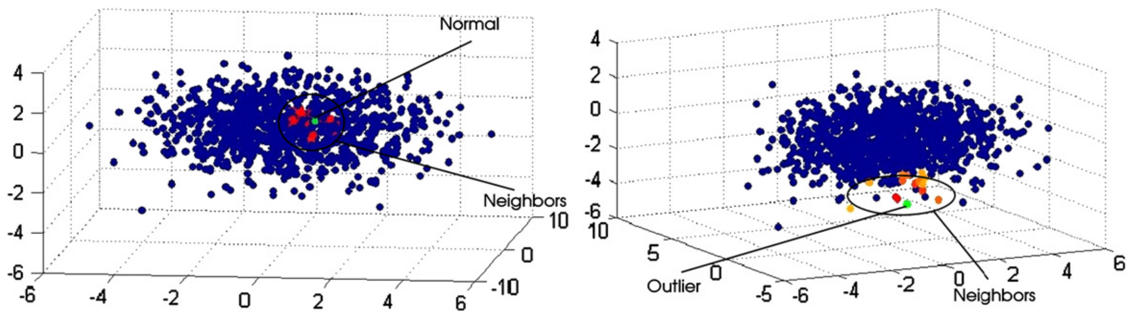

Fig. 2. Gaussian kernel with fixed ε. Left: normal point. Right: outlier point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

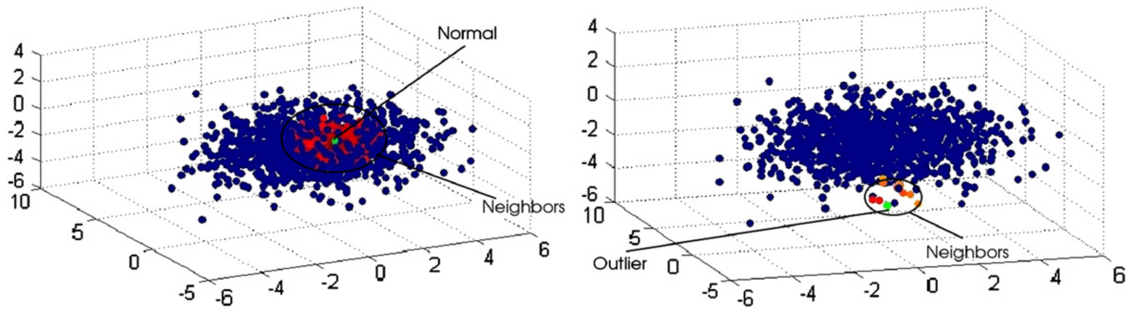

Fig. 3. Gaussian kernel with an adaptive scale control. Left: normal point. Right: outlier point. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

adaptive scale control for each point in the graph. This adaptive scale control is a weight function that has the following property: data points in dense areas will have a large weight ("bonus") and data points in sparse areas will have a small weight ("penalty"). Since dense-areas points have many close points and sparse areas do not, we define the weight function ω that represents the local geometry around each data point.

$$\omega_{\epsilon_i}(x_i) = \int_{\tilde{X}_i} e^{-\|x_i - x_j\|^2/\epsilon_i}\, d\mu(x_j), \tag{3}$$

where $\tilde{X}_i$ is a big cloud of points around x_i, μ is the distribution of the points in $\tilde{X}_i$, $x_j \in \tilde{X}_i$ and ϵ_i is an initial scale control that is adapted to $\tilde{X}_i$. In our experiments, we defined around each point x_i a cloud $\tilde{X}_i$ that included the $m/3$ nearest neighbors of x_i. Hence, each cloud of points contained third of the data points.

Typically, two alternatives are considered in order to determine the scale controls ϵ_i [40,41]. In the first alternative, the same scale control is defined for all ϵ_i (see for example, [42–46]). In these methods, a Gaussian radial basis function network with a constant width at each node is used to approximate a function. Although the width in each node can have a different value, the same width for every node is sufficient for universal approximation [43]. Therefore, the widths are fixed to a single global value to provide a simpler strategy. In practice, the width has effects on the numerical properties of the learning algorithms but not on the general approximation ability of the radial basis function network. However, these methods depend on uniform distribution of the data and hence inadequate in practice, where the data distribution is non-uniform.

In the second alternative, local estimation of the scale controls is performed. A common way to estimate the scale controls is to take into consideration the distribution variances of the data points in different regions (see for example, [47–49]). The width of the Gaussian kernel in a local region is determined by the mean value or by the standard deviation in each region [47]. In [48], the width is set locally to the Euclidean distance between the local center and its nearest neighbor. These methods offer a greater adaptability to the data than a fixed scale control.

Therefore, in order to determine the scale control ϵ_i in Eq. (3), we calculate the variance of the squares of the distances between the point x_i to all the points x_j that belong to the cloud of points around it:

$$\epsilon_i = \sum_{x_j \in \tilde{X}_i} \frac{\left\| \|x_i - x_j\|^2 - \overline{\tilde{X}}_i \right\|^2}{|\tilde{X}_i|}, \tag{4}$$

where $\overline{\tilde{X}}_i = \sum_{x_j \in \tilde{X}_i} \|x_i - x_j\|^2 / |\tilde{X}_i|$.

Assume $\omega_\epsilon(x_i)$, $i = 1, \ldots, m$ defines an adaptive weight for each data point. Since we construct an affinity matrix between pairs of data points, we need to define a pairwise weight function. We determine this way not only an adaptive scale for each point in X, but we also determine an adaptive scale for each pair of points. We define the pairwise weight function Ω_ϵ to be

$$\Omega_\epsilon(x_i, x_j) = \sqrt{\int_{\tilde{X}_i} e^{-\|x_i - x_k\|^2/\epsilon_i}\, d\mu(x_k) \int_{\tilde{X}_j} e^{-\|x_j - x_k\|^2/\epsilon_j}\, d\mu(x_k)} = \sqrt{\omega_{\epsilon_i}(x_i)\omega_{\epsilon_j}(x_j)}. \tag{5}$$

Ω_ϵ satisfies the "bonus" and "penalty" properties and it takes into consideration the distribution of the different regions in the data. Moreover, Ω_ϵ is symmetric and non-negative. Now, we construct the adaptive Gaussian kernel W as follows:

$$w_\epsilon(x_i, x_j) = e^{-\|x_i - x_j\|^2/\sqrt{\omega_{\epsilon_i}(x_i)\omega_{\epsilon_j}(x_j)}} = e^{-\|x_i - x_j\|^2/\Omega_\epsilon(x_i, x_j)}, \quad i,j = 1, \ldots, m. \tag{6}$$

Since both the Euclidean distance metric and Ω_ϵ are symmetric and non-negative, the constructed kernel W is symmetric and non-negative as well. This adaptive scale control provides better and compact description of the local geometric properties of the pairwise distances matrix for X.

Fig. 3 shows an example of the same data points in R^3. In this example, we constructed the adaptive Gaussian kernel that was

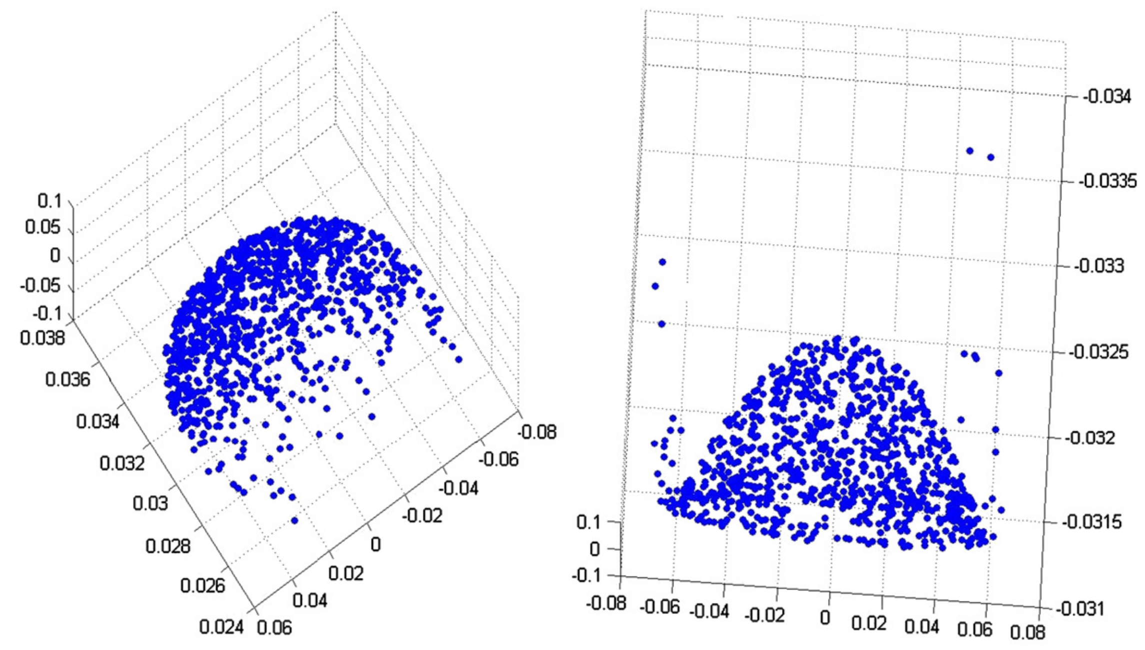

Fig. 4. Left: diffusion maps with fixed scale control. Right: adaptive scale control ("bonus" and "penalty" are used).

described above. In the left image, we see that the green point has more neighbors than when we used the fixed scale control (Fig. 2). The reason is that this green point is in a very dense area and therefore it is close to many points. Hence, the adaptive scale control is much bigger than the initial scale control (a "bonus" was awarded) and as a result more points were pulled toward the green point. In the right image, we see that the green point has fewer neighbors than when we used the fixed scale control. The reason is that this green point is an outlier and therefore it is far from most of the points. Therefore, the adaptive scale control is much smaller than the initial scale control (a "penalty" was assigned) and as a result more points were pushed away from the green point. These selections of the adaptive scale controls provide a more accurate Gaussian kernel.

Fig. 4 shows the diffusion maps coordinates of the same data points that use a Gaussian kernel with a fixed scale control and the proposed kernel with the adaptive scale control. As we can see, when we use a fixed scale control (the left image) we get a jelly-fish shape where the separation between normal (the body) and outliers (the tails) points is unclear. When we use the adaptive scale control (the right image), we get a bell shape where the separation between the normal points (the bell) and the outliers is very clear. This structure represents more accurately the geometry of the original points.

Fig. 5 shows the performance of a Gaussian kernel with a fixed scale control and the proposed kernel with an adaptive scale control. For each set of data points in R^2 (a single row in Fig. 5), we applied the diffusion maps with different fixed scale controls and with an adaptive scale control. We used the first three diffusion coordinates in order to cluster the data points. The left column of images in Fig. 5 shows the clustering results with an adaptive scale control. The remaining columns show the results of clustering with different fixed scale controls. We notice that the clustering performance with an adaptive scale control is very accurate. All the datasets were clustered correctly according to their natural clusters. In order to cluster correctly using a fixed scale control, it is necessary to choose manually an appropriate scale control for each dataset. In addition, a scale control, which is appropriate for one dataset (for example, 0.01 for the fourth dataset), is inappropriate for other datasets (for example, the second dataset requires a 100 times bigger scale control and the third dataset requires a 10 times bigger scale control). Moreover, for some datasets, all the fixed scale controls were inappropriate (for example, the fifth and sixth datasets).

Therefore, the right selection of a scale control is crucial in order to catch correctly and accurately the geometry of the data. The proposed kernel with an adaptive scale control does it correctly.

4.2. Construction of a categorical Gaussian kernel

In this section, we describe how to construct a Gaussian kernel for categorical data. The weight function $W_\epsilon \triangleq w_\epsilon(x_i,x_j)$ measures the pairwise similarity between points. W_ϵ uses common distance measures for numerical data such as Euclidean, cosine and Mahalanobis distances. However, these distance measures are not applicable for categorical data. In order to construct a weight function for categorical data, we use the weighted Hamming distance to measure the distance between categorical data points. Let $X=\{x_1,\dots,x_m\}$ be a set of n-dimensional categorical data points and let $X^l=\{x_1^l,\dots,x_m^l\}$ where x_i^l is the lth component in the n-dimensional x_i. Denote by k^l the number of categories of each variable X^l, $l=1,\dots,n$, which denotes one feature. The weighted Hamming distance Δ between two points $x_i,x_j \in X$ is

$$\Delta(x_i,x_j)=\sum_{l=1}^{n}\frac{\delta(x_i^l,x_j^l)}{k^l},\quad x_i^l,x_j^l \in X^l, \tag{7}$$

where

$$\delta(x_i^l,x_j^l)=\begin{cases}0 & \text{if } x_i^l=x_j^l,\\ 1 & \text{otherwise}.\end{cases} \tag{8}$$

Therefore, the Gaussian kernel of the weighted Hamming distance of categorical data is

$$w_\epsilon(x_i,x_j)=e^{-\Delta(x_i,x_j)/\epsilon}. \tag{9}$$

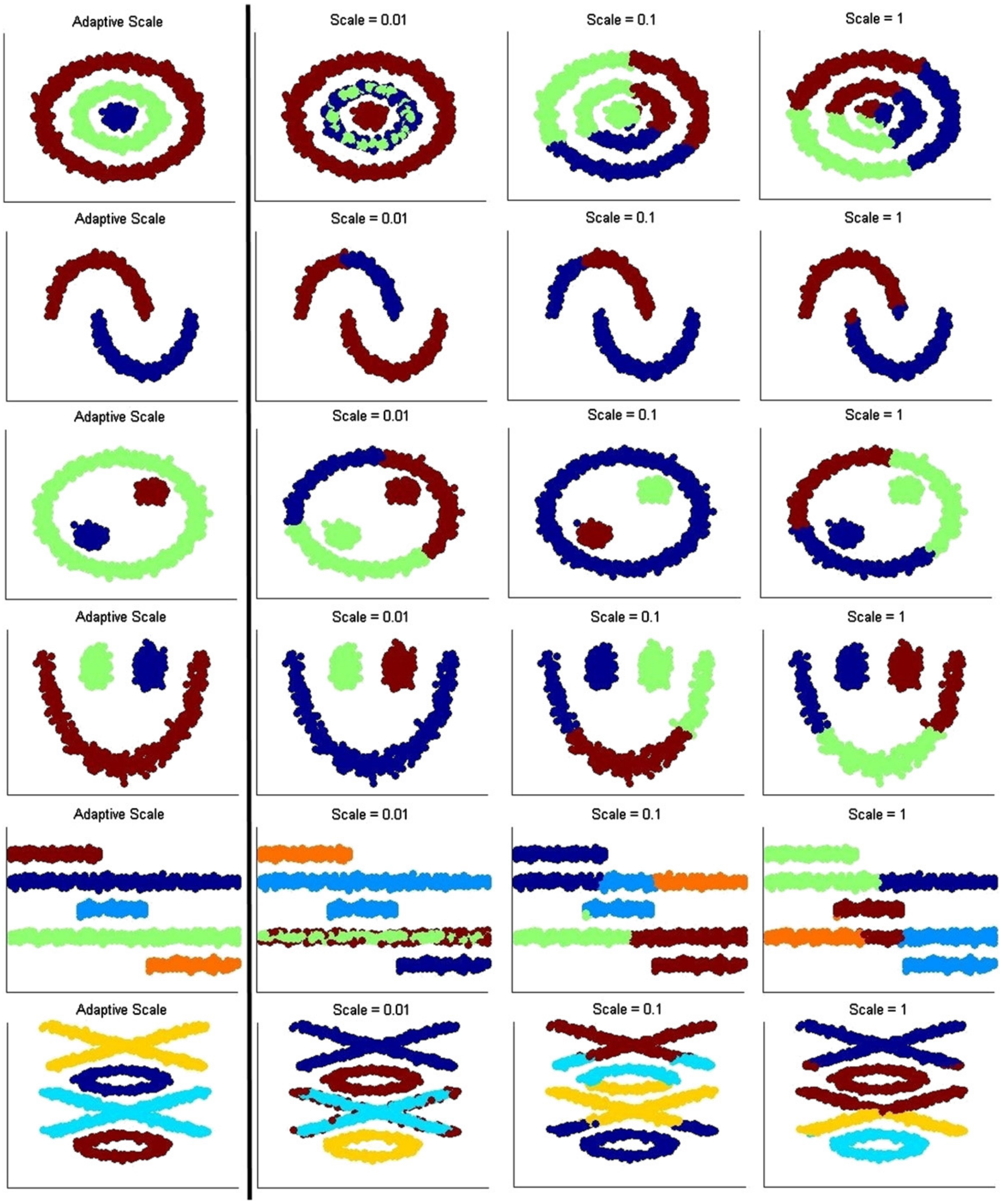

Fig. 5. Left column: adaptive scale control (“bonus” and “penalty” are used). Three right columns: diffusion maps with different fixed scale controls.

5. SpectralCAT: categorical spectral clustering of numerical and nominal data

In Section 5.1, we provide a detailed description of the SpectralCAT algorithm for clustering numerical data or nominal data or mixed data. Section 5.2 describes SpectralCAT++, which extends the SpectralCAT algorithm.

5.1. SpectralCAT clustering algorithm

The flow of the SpectralCAT algorithm is presented in Fig. 6.

Let $X=\{x_1,\ldots,x_m\}$ be a set of n-dimensional points in $\mathbb{R}^n$ where $x_i=\{x_i^1,\ldots,x_i^n\}$, $i=1,\ldots,m$. X contains nominal values or numerical values or mix of nominal and numerical values.

Data transformation by automatic categorization of X: The high-dimensional data is transformed to common categorical scales. This is done automatically by discovering the optimal transformation for each feature and attribute in the dataset. Denote column $l, 1 \le l \le n$, in X by $X^l \overset{\Delta}{=} \{x_i^l : 1 \le i \le m\}$. Each column vector l is transformed into categorical vector $\hat{X}^l$ by the following data.

If X^l contains categorical data: The set of unique nominal values in X^l denoted by $D^l \overset{\Delta}{=} \{d_1^l,\ldots,d_{k^l}^l\}$, $k^l \le m$ is the number of unique nominal values in X^l. Therefore, each point in X^l contains a nominal value from the set D: $x_i^l \in D$, $i=1,\ldots,m$. In order to transform each $x_i^l \in X^l$ to $\hat{x}_i^l$, x_i^l is replaced with the corresponding category number of its nominal value $\hat{x}_i^l=\{j : x_i^l = d_j, 1 \le j \le k^l\}$.

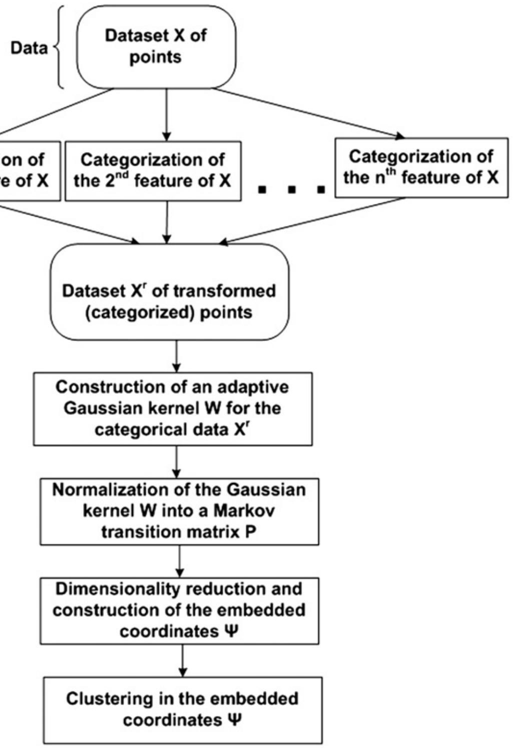
Data
Dataset X of points
Categorization of the 1st feature of X
Categorization of the 2nd feature of X
Categorization of the nth feature of X
Dataset X' of transformed (categorized) points
Construction of an adaptive Gaussian kernel W for the categorical data X'
Normalization of the Gaussian kernel W into a Markov transition matrix P
Dimensionality reduction and construction of the embedded coordinates Ψ
Clustering in the embedded coordinates Ψ

Fig. 6. The flow of the SpectralCAT algorithm.

Otherwise, X^l contains numerical data: Assume f_k performs a numerical clustering function that associates each $x_i^l \in X^l$, $i=1,\ldots,m$, to one of the clusters $c_1,\ldots,c_k$. We apply the clustering method f_k to X^l while calculating the corresponding validity indices $S_{k,m}^l$ (Eq. (2)) on X^l, $k=2,3,\ldots$. k_{best}^l is defined as the smallest k for which $S_{k,m}^l$ has a local maximum. Once the local maxima is found and k_{best}^l is discovered, the process is terminated and the clustering method $f_{k_{best}^l}$ is applied to X^l. It associates each $x_i^l \in X^l$, $i=1,\ldots,m$, to one of the clusters $c_1,\ldots,c_{k_{best}^l}$. In order to transform each $x_i^l \in X^l$ to $\hat{x}_i^l$, x_i^l is replaced with the corresponding category number by its nominal value to become $\hat{x}_i^l = \{j : f_{k_{best}^l}(x_i^l) = c_j, 1 \le j \le k_{best}^l\}$. The optimal number of clusters in X^l is denoted by k^l, where the subscript "best" is removed. k-means as f_k was used in our experiments.

This process is repeated for each l, $l=1,\ldots,n$. In the end, the original dataset X is replaced by the transformed matrix $\hat{X}$. $(k^1,\ldots,k^n)$ is the corresponding weight vector for $\hat{X}$.

Spectral clustering of the categorized data $\hat{X}$: The behavior of the transformed dataset is analyzed by projecting it onto a low-dimensional space. This step clusters the data.

Construction of an adaptive Gaussian kernel W_ϵ for categorical data: We combine the construction of a categorical Gaussian kernel (Section 4.2) with the construction of an adaptive Gaussian kernel (Section 4.1). Therefore, for any pair of points $\hat{x}_i,\hat{x}_j \in \hat{X}$, the pairwise weight function Ω_ϵ (Eq. (5)) is defined according to the Gaussian kernel of the categorical data (Eq. (9)) as

$$\Omega_\epsilon(x_i,x_j) = \sqrt{\int_{\hat{X}} e^{-\Delta(\hat{x}_i,\hat{x}_k)/\epsilon_i}\, d\mu(\hat{x}_k) \int_{\hat{X}} e^{-\Delta(\hat{x}_j,\hat{x}_k)/\epsilon_j}\, d\mu(\hat{x}_k)},$$

where the weighted distance Δ between two points was defined in Eq. (7) according to the weight vector $(k^1,\ldots,k^n)$ and ϵ_i,ϵ_j are defined according to Eq. (4). Then, the adaptive Gaussian kernel for categorical data W_ϵ is defined according to Eqs. (6) and (7) as $w_\epsilon(\hat{x}_i,\hat{x}_j) = e^{-\Delta(\hat{x}_i,\hat{x}_j)/\Omega_\epsilon(x_i,x_j)}, i,j = 1,\ldots,m$. Since both the distance metric Δ and the pairwise weight function Ω_ϵ are symmetric and non-negative, the constructed kernel W_ϵ is symmetric and non-negative as well.

Normalizing the Gaussian kernel W_ϵ into a Markov transition matrix P: The non-negativity property of W_ϵ allows us to normalize it into a Markov transition matrix P where the states of the corresponding Markov process are the data points. This enables to analyze $\hat{X}$ as a random walk. W_ϵ is normalized into a Markov matrix P:

$$p(\hat{x}_i,\hat{x}_j) = \frac{w_\epsilon(\hat{x}_i,\hat{x}_j)}{\sqrt{\sum_{q=1}^m w_\epsilon(\hat{x}_i,\hat{x}_q)}\sqrt{\sum_{q=1}^m w_\epsilon(\hat{x}_j,\hat{x}_q)}}.$$

P is a Markov matrix since the sum of each row in P is 1 and $p(\hat{x}_i,\hat{x}_j) \ge 0$. Thus, $p(\hat{x}_i,\hat{x}_j)$ can be viewed as the probability to move from $\hat{x}_i$ to $\hat{x}_j$.

Construction of the diffusion maps: Since P is a symmetric positive semi-definite kernel, the following eigen-decomposition exists: $p(\hat{x}_i,\hat{x}_j) = \sum_{w \ge 1}^m \lambda_w v_w(\hat{x}_i) v_w(\hat{x}_j)$, where λ_w are the eigenvalues and v_w are the eigenvectors. Since the spectrum decays, only few terms are required to achieve a sufficient good accuracy. Let η be the number of the retained terms. Hence, the diffusion maps Φ are given by $\Phi(\hat{x}_i) = (\lambda_0 v_0(\hat{x}_i), \lambda_1 v_1(\hat{x}_i), \ldots, \lambda_\eta v_\eta(\hat{x}_i))^T$. Φ is the low-dimensional embedding of the data into an Euclidean space and it provides coordinates for the set $\hat{X}$.

Clustering using the embedding matrix Φ: Φ is used in order to cluster the data in the embedding. Since Φ provides coordinates on the set $\hat{X}$, we can use any numerical clustering function f_k in order to associate each $\hat{x}_i \in \hat{X}$, $i=1,\ldots,m$, to one of the clusters $c_j \in C, j=1,\ldots,k$, where k is the number of clusters in C. k-means as f_k was used in our experiments.

In Appendix C, we present the pseudocode for the SpectralCAT algorithm.

5.2. SpectralCAT++: an extension to the SpectralCAT algorithm

This section describes the SpectralCAT++ that extends the data transformation step in the SpectralCAT clustering algorithm. As described in Section 5.1, the high-dimensional dataset is transformed into categorical scales. As a result, the transformed data contains the same number of features, but each feature is categorical. The key idea in SpectralCAT++ is the increase in the number of features in the transformed data.

We recall from Section 3.1 that the process of categorization is applicable to any number of dimensions and the definition of a single dimension in X was generalized to be multidimensional (features). This enables to transform several features into a single categorical feature. In addition to the transformation of each feature in the data into its optimal categorical scales (as done by SpectralCAT), SpectralCAT++ transforms different combinations of parameters, that form multidimensional features, into categorical scales. This is done for n-tuple features, $n=1,2,\ldots$, where each combination of features is transformed into a single categorical feature as described in Section 3.1.

Therefore, for n-dimensional dataset, each of the n features is transformed into categorical scales. Then, each of the $\binom{n}{2}$ combinations of two features is transformed into categorical scales. This adds $\binom{n}{2}$ new features to the transformed data. We continue by adding $\binom{n}{3}$ combinations of three features, and so on. As a result, the dimensionality of the transformed data increases from n (the original dimensionality of the data) to $\binom{n}{1}+\binom{n}{2}+\binom{n}{3}+\cdots+\binom{n}{q}$, where q is the maximal number of desired combinations of features. Since the number of combinations for high-dimensional datasets is big, this strategy is impractical. Therefore, the number of combinations is reduced by selecting only several random combinations of features from the data. Let $0 \leq \sigma \leq 1$ be the accuracy factor. Then, we randomly select $e^{\sigma^{\log\binom{n}{i}}} = \binom{n}{i}^{\sigma}$ combinations of parameters where $i=2,\ldots,q$ and $q \leq n$. σ controls the number of selected combinations and thus controls the number of added computations. Hence, the dimensionality of the transformed data increases from n to $n+\sum_{i=2}^{q}\binom{n}{i}^{\sigma}$. The assembly of these single categorical features generates the final categorical dataset, which is the input to the spectral clustering step. In Section 6, we show that even for small values of σ, where the overhead of the added computation is relatively small, the accuracy of the clustering increases.

6. Experimental evaluation

We used in our experiments 16 public datasets from UCI repository [36]. These datasets belong to wide variety of domains and problems in data mining and machine learning. Table 1 shows the properties of each dataset. The first column in the table presents the names of the datasets. The second column presents the description of the dataset. The third column presents the type of the dataset: some datasets contain numerical data, some contain nominal data and some contain mix of numerical and nominal data. The fourth column presents the number of

Table 1
The properties of the datasets that were used to analyze the performance of the algorithms.

Dataset name	Description	Type	Num. of dimensions	Num. of classes	Num. of samples
Iris	Sizes of Iris plants	Numerical	4	3	150
Wine	Chemical analysis of wines	Numerical	13	3	178
Glass	Elements and characteristics of glasses	Numerical	9	7	214
Segmentation	Image segmentation data	Numerical	19	7	2310
Ionosphere	Radar data	Numerical	34	2	351
Heart SPECTF	Diagnosis of cardiac Single Proton Emission Computed Tomography (SPECT) images	Numerical	44	2	80
Ecoli	Protein localization sites	Numerical	7	8	336
Yeast	Protein localization sites	Numerical	8	10	1484
Sat	Multi-spectral values of pixels in a satellite image	Numerical	36	6	4435
Pageblock	Blocks of the page layout of documents	Numerical	10	5	5473
Soybean	Soybean characteristics	Nominal and numeric	33	19	310
Dermatology	Clinical and histopathological characteristics of patients	Nominal and numeric	33	6	366
Adult	Data from the census bureau database	Nominal and numeric	14	2	30,612
Heart Cleveland	Diagnosis from Cleveland clinic foundation	Nominal and numeric	13	5	303
Zoo	Characteristics of animals	Nominal and numeric	17	7	101
Vowel	LPC derived log area ratio coefficients	Nominal	12	11	528

Table 2
Descriptions of the compared algorithms where num = numerical and nom = nominal.

Algorithm name	Description	Ref.	Data type
SpectralCAT	Our proposed method in this paper.	Section 5	Num, nom, mix
Random	Each point is randomly associated to one of the k clusters—this method is used as a reference point, in order to estimate the performance of the clustering algorithms.	–	–
k-modesCAT	First, we transform the data by categorization (the first step in SpectralCAT). Then, we apply k-modes, which is a categorical clustering algorithm, on the transformed data. This method is used in order to understand the effect of the transformation step without the spectral clustering step (the second step in SpectralCAT).	Section 3, [3]	Num, nom, mix
k-meansCAT	First, we transform the data by categorization (the first step in SpectralCAT). Then, we apply the popular clustering algorithm k-means on the transformed data. This method is used in order to understand the effect of the transformation step without the spectral clustering step (the second step in SpectralCAT)	Section 3, [2]	Num, nom, mix
Diffusion maps	We use the "vanilla" diffusion maps and then we cluster the embedded data using k-means. This method is used in order to understand the effect of the spectral clustering step without the transformation step, the adaptive Gaussian kernel and the Hamming distance.	[37,2]	Num
k-means	One of the most used clustering algorithm that was designed to cluster numerical data.	[2]	Num
k-means++	An algorithm for choosing the initial values for k-means clustering. It initializes the cluster centers before proceeding with the standard k-means optimization iterations (see Section 2).	[6]	Num
Kernel k-means	Kernel k-means maps the data to a higher dimension feature space using a non-linear function. Then, it partitions the points by linear separations in the new space (see Section 2).	[18]	Num
PCA k-means	First, we used PCA in order to reduce the dimensionality of the data by projecting the data onto the first principal components. Then, we cluster the projected data using k-means.	[50,2]	Num
Birch	An agglomerative hierarchical algorithm that is used for clustering large numerical datasets (see Section 2).	[9]	Num
k-modes	Emerged from the k-means algorithm and it was designed to cluster categorical datasets (see Section 2).	[3]	Nom
k-prototypes	The k-prototypes algorithm integrates the k-means and k-modes algorithms to enable clustering of mixed data (see Section 2).	[1]	Mix
LDA-Km	The LDA-Km algorithm combines linear discriminant analysis and k-means clustering to select the most discriminative subspace (see Section 2).	[7]	Num
Gower k-means	The k-means algorithm applied to the Gower similarity index to enable clustering of mixed data (see Section 2).	[2,30]	Num, nom, mix
Ng, Jordan and Weisss	The k-means algorithm applied to the first k eigenvectors of the normalized Laplacian (see Section 2)	[2,25].	Num
Kannan, Vempala and Vetta	A spectral algorithm that projects all the points onto a subspace defined by the top k right singular vectors (see Section 2).	[23]	Num
Shi and Malik	A method for using the first non-trivial eigenvector to approximate the optimal normalized cut of a graph (see Section 2).	[2,24]	Num

Table 3
Overall accuracy results: comparison between SpectralCAT and different algorithms. Bolded italic numbers represent the best achieved accuracy. The accuracy is defined according to the Purity index [51].

Algorithm name	Iris	Wine	Glass	Segmentation	Ionosphere	Heart SPECTF	Ecoli	Yeast	Sat	Pageblock	Soybean	Dermatology	Adult	Heart Cleveland	Zoo	Vowel
Spectral-CAT	0.97	***0.97***	***0.7***	***0.65***	0.89	***0.79***	0.74	0.42	***0.74***	0.9	***0.78***	***0.87***	***0.81***	0.82	***0.93***	0.38
Random	0.39	0.41	0.41	0.19	***0.89***	0.56	0.45	0.35	0.26	***0.9***	0.24	0.32	0.74	0.55	0.43	0.17
k-modes CAT	0.64	0.48	0.47	0.24	0.89	0.66	0.47	0.35	0.54	0.9	0.67	0.83	0.8	***0.83***	0.91	0.13
k-means CAT	0.89	0.7	0.59	0.6	0.89	0.68	0.86	0.54	0.73	0.9	0.73	0.43	0.75	0.56	0.82	0.31
Diffusion maps	0.9	0.71	0.65	0.62	0.89	0.75	***0.87***	0.53	0.73	0.9	0.68	0.41	0.75	0.67	0.87	0.34
k-means	0.89	0.7	0.59	0.59	0.89	0.68	0.86	0.53	0.73	0.9	0.74	0.42	0.75	0.56	0.86	0.29
k-means++	0.89	0.7	0.6	0.59	0.89	0.68	0.85	0.53	0.73	0.9	0.73	0.37	0.75	0.59	0.86	0.31
Kernel *k*-means	0.96	0.5	0.65	0.24	0.89	0.71	***0.87***	***0.56***	0.28	0.89	0.68	0.41	0.74	0.6	0.89	0.31
PCA *k*-means	0.89	0.7	0.65	0.62	0.89	0.68	0.79	0.49	0.73	0.9	0.6	0.43	0.75	0.56	0.86	0.29
Birch	0.89	0.7	0.58	0.63	0.89	0.66	0.86	0.53	0.73	0.9	0.76	0.43	0.75	0.55	0.83	0.31
k-modes	0.59	0.46	0.47	0.24	0.89	0.64	0.47	0.38	0.48	0.9	0.63	0.8	***0.81***	***0.83***	0.9	0.15
k-prototypes	–	–	–	–	–	–	–	–	–	–	–	0.77	0.75	0.67	0.88	***0.42***
LDA-Km	***0.98***	0.83	0.51	–	–	–	–	–	–	–	0.76	–	–	–	0.84	–
Gower *k*-means	0.88	0.96	0.56	0.64	0.89	0.56	0.82	0.53	0.73	0.9	0.59	0.85	0.79	0.58	0.88	0.17
Ng, Jordan and Weisss	0.81	0.8	0.51	0.51	0.89	0.61	0.65	0.39	0.7	0.9	0.43	0.41	0.77	0.55	0.75	0.12
Kannan, Vempala and Vetta	0.81	0.81	0.46	0.35	0.89	0.63	0.58	0.36	0.5	0.9	0.37	0.4	0.77	0.54	0.71	0.09
Shi and Malik	0.71	0.62	0.46	0.51	0.89	0.6	0.6	0.37	0.67	0.9	0.38	0.41	0.76	0.55	0.77	0.13

dimensions (features) in each dataset: the lowest number of dimensions is 4 and the highest is 44. The fifth column presents the number of classes in each dataset: the minimal number of classes is 2 and the maximal number is 19. The sixth column presents the number of samples in each dataset: the minimal number of samples is 80 and the maximal number is 30,612.

In order to evaluate the performance of SpectralCAT, we compare its clustering results to several known clustering algorithms. Table 2 describes each of these clustering algorithms. The last column in this table presents the type of the data that each algorithm is designed to cluster: some algorithms were designed to cluster numerical data, some nominal data and some mix of numerical and nominal data. In order to understand the effect of our proposed categorization step (the first step in the algorithm) without the dimensionality reduction step (the second step in the algorithm), we compare our method to two clustering methods that we call *k*-modesCAT and *k*-meansCAT. First, we transform the data by categorization (the first step in SpectralCAT). Then, we apply *k*-modes, which is a categorical clustering algorithm (*k*-modesCAT algorithm in Table 1), or *k*-means, which is a numerical clustering algorithm (*k*-meansCAT algorithm in Table 1), on the transformed data. On the other hand, in order to understand the effect of the spectral clustering step without the categorization step, the adaptive Gaussian kernel and the Hamming distance, we compare our method to vanilla Diffusion Maps and then we cluster the embedded data using *k*-means (Diffusion Maps algorithm in Table 1).

For each clustering algorithm, we measured the overall accuracy as follows: let $X=\{x_1,\ldots,x_m\}$ be a set of n-dimensional points in $\mathbb{R}^n$. Let $L=\{l_1,\ldots,l_q\}$ be a set of different classes. For each n-dimensional point $x_i\in X$, the corresponding label y_i, $y_i=l_j, 1\le j\le q$ is assigned. Therefore, $Y=\{y_1,\ldots,y_m\}$ is a set of labels for the dataset X. Since the compared algorithms use unsupervised clustering methods, Y is used only for measuring the quality of the clustering algorithms and not for the clustering process itself. Let f be a clustering algorithm to be evaluated. Let k be the number of clusters that f generates. Then, f_k is a clustering algorithm that associates each $x_i\in X$, $i=1,\ldots,m$, to one of the clusters $c_r\in C$, $r=1,\ldots,k$, where k is the number of clusters in C. For each $c_r\in C$, c_r is labeled according to the majority of the

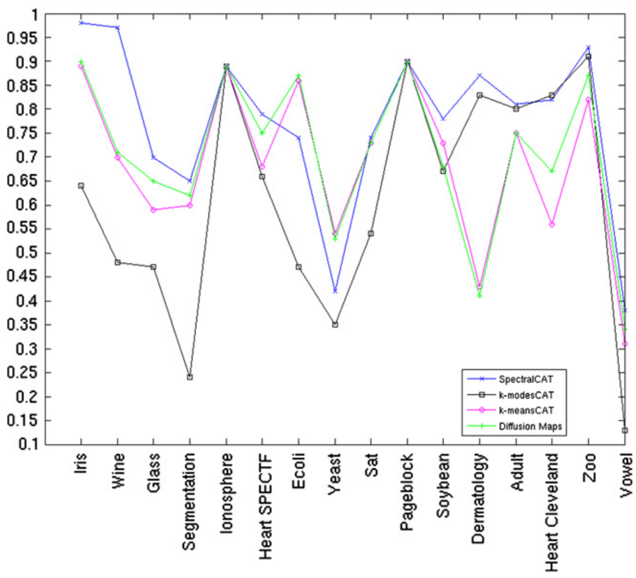

Fig. 7. Comparison between the accuracy obtained by SpectralCAT, *k*-modesCAT, *k*-meansCAT and diffusion maps. The accuracy is defined according to the Purity index [51].

Table 4
Overall accuracy results from the application of SpectralCAT++ with different parameters. Bolded italic numbers represent the best achieved accuracy. The accuracy is defined according to the Purity index [51].

Algorithm	Iris	Wine	Glass	Segmentation	Ionosphere	Heart SPECTF	Ecoli	Yeast	Sat	Pageblock	Soybean	Dermatology	Adult	Heart Cleveland	Zoo	Vowel
Spectral-CAT	0.97	0.97	0.7	0.65	0.89	0.79	0.74	0.42	***0.74***	0.9	***0.78***	0.87	0.81	0.82	0.93	0.38
Spectral-CAT++ with $q=2$ and $\sigma=1$	0.97	0.98	***0.71***	0.65	0.89	0.76	0.8	0.5	0.73	0.9	***0.78***	0.89	0.81	***0.84***	0.93	0.41
Spectral-CAT++ with $q=3$ and $\sigma=1$	***0.98***	0.97	***0.71***	0.68	0.89	0.79	0.8	0.49	0.71	0.9	0.77	0.93	0.8	0.78	0.91	0.4
Spectral-CAT++ with $q=2$ and $\sigma=0.8$	0.96	0.96	0.7	0.66	0.89	***0.8***	0.81	0.52	0.72	0.9	0.76	0.93	***0.82***	***0.84***	0.93	0.42
Spectral-CAT++ with $q=3$ and $\sigma=0.8$	0.97	0.97	0.7	0.66	0.89	***0.8***	0.8	0.53	0.72	0.9	0.77	***0.95***	***0.82***	0.83	0.94	***0.43***
Spectral-CAT++ with $q=2$ and $\sigma=0.3$	0.96	***0.99***	0.69	***0.71***	0.89	0.79	0.78	***0.54***	0.68	0.9	0.77	0.91	0.81	***0.84***	0.93	0.4
Spectral-CAT++ with $q=3$ and $\sigma=0.3$	0.96	0.97	0.7	***0.71***	0.89	0.74	***0.83***	0.53	0.7	0.9	0.75	0.93	***0.82***	0.82	***0.95***	0.39

records in the cluster. Formally, let $B_{c_r}^p = \sum_{x_i \in c_r} \delta(x_i, l_p)$, where

$$\delta(x_i,l_p) = \begin{cases} 1 & \text{if } y_i = l_p, \\ 0 & \text{otherwise,} \end{cases} \quad 1 \le p \le q, \ 1 \le r \le k.$$

Then, the label of each cluster $c_r, r = 1, \ldots, k$, is denoted by $M_{c_r} \overset{\Delta}{=} \{l_p : \max_{1 \le p \le q} B_{c_r}^p\}$.

In order to evaluate the accuracy of the clustering algorithms, we measure the number of records in each cluster whose labels are equal to the label of the majority of the records in the cluster (M_{c_r}). Therefore, the accuracy of the clustering algorithm is $\sum_{r=1}^{k} \max_{1 \le p \le q} B_{c_r}^p / m$. This accuracy index is also known as the Purity index [51].

The accuracy results for LDA-Km were taken from [7]. In our experiments, we used the same configuration for the k-means algorithm for all of the k-means based algorithms. In a single run of this algorithm in this implementation of k-means, the k-means clustering was repeated several times. In each iteration, a new set of initial cluster centroid positions (seeds) were used. This implementation of the k-means algorithm provides a solution with the lowest value for the within-cluster sums of point-to-centroid distances. Therefore, this implementation takes into account the need to have initialization with different seeds. The best solution is chosen based on a general criteria and it is not based on the target function.

Table 3 presents the accuracy results from the application of the different algorithms to the evaluation datasets. For each clustering algorithm and each dataset, we chose k as the number of classes in the dataset (according to Table 1).

Application of the random clustering to Ionosphere and Pageblock datasets achieved the same results as all the algorithms achieved (on these two datasets). SpectralCAT outperformed all of the algorithms that were applied to 9 out of the 14 remaining datasets (one other algorithm achieved on one dataset the same result as SpectralCAT).

As mentioned above, in order to understand the effect of the two steps of SpectralCAT, we compared our method to k-modesCAT, k-meansCAT and Diffusion Maps. Fig. 7 shows the comparison results between the basic SpectralCAT and these methods. The x-axis shows the different datasets and the y-axis shows the accuracy results.

In most cases, even the basic SpectralCAT gets better results than these methods and overall SpectralCAT achieved the best results. Therefore, it is important to perform the two steps and not only one of them.

Table 4 presents the accuracy results from the application of SpectralCAT++ to the evaluation datasets. As described in Section 5.2, we randomly select $\binom{n}{i}^\sigma$ combinations of parameters where $i = 2, \ldots, q$ and $0 \le \sigma \le 1$ is the accuracy factor. In this experiment, we repeated the tests with different values for q and different values for σ.

For four datasets, SpectralCAT++ did not improve the accuracy results of SpectralCAT. For 12 datasets, SpectralCAT++ improved the accuracy results of SpectralCAT algorithm and in some cases even significantly. It can be noticed that even for a small value of σ, where the overhead of the added computation is relatively small, SpectralCAT++ improves in most cases the accuracy results of SpectralCAT.

Table 5 summarizes this experiment and presents the accuracy results for different algorithms on the evaluation datasets, including the results of SpectralCAT++.

7. Conclusions

We presented two automated techniques for unsupervised data clustering that contains numerical or nominal or mix of

Table 5
Overall accuracy results where num = numerical and nom = nominal. Bolded italic numbers represent the best achieved accuracy. The accuracy is defined according to the Purity index [51].

	Iris	Wine	Glass	Segmentation	Ionosphere	Heart SPECTF	Ecoli	Yeast	Sat	Pageblock	Soybean	Dermatology	Adult	Heart Cleveland	Zoo	Vowel
Dataset information																
Dataset type	Num	Num	Num	Num	Num	Num	Num	Num	Num	Num	Nom	Num nom	Num nom	Num nom	Num nom	Num nom
Features	4	13	9	19	34	44	7	8	36	10	33	33	14	13	17	12
Classes	3	3	7	7	2	2	8	10	6	5	19	6	2	5	7	11
Samples	150	178	214	2310	351	80	336	1484	4435	5473	310	366	30612	303	101	528
Algorithm																
SpectralCAT	0.97	0.97	0.7	0.65	0.89	0.79	0.74	0.42	***0.74***	0.9	***0.78***	0.87	0.81	0.82	0.93	0.38
Spectral-CAT++ ($q=2$, $\sigma=1$)	0.97	0.98	***0.71***	0.65	0.89	0.76	0.8	0.5	0.73	0.9	***0.78***	0.89	0.81	***0.84***	0.93	0.41
Spectral-CAT++ ($q=3$, $\sigma=1$)	***0.98***	0.97	***0.71***	0.68	0.89	0.79	0.8	0.49	0.71	0.9	0.77	0.93	0.8	0.78	0.91	0.4
Spectral-CAT++ ($q=2$, $\sigma=0.8$)	0.96	0.96	0.7	0.66	0.89	***0.8***	0.81	0.52	0.72	0.9	0.76	0.93	***0.82***	***0.84***	0.93	0.42
Spectral-CAT++ ($q=3$, $\sigma=0.8$)	0.97	0.97	0.7	0.66	0.89	***0.8***	0.8	0.53	0.72	0.9	0.77	***0.95***	***0.82***	0.83	0.94	***0.43***
Spectral-CAT++ ($q=2$, $\sigma=0.3$)	0.96	***0.99***	0.69	***0.71***	0.89	0.79	0.78	0.54	0.68	0.9	0.77	0.91	0.81	***0.84***	0.93	0.4
Spectral- CAT++ ($q=2$, $\sigma=0.3$)	0.96	0.97	0.7	***0.71***	0.89	0.74	0.83	0.53	0.7	0.9	0.75	0.93	***0.82***	0.82	***0.95***	0.39
Random	0.39	0.41	0.41	0.19	***0.89***	0.56	0.45	0.35	0.26	***0.9***	0.24	0.32	0.74	0.55	0.43	0.17
k-modesCAT	0.64	0.48	0.47	0.24	0.89	0.66	0.47	0.35	0.54	0.9	0.67	0.83	0.8	0.83	0.91	0.13
k-meansCAT	0.89	0.7	0.59	0.6	0.89	0.68	0.86	0.54	0.73	0.9	0.73	0.43	0.75	0.56	0.82	0.31
Diffusion maps	0.9	0.71	0.65	0.62	0.89	0.75	***0.87***	0.53	0.73	0.9	0.68	0.41	0.75	0.67	0.87	0.34
k-means	0.89	0.7	0.59	0.59	0.89	0.68	0.86	0.53	0.73	0.9	0.74	0.42	0.75	0.56	0.86	0.29
k-means++	0.89	0.7	0.6	0.59	0.89	0.68	0.85	0.53	0.73	0.9	0.73	0.37	0.75	0.59	0.86	0.31
Kernel k-means	0.96	0.5	0.65	0.24	0.89	0.71	***0.87***	***0.56***	0.28	0.89	0.68	0.41	0.74	0.6	0.89	0.31
PCA k-means	0.89	0.7	0.65	0.62	0.89	0.68	0.79	0.49	0.73	0.9	0.6	0.43	0.75	0.56	0.86	0.29
Birch	0.89	0.7	0.58	0.63	0.89	0.66	0.86	0.53	0.73	0.9	0.76	0.43	0.75	0.55	0.83	0.31
k-modes	0.59	0.46	0.47	0.24	0.89	0.64	0.47	0.38	0.48	0.9	0.63	0.8	0.81	0.83	0.9	0.15
k-prototypes	–	–	–	–	–	–	–	–	–	–	–	0.77	0.75	0.67	0.88	0.42
LDA-Km	***0.98***	0.83	0.51	–	–	–	–	–	–	–	0.76	–	–	–	0.84	–
Gower k-means	0.88	0.96	0.56	0.64	0.89	0.56	0.82	0.53	0.73	0.9	0.59	0.85	0.79	0.58	0.88	0.17
Ng, Jordan and Weisss	0.81	0.8	0.51	0.51	0.89	0.61	0.65	0.39	0.7	0.9	0.43	0.41	0.77	0.55	0.75	0.12
Kannan, Vempala and Vetta	0.81	0.81	0.46	0.35	0.89	0.63	0.58	0.36	0.5	0.9	0.37	0.4	0.77	0.54	0.71	0.09
Shi and Malik	0.71	0.62	0.46	0.51	0.89	0.6	0.6	0.37	0.67	0.9	0.38	0.41	0.76	0.55	0.77	0.13

numerical and nominal attributes. These techniques are based on automatic transformation of high-dimensional data to common categorical scales. Then, spectral clustering of the transformed data is performed by projecting it onto a low-dimensional space. An empirical evaluation of our approach was presented. Our methods were compared to several clustering algorithms by applying them to 16 public datasets from different domains and types. Our experiments showed that our methods outperform in most case these algorithms. Although some of the compared algorithms were designed to cluster specific data types (numerical, nominal or mix), we still outperformed most of them to obtain the best results. The experiments show that SpectralCAT and SpectralCAT++ are generic and suitable to operate on different data types from various domains including high-dimensional data.

Appendix A. Justification of the Calinski–Harabasz index

In order to justify the use of this validity index, we use the fact that the sum of squares of m points is equal to the sum of the upper (or lower) triangular matrix of the pairwise distances between the m points divided by m (Appendix B, point 2). Therefore,

$$\sum_{i=1}^{m}(x_i^l-\overline{x}^l)^2=\frac{\sum_{i<j}^{m}(x_i^l-x_j^l)^2}{m}. \tag{A.1}$$

According to Eq. (A.1), we reformulate S^l to be

$$S^l=\frac{\sum_{i<j}^{m}(x_i^l-x_j^l)^2}{m}=\frac{(m-1)\overline{\delta}^l}{2}, \tag{A.2}$$

where $\overline{\delta}^l=(1/(m(m-1)/2))\sum_{i<j}^{m}(x_i^l-x_j^l)^2$ is the mean of the triangular matrix of the pairwise distances between the m points (excluding the zero diagonal). Similarly, we reformulate $S_w^l(k)$ to be

$$S_w^l(k)=\sum_{j=1}^{k}\frac{\sum_{i<r,x_i^l,x_r^l\in c_j^l}^{|c_j^l|}(x_i^l-x_r^l)^2}{|c_j^l|}=\frac{\sum_{j=1}^{k}(|c_j^l|-1)\overline{\delta_j}^l}{2}, \tag{A.3}$$

where $\overline{\delta_j}^l=(1/(|c_j^l|(|c_j^l|-1)/2))\sum_{i<r,x_i^l,x_r^l\in c_j^l}^{|c_j^l|}(x_i^l-x_r^l)^2$ is the mean of the triangular matrix of the pairwise distances between the $|c_j^l|$ points in c_j^l (excluding the zero diagonal). Since $S_b^l(k)=S^l-S_w^l(k)$ (Eq. (1)) and according to Eqs. (A.2) and (A.3), we reformulate $S_b^l(k)$ to be

$$\begin{aligned}S_b^l(k)&=\frac{(m-1)\overline{\delta}^l}{2}-\frac{\sum_{j=1}^{k}(|c_j^l|-1)\overline{\delta_j}^l}{2}\\&=\frac{(k-1)\overline{\delta}^l+(m-k)\overline{\delta}^l-\sum_{j=1}^{k}(|c_j^l|-1)\overline{\delta_j}^l}{2}\\&=\frac{(k-1)\overline{\delta}^l+\sum_{j=1}^{k}(|c_j^l|-1)(\overline{\delta}^l-\overline{\delta_j}^l)}{2}=\frac{(k-1)\overline{\delta}^l+(m-k)A_k}{2},\end{aligned} \tag{A.4}$$

where $A_k=\sum_{j=1}^{k}(|c_j^l|-1)(\overline{\delta}^l-\overline{\delta_j}^l)/(m-k)$ is a weighted mean of the differences between the mean of the squared distances (between the m points) and the mean of the squared distances between each group points.

Therefore, by Eqs. (A.3) and (A.4), Eq. (2) becomes

$$\begin{aligned}S_{k,m}^l&=\frac{(m-k)S_b^l(k)}{(k-1)S_w^l(k)}\\&=\frac{\dfrac{(k-1)\overline{\delta}^l+(m-k)A_k}{2(k-1)}}{\dfrac{\sum_{j=1}^{k}(|c_j^l|-1)\overline{\delta_j}^l}{2(m-k)}}=\frac{\overline{\delta}^l+\dfrac{(m-k)A_k}{(k-1)}}{\overline{\delta}^l-\left(\overline{\delta}^l-\dfrac{\sum_{j=1}^{k}(|c_j^l|-1)\overline{\delta_j}^l}{(m-k)}\right)}=\frac{\overline{\delta}^l+\dfrac{(m-k)A_k}{(k-1)}}{\overline{\delta}^l-A_k}.\end{aligned} \tag{A.5}$$

Several properties of the validity index were described in [4]. We present these properties and their corresponding analyzes according to Eq. (A.5). Note that our contribution for the properties is for the case where the points are uniformly distributed in space.

- If the squared distances between all pairs of points are equal then by definition $\overline{\delta}^l=\overline{\delta_1}^l=\overline{\delta_2}^l=\cdots=\overline{\delta_k}^l$. Therefore, $A_k=0$ and $S_{k,m}^l=1$.
- If for each cluster $c_j^l, j=1,\ldots,k$, $\overline{\delta_j}^l=\epsilon_j$, where $\epsilon_j=0^+$ (the variation within each cluster is almost zero), then $A_k=\overline{\delta}^l-\sum_{j=1}^{k}(|c_j^l|-1)\epsilon_j/(m-k)=\overline{\delta}^l-\epsilon$, where $\epsilon=0^+$. Therefore, $S_{k,m}^l=((m-1)\overline{\delta}^l-(m-k)\epsilon)/(k-1)\epsilon$ and since $\epsilon=0^+$ then $S_{k,m}^l$ is maximized.
- If the m points are uniformly distributed in space, then for each selection of k, $|c_1^l|=|c_2^l|=\cdots=|c_k^l|$ and $\overline{\delta_1}^l=\overline{\delta_2}^l=\cdots=\overline{\delta_k}^l=\delta_k^l$ (all the clusters have the same size and the same variation). Therefore,

$$A_k=\frac{\sum_{j=1}^{k}(|c_j^l|-1)(\overline{\delta}^l-\delta_k^l)}{(m-k)}=\frac{(m-k)(\overline{\delta}^l-\delta_k^l)}{(m-k)}=\overline{\delta}^l-\delta_k^l$$

and

$$S_{k,m}^l=\frac{\overline{\delta}^l+\dfrac{(m-k)(\overline{\delta}^l-\delta_k^l)}{k-1}}{\overline{\delta}^l-\overline{\delta}^l+\delta_k^l}=\frac{(m-1)\overline{\delta}^l-(m-k)\delta_k^l}{(k-1)\delta_k^l}.$$

We show now that $S_{k,m}^l$ is monotonic increasing, i.e. for all k, $S_{k,m}^l<S_{k+1,m}^l$.

$$S_{k,m}^l<S_{k+1,m}^l$$

$$\frac{(m-1)\overline{\delta}^l-(m-k)\delta_k^l}{(k-1)\delta_k^l}<\frac{(m-1)\overline{\delta}^l-(m-k-1)\delta_{k+1}^l}{k\delta_{k+1}^l}$$

$$\begin{aligned}&k(m-1)\overline{\delta}^l\delta_{k+1}^l-k(m-k)\delta_k^l\delta_{k+1}^l\\&\quad<(k-1)(m-1)\overline{\delta}^l\delta_k^l-k(m-k)\delta_k^l\delta_{k+1}^l+(m-1)\delta_k^l\delta_{k+1}^l\end{aligned}$$

$$k\overline{\delta}^l\delta_{k+1}^l<(k-1)\overline{\delta}^l\delta_k^l+\delta_k^l\delta_{k+1}^l$$

$$0<\delta_k^l\left(\frac{1}{\delta_{k+1}^l}-\frac{1}{k\delta_{k+1}^l}+\frac{1}{k\overline{\delta}^l}-\frac{1}{\delta_k^l}\right). \tag{A.6}$$

Since $\delta_k^l>0$ and $1/k\overline{\delta}^l>0$ then it is sufficient to show that

$$0<\frac{1}{\delta_{k+1}^l}-\frac{1}{k\delta_{k+1}^l}-\frac{1}{\delta_k^l}$$

$$\frac{1}{\delta_k^l}<\frac{k-1}{k\delta_{k+1}^l}$$

$$\frac{k}{k-1}<\frac{\delta_k^l}{\delta_{k+1}^l}. \tag{A.7}$$

For simplicity of the computation, we assume that the m points, which have uniform discrete distribution, are in the

range $[1,m]$. Therefore, the variance $\overline{\delta}^l$ of the m points is $(m^2-1)/12$ (Appendix B, point 3).
For the general case, where the m points are distributed uniformly in the range $[a,b]$ we get $x_1^l=a, x_2^l=a+\alpha, x_3^l=a+2\alpha,\ldots,x_m^l=b$. Therefore, $m=(b-a)/\alpha+1$ and

$$\overline{\delta}^l=\alpha^2\frac{\left(\frac{b-a}{\alpha}+1\right)^2-1}{12}=\frac{\alpha^2(m^2-1)}{12}. \qquad \text{(A.8)}$$

Since the m points are uniformly distributed then the number of points in each cluster is m/k. Therefore, from Eqs. (A.7) and (A.8) we have to show that

$$\frac{k}{k-1}<\frac{\delta_k^l}{\delta_{k+1}^l}$$

$$\frac{k}{k-1}<\frac{12\alpha^2\left(\left(\frac{m}{k}\right)^2-1\right)}{12\alpha^2\left(\left(\frac{m}{k+1}\right)^2-1\right)}$$

$$\frac{k}{k-1}<\frac{\frac{m^2-k^2}{k^2}}{\frac{m^2-(k+1)^2}{(k+1)^2}}$$

$$k^3(m^2-(k+1)^2)<(m^2-k^2)(k+1)^2(k-1)$$

$$k^3m^2-k^5-2k^4-k^3<m^2k^3-k^5+m^2k^2-k^4-km^2+k^3-m^2+k^2$$

$$m^2(k+1)<m^2k^2+k^4+2k^3+k^2, \qquad \text{(A.9)}$$

which is true since $k^2>(k+1)$ for $k>1$.
Therefore, for all $k>1$, $S_{k,m}^l<S_{k+1,m}^l$, $S_{k,m}^l$ is monotonic increasing and the (local) maximum is obtained when $k=m$.
Fig. A.1 shows an example of the application of this process to uniform distribution data points. It can be noticed that the validity indexes function is monotonic increasing.

- For the general case, if $S_{k-1,m}^l<S_{k,m}^l$ then the following inequality exists:

$$S_{k-1,m}^l<S_{k,m}^l$$

$$\frac{\overline{\delta}^l+\frac{(m-k+1)A_{k-1}}{(k-2)}}{\overline{\delta}^l-A_{k-1}}<\frac{\overline{\delta}^l+\frac{(m-k)A_k}{(k-1)}}{\overline{\delta}^l-A_k}$$

$$(m-1)A_{k-1}\frac{\overline{\delta}^l}{k-2}<\frac{(m-1)A_kA_{k-1}}{(k-1)(k-2)}+\frac{(m-1)A_k\overline{\delta}^l}{k-1}$$

$$\frac{A_{k-1}\overline{\delta}^l}{A_k}<\frac{A_{k-1}+(k-2)\overline{\delta}^l}{k-1}$$

$$\frac{k-1}{\frac{A_{k-1}}{\overline{\delta}^l}+k-2}<\frac{A_k}{A_{k-1}}. \qquad \text{(A.10)}$$

Therefore, the increase from $k-1$ clusters to k clusters will cause an increase from $S_{k-1,m}^l$ to $S_{k,m}^l$ if $A_k/A_{k-1}>(k-1)/A_{k-1}/\overline{\delta}^l+k-2$.
- If the m points are clustered naturally into k clusters, then the increase from $k-1$ clusters to k clusters will cause a significant increase from A_{k-1} to A_k since the within-cluster variation is getting smaller. Therefore, the ratio between the numerator and the denominator of $S_{k,m}^l$ according to Eq. (A.5) will be significantly bigger than the ratio between the numerator and the denominator of $S_{k-1,m}^l$. Hence, there is an increase from $S_{k-1,m}^l$ to $S_{k,m}^l$.

Fig. A.2 shows an example from the application of this process to 20 clusters each of uniformly distributed data points. It can be noticed that the increase from 19 clusters to 20 natural clusters causes a significant increase from S_{19} to S_{20} (marked by the arrow).

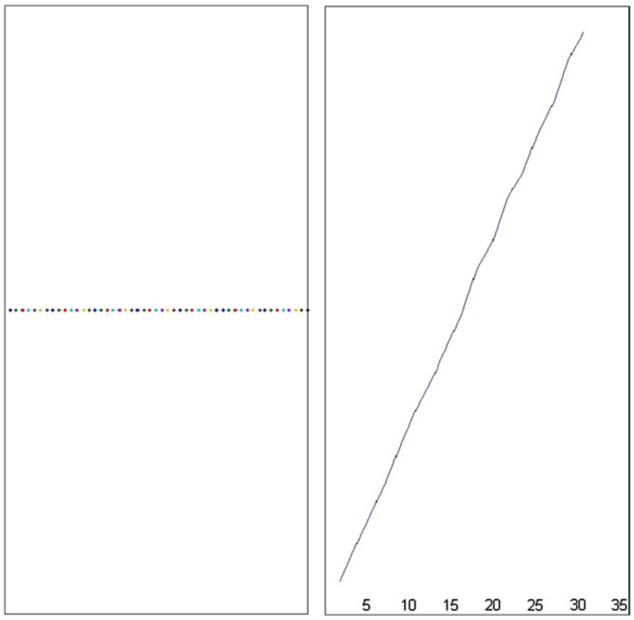

Fig. A.1. Left: uniform distributed data points. Right: the validity indexes for the uniform distributed data points.

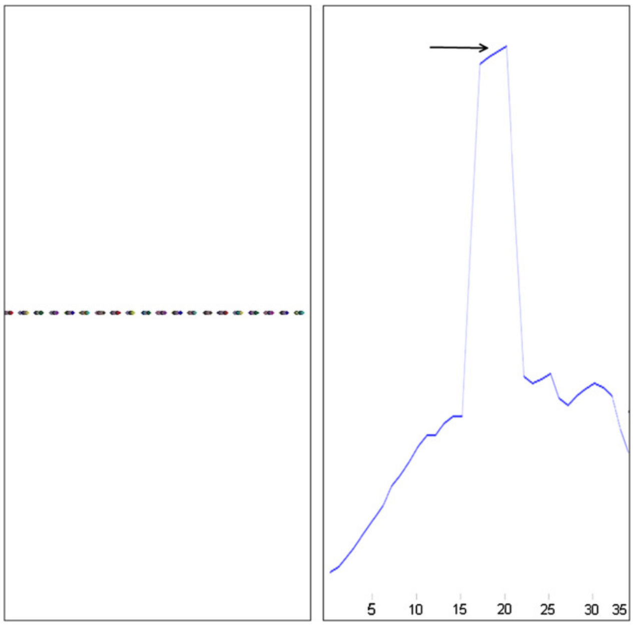

Fig. A.2. Left: 20 clusters each of uniformly distributed data points. Right: the validity indexes for these data points. The arrow marks the significant increase from S_{19} to S_{20}.

Appendix B. Statistical properties

This section presents some well-known statistical properties.

1. The total sum of squares (S^l) equals to the sum of the between-cluster sum of squares ($S_b^l(k)$) and the within-cluster sum of squares ($S_w^l(k)$):

$$\begin{aligned}
S_w^l(k)+S_b^l(k) &= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}(x^l-\mu_j^l)(x^l-\mu_j^l)^T+\sum_{j=1}^{k}|c_j^l|(\mu_j^l-\overline{x}^l)(\mu_j^l-\overline{x}^l)^T\\
&= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}(x^l-\mu_j^l)^2+\sum_{j=1}^{k}|c_j^l|(\mu_j^l-\overline{x}^l)^2\\
&= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^{l^2}-\sum_{j=1}^{k}\sum_{x^l\in c_j^l}\mu_j^{l2}+2\sum_{j=1}^{k}\sum_{x^l\in c_j^l}\mu_j^{l2}\\
&\quad -2\sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^l\mu_j^l+\sum_{j=1}^{k}|c_j^l|\mu_j^{l2}-\sum_{j=1}^{k}|c_j^l|\overline{x}^{l2}\\
&\quad +2\sum_{j=1}^{k}|c_j^l|\overline{x}^{l2}-2\sum_{j=1}^{k}|c_j^l|\mu_j^l\overline{x}^l\\
&= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^{l^2}-\sum_{j=1}^{k}|c_j^l|\mu_j^{l2}+2\sum_{j=1}^{k}|c_j^l|\mu_j^{l2}\\
&\quad -2\sum_{j=1}^{k}\mu_j^l\sum_{x^l\in c_j^l}x^l+\sum_{j=1}^{k}|c_j^l|\mu_j^{l2}-\sum_{j=1}^{k}|c_j^l|\overline{x}^{l2}\\
&\quad +2m\overline{x}^{l2}-2\overline{x}^l\sum_{j=1}^{k}|c_j^l|\mu_j^l=\sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^{l^2}-m\overline{x}^{l2}\\
&= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^{l^2}-m\overline{x}^{l2}+2m\overline{x}^{l2}-2\overline{x}^l\sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^l\\
&= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^{l^2}+m\overline{x}^{l2}-2\sum_{j=1}^{k}\sum_{x^l\in c_j^l}x^l\overline{x}^l\\
&= \sum_{j=1}^{k}\sum_{x^l\in c_j^l}(x^l-\overline{x}^l)^2=S^l.
\end{aligned}$$

2. The sum of squares of m points is equal to the sum of the upper (or lower) triangular matrix of the pairwise distances between the m points divided by m:

$$\begin{aligned}
\sum_{i=1}^{m}(x_i^l-\overline{x}^l)^2 &= (x_1^l-\overline{x}^l)^2+(x_2^l-\overline{x}^l)^2+\cdots+(x_m^l-\overline{x}^l)^2\\
&= x_1^{l\,2}+x_2^{l\,2}+\cdots+x_m^{l\,2}+m\overline{x}^{l2}-2x_1^l\overline{x}^l-2x_2^l\overline{x}^l-\cdots-2x_m^l\overline{x}^l\\
&= x_1^{l\,2}+x_2^{l\,2}+\cdots+x_m^{l\,2}+m\overline{x}^l\left(\overline{x}^l-2\frac{x_1^l+x_2^l+\cdots+x_m^l}{m}\right)\\
&= x_1^{l\,2}+x_2^{l\,2}+\cdots+x_m^{l\,2}-\frac{(x_1^l+x_2^l+\cdots+x_m^l)^2}{m}\\
&= x_1^{l\,2}+x_2^{l\,2}+\cdots+x_m^{l\,2}-\frac{x_1^{l\,2}+x_2^{l\,2}+\ldots+x_m^{l\,2}+2\sum_{i<j}^{m}(x_i^l-x_j^l)^2}{m}\\
&= \frac{(m-1)x_1^{l\,2}+(m-1)x_2^{l\,2}+\cdots+(m-1)x_m^{l\,2}-2\sum_{i<j}^{m}(x_i^l-x_j^l)^2}{m}\\
&= \frac{(x_1^{l\,2}+x_2^{l\,2}-2x_1^lx_2^l)+(x_1^{l\,2}+x_3^{l\,2}-2x_1^lx_3^l)+\cdots+(x_{m-1}^{l\,2}+x_m^{l\,2}-2x_{m-1}^lx_m^l)}{m}\\
&= \frac{\sum_{i<j}^{m}(x_i^l-x_j^l)^2}{m}.
\end{aligned}$$

3. The variance $\overline{\delta}^l$ of m uniform discrete distributed points: Since $\sum_{i=1}^{m} i^2=m(m+1)(2m+1)/6$, $\sum_{i=1}^{m} i=m(m+1)/2$ and

$$\begin{aligned}
\overline{\delta}^l &= \frac{\sum_{i=1}^{m}(x_i^l-\overline{x}^l)^2}{m}=\frac{\sum_{i=1}^{m}x_i^{l2}-2\overline{x}^l\sum_{i=1}^{m}x_i^l+m\overline{x}^{l2}}{m}\\
&= \frac{\sum_{i=1}^{m}x_i^{l2}}{m}-2\overline{x}^l\frac{\sum_{i=1}^{m}x_i^l}{m}+\overline{x}^{l2}=\frac{\sum_{i=1}^{m}x_i^{l2}}{m}-\overline{x}^{l2}
\end{aligned}$$

then

$$\begin{aligned}
\overline{\delta}^l &= \frac{m(m+1)(2m+1)}{6m}-\left(\frac{(m+1)}{2}\right)^2\\
&= \frac{2m^2+3m+1}{6}-\frac{m^2+2m+1}{4}\\
&= \frac{4m^2+6m+2-3m^2-6m-3}{12}=\frac{m^2-1}{12}.
\end{aligned}$$

Appendix C. Pseudocode of the spectralCAT algorithm

Algorithm 1. Data transformation by automatic categorization

```
Input:
  $X=\{x_1,\ldots,x_m\}$ (q-dimensional numerical points to categorize)
  $f_k$ (a clustering function that partitions X into k clusters and returns the corresponding assignments)
  max_k (maximum number of categories to examine)
Output:
  $\hat{X}=\{\hat{x}_1,\ldots,\hat{x}_m\}$ (a vector of categorical points)

Procedure Categorize
  For each $k\in\{2,\ldots,max\_k\}$
    $S(k)=CalinskiHarabasz(f_k(X),X)$ (the Calinski-Harabasz index for the result of the clustering)
  End
  $k_{best}=min_{k\in\{2,\ldots,max\_k\}}\{LocalMax(S(k))=True\}$ (the smallest k for which S(k) has a local maximum)
  $\hat{X}=f_{k_{best}}(X)$
  return $\hat{X}$
End
```

Algorithm 2. `SpectralCAT`

```
Input:
  $X=\{x_1,\ldots,x_m\}$ (n-dimensional dataset to cluster)
  $f_k$ (a clustering function that partitions X into k clusters and returns the corresponding assignments)
  k (the number of clusters to partition X)
Output:
  $IDX=\{IDX_1,\ldots,IDX_m\}$ (an assignment of each point in X to one of k clusters)

Procedure SpectralCAT
  For each $l\in\{1,\ldots,n\}$
    Set $X^l=\{x_1^l,\ldots,x_n^l\}$ (the lth column in X)
    If Numerical($X^l$) = True (the lth column contains numerical data)
      $\hat{X}^l$ = Categorize ($X^l,f_k,max\_k$) (categorize the lth column according to Alg. 1)
    Else
      $\hat{X}^l=X^l$
    End
  End
  For each $i,j\in\{1,\ldots,m\}$ (build a pairwise distance matrix)
    $W_{i,j}=Hamming(\hat{x}_i,\hat{x}_j)$
    End
    For each $i,j \in \{1,\ldots,m\}$ (build a pairwise weight function - Eqs. (5) and (6))
        $\Omega_{i,j} = GaussianWeight(W_{i,j})$
        $W_{\epsilon i,j} = e^{-W_{i,j}/\Omega_{i,j}}$ (the adaptive Gaussian kernel)
    End
    For each $i \in \{1,\ldots,m\}$ (build the degree of each point)
        $d_i = \int_{\hat{X}} W_{\epsilon i,j} d\mu(\hat{x}_j)$
    End
    For each $i,j \in \{1,\ldots,m\}$ (construction of a Markov transition matrix)
        $P_{i,j} = \frac{W_{\epsilon i,j}}{\sqrt{d_i}\sqrt{d_j}}$
    End
    $[\lambda, v] = SVD(P)$ (the eigen-decomposition of the Markov transition matrix)
    For each $i \in \{1,\ldots,m\}$ (construction of the diffusion coordinates with $\eta$ terms)
        $\Phi_i = (\lambda_0 v_0(\hat{x}_i), \lambda_1 v_1(\hat{x}_i), \ldots, \lambda_\eta v_\eta(\hat{x}_i))^T$
    End
    $IDX = f_k(\Phi)$ (clustering of the data in the embedding)
    return $IDX$
End
```

References

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

[2] J. MacQueen, Some methods for classification and analysis of multivariate observations, Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability, vol. 1, University of California Press, Berkeley, CA, 1967, pp. 281–297.

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

[4] T. Calinski, J. Harabasz, A dendrite method for cluster analysis, Communications in Statistics 3 (1974) 1–27.

[5] C.M.G. Gan, J. Wu, Data Clustering: Theory, Algorithms, and Applications, SIAM, 2007.

[6] S.V.D. Arthur, *k*-means++: the advantages of careful seeding, in: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, Louisiana, 2007, pp. 1027–1035.

[7] C. Ding, T. Li, Adaptive dimension reduction using discriminant analysis and *k*-means clustering, in: Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, 2007, pp. 521–528.

[8] G. David, A. Averbuch, R. R. Coifman, Hierarchical clustering via localized diffusion folders, in: Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Fall Symposium Series 2010, 〈http://aaai.org/ocs/index.php/FSS/FSS10/paper/view/2210〉.

[9] R.R.T. Zhang, M. Livny, Birch: an efficient data clustering method for very large databases, SIGMOD Record 25 (1996) 103–114.

[10] R.R.S. Guha, K. Shim, Cure: an efficient clustering algorithms for large databases, in: ACM SIGMOD International Conference on Management of Data, Seattle, WA, 1998, pp. 73–84.

[11] J.S.M. Ester, H.P. Kriegel, X. Xu, A density-based algorithm for discovering clusters in large spatial database with noise, in: International Conference on Knowledge Discovery in Databases and Data Mining (KDD-96), AAAI Press, Portland, Oregon, 1996, pp. 226–231.

[12] L. Kaufman, P. Rousseeuw, Finding Groups in Data—An introduction to Cluster Analysis, John Wiely & Sons, Inc., New York, 1990.

[13] J.G.V. Ganti, R. Ramakrishnan, Cactus: clustering categorical data using summaries, in: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), ACM Press, San Diego, CA, USA, 1999, pp. 73–83.

[14] J.C.D. Barbara, Y. Li, Coolcat: an entropy-based algorithm for categorical clustering, in: Proceedings of the 11th ACM Conference on Information and Knowledge Management (CIKM 02), ACM Press, McLean, Virginia, USA, 2002, pp. 582–589.

[15] R.R.S. Guha, K. Shim, Rock: a robust clustering algorithm for categorical attributes, Journal of Information Systems 25 (2000) 345–366.

[16] J.K.D. Gibson, P. Raghavan, Clustering categorical data: an approach based on dynamical systems, in: Proceedings of the 24th International Conference on Very Large Data Bases (VLDB), Morgan Kaufmann, New York, NY, USA, 1998, pp. 311–322.

[17] H.S.A. Ben-Hur, D. Horn, V. Vapnik, Support vector clustering, Journal of Machine Learning Research 2 (2001) 125–137.

[18] M. Girolami, Mercer kernel based clustering in feature space, IEEE Transactions on Neural Networks 13 (2002) 780–784.

[19] R. Zhang, A. Rudnicky, A large scale clustering scheme for kernel *k*-means, in: Proceedings of the 16th International Conference on Pattern Recognition (ICPR 02), Quebec City, Canada, 2002, pp. 289–292.

[20] J. Couto, Kernel *k*-means for categorical data, Advances in Intelligent Data Analysis VI 3646 (2005) 46–56.

[21] U. von Luxburg, A tutorial on spectral clustering, Statistics and Computing 17 (4) (2007) 395–416.

[22] F.R. Bach, M.I. Jordan, Learning spectral clustering, Proceedings of NIPS, vol. 16, MIT Press, 2004, pp. 305–312.

[23] S.V.R. Kannan, A. Vetta, On clusterings: good, bad and spectral, in: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, 2000.

[24] J. Shi, J. Malik, Normalized cuts and image segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence 22 (2000) 888–905.

[25] M.I.J.A.Y. Ng, Y. Weiss, On spectral clustering: analysis and an algorithm, Advances in Neural Information Processing Systems (NIPS), vol. 14, 2002.

[26] K. Fukunaga, L. Hostetler, The estimation of the gradient of a density function, with applications in pattern recognition, IEEE Transactions on Information Theory in Information Theory 21 (1) (1975) 32–40.

[27] D. Comaniciu, P. Meer, Mean shift: a robust approach towards feature space analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 24 (5) (2002) 603–619.

[28] Y. Cheng, Mean shift, mode seeking, and clustering, IEEE Transactions on Pattern Analysis and Machine Intelligence 17 (8) (1995) 790–799.

[29] D. Comaniciu, An algorithm for data-driven bandwidth selection, IEEE Transactions on Pattern Analysis and Machine Intelligence 2 (2003) 281–288.

[30] J.C. Gower, A general coefficient of similarity and some of its properties, Biometrics 27 (1971) 857–871.

[31] R.K.J. Dougherty, M. Sahami, Supervised and unsupervised discretization of continuous features, in: Machine Learning: Proceedings of the Twelfth International Conference, Morgan Kaufmann Publishers, 1995.

[32] J. Catlett, On changing continuous attributes into ordered discrete attributes, in: Proceedings of the European Working Session on Learning, Springer-Verlag, Berlin, Germany, 1991, pp. 164–178.

[33] R.S. Michalski, A planar geometric model for representing multidimensional discrete spaces and multiple-valued logic functions, Technical Report, University of Illinois at Urbaba-Champaign, 1978.

[34] J. Han, M. Kambert, Data Mining: Concepts and Techniques, Morgan Kaufmann, San Francisco, 2001.

[35] A.V. Oppenheim, R.W. Schafer, Discrete-Time Signal Processing, Prentice-Hall, 1989.

[36] C. Blake, C. Merz, Uci repository of machine learning databases, University of California, Department of Information and Computer Science, Irvine, CA, USA, 1998 〈http://www.ics.uci.edu/mlearn/MLRepository.html〉.

[37] R.R. Coifman, S. Lafon, Diffusion maps, Applied and Computational Harmonic Analysis 21 (2006) 5–30.

[38] R.R. Coifman, S. Lafon, Geometric harmonics: a novel tool for multiscale out-of-sample extension of empirical functions, Applied and Computational Harmonic Analysis 21 (2006) 31–52.

[39] F.R.K. Chung, Spectral Graph Theory, in: AMS Regional Conference Series in Mathematics, vol. 92, 1997.

[40] A.L.L.L.N. Benoudjit, C. Archambeau, M. Verleysen, Width optimization of the Gaussian kernels in radial basis function networks, in: Proceedings of the European Symposium on Artificial Neural Networks, ESANN 2002 Bruges, Belgium, 2002, pp. 425–432.

[41] N. Benoudjit, M. Verleysen, On the kernel widths in radial-basis function networks, Neural Processing Letters 18 (2003) 139–154.

[42] V.D. Sanchez, On the number and the distribution of RBF centers, Neurocomputing 7 (1995) 197–202.

[43] S. Chen, S. Billings, Neural networks for nonlinear dynamic system modelling and identification, International Journal of Control 56 (1992) 319–346.

[44] M. Orr, Introduction to radial basis function networks, 1996 〈http://www.anc.ed.ac.uk/rbf/papers/intro.ps〉.

[45] J. Park, I. Sandberg, Universal approximation using radial basis function networks, Neural Computation 3 (1991) 246–257.

[46] S. Haykin, Neural Networks: a Comprehensive Foundation, second ed., Prentice Hall, 1998.

[47] M. Verleysen, K. Hlavackova, Learning in RBF networks, in: International Conference on Neural Networks (ICNN), Washington, DC, 1996, pp. 199–204.

[48] A. Saha, J. Keeler, Algorithms for better representation and faster learning in radial basis function networks, in: Advances in Neural Information Processing Systems, vol. 2, Morgan Kaufmann Publishers Inc., San Francisco, CA, 1990.

[49] J. Moody, C. Darken, Fast learning in networks of locally-tuned processing units, Neural Computation 1 (1989) 281–294.

[50] K. Pearson, On lines and planes of closest fit to systems of points in space, Philosophical Magazine 2 (1901) 559–572.

[51] M.S.P-N. Tan, V. Kumar, Introduction to Data Mining, Pearson Addison-Wesley, 2006.

Gil David received his M.Sc. degree in Computer Science at the Hebrew University in Jerusalem, Israel in 2003 and received his Ph.D. degree at the Department of Computer Science at Tel-Aviv University in Tel-Aviv, Israel in 2009. Currently he is a postdoctoral associate at the Department of Mathematics, Program in Applied Mathematics at Yale University in New Haven, USA. His main interests are anomaly detection, data mining, diffusion geometry, pattern recognition and data analysis.

Amir Averbuch received his M.Sc. degree in Mathematics at the Hebrew University in Jerusalem, Israel in 1975 and received his Ph.D. degree at the Department of Computer Science at Columbia University in New York, USA in 1983. From 1976 to 1986 he was a Research Staff Member at the Department of Computer Science, IBM T.J. Watson Research Center, Yorktown Heights, NY. In 1987, he joined the School of Computer Science, Tel-Aviv University, where he is now Professor of Computer Science. His main interests are applied harmonic analysis, wavelets, signal/image processing, numerical computation and scientific computing.