Chapter 14
Polar Classification of Nominal Data

Guy Wolf, Shachar Harussi, Yaniv Shmueli, and Amir Averbuch

Abstract Many modern systems record various types of parameter values. Numerical values are relatively convenient for data analysis tools because there are many methods to measure distances and similarities between them. The application of dimensionality reduction techniques for data sets with such values is also a well known practice. Nominal (i.e., categorical) values, on the other hand, encompass some problems for current methods. Most of all, there is no meaningful distance between possible nominal values, which are either equal or unequal to each other. Since many dimensionality reduction methods rely on preserving some form of similarity or distance measure, their application to such data sets is not straightforward. We propose a method to achieve clustering of such data sets by applying the diffusion maps methodology to it. Our method is based on a distance metric that utilizes the effect of the boolean nature of similarities between nominal values (i.e., equal or unequal) on the diffusion kernel and, in turn, on the embedded space resulting from its principal components. We use a multi-view approach by analyzing small, closely related, sets of parameters at a time instead of the whole data set. This way, we achieve a comprehensive understanding of the data set from many points of view.

Keywords Clustering · Unsupervised learning · Diffusion maps · Nominal data

G. Wolf (✉) · S. Harussi · Y. Shmueli · A. Averbuch
School of Computer Science, Tel Aviv University, P.O. Box 39040, Tel Aviv 69978, Israel
e-mail: guy.wolf@cs.tau.ac.il

S. Harussi
e-mail: harussis@tau.ac.il

Y. Shmueli
e-mail: yaniv.shmueli@gmail.com

A. Averbuch
e-mail: amir@math.tau.ac.il

G. Wolf · S. Harussi · A. Averbuch
Department of Mathematical Information Technology, University of Jyväskylä, P.O. Box 35 (Agora), 40014 Jyväskylä, Finland

S. Repin et al. (eds.), *Numerical Methods for Differential Equations, Optimization, and Technological Problems*, Computational Methods in Applied Sciences 27,
DOI 10.1007/978-94-007-5288-7_14,

14.1 Introduction

One of the most sought after tasks nowadays is that of finding patterns and structures in large volumes of high dimensional data. As storage becomes cheaper, network bandwidth increases and sampling technologies become more advanced, the amounts of data collected from various systems increase exponentially. A common trend in many applications is to log and record every action of the system for future analysis. In particular, errors and exceptions are common types of massively recorded items.

The task of unsupervised learning of high-dimensional data has been studied extensively in statistical and machine learning literature. Usually, an assumption concerning the underlying structure of the data is made. One common assumption is that the data consist of classes that represent some form of similarity between data points from the same class. Detecting the classes and classifying the data points is often done by clustering algorithms applied to a data representation, which preserves some desired properties (i.e., similarities) of the data set.

Classical clustering algorithms are loosely divided into two major categories. *Partitional* algorithms aim at finding an optimal partition of the data set into the desired clusters. *Hierarchical* algorithms, on the other hand, aim at constructing a hierarchy of clusters from the data. This is usually done in several iterations, each refining the previous one while providing the hierarchy with an additional level. Classic partitioned algorithms are k-Means [27] and its variants (e.g., Fuzzy c-Means [3] and k-Prototypes [22], which adapts k-Means to handle categorical values). Typical hierarchical algorithms are BIRCH [42], CURE [18], and Chameleon [25]. Modern algorithms also use additional approaches. Density-based clustering algorithms, such as DBSCAN [16], DENCLUE [20], and OPTICS [2], define clusters as dense areas separated by sparse ones. Grid-based clustering algorithms analyze cells rather than single samples, thus being more efficient computationally. Some typical examples of such algorithms are STING [37], STING+ [38], WaveCluster [32], CLIQUE [1], GDILC [43], and Localized Diffusion Folders [12].

The data types handled by a clustering algorithm can be divided into three major categories [36]: numerical, nominal, and transactional. Most of the study of clustering algorithms deals with numerical data sets, in which there is a relatively simple notion of proximity or similarity between samples. Most of the algorithms mentioned above deal with numerical data; additional examples can be found in [4, 17, 24].

While there are significantly less clustering algorithms designed for handling nominal data, some classic examples do exist. Notable examples of such clustering algorithms are k-Modes [21] and ROCK [19], both of which deal directly with nominal data, and OPOSSUM [34], which deals with ordinal data (i.e., discrete values with order). A different approach is to transform the categories in the data to numerical values. This can be done either by using some order between them or by using binary encoding (1 means a category that appeared for a sample while 0 means the opposite), which would result in a large but very sparse data set. Some examples of this approach can be found in [31, 33].

The last category (i.e., transactional data) is poorly structured in the sense that each sample, also called transaction, contains a variable set of values describing it. Since there is no constant order to the properties of an entry, and their amount may change from sample to sample, comparing samples in such data sets becomes a fairly complicated task. Such data sets may, in some cases, be flattened and reformatted into a nominal format (e.g., by setting an absolute order to the properties and using a special value to express N/A values), in which case the previously mentioned algorithms can be applied. Some examples of algorithms that directly analyze transactional data are LargeItem [35], SLR [41] and CLOPE [39]. Some recent methods for analyzing both nominal and transactional data can be found in [36].

In recent years, dimensionality reduction techniques were used to obtain low-dimensional representations that amplify the similarities between data points. A popular and successful dimensionality reduction method for this purpose is Diffusion Maps (DM) [8, 26]. This method is based on defining the similarities between data points by using a diffusion kernel, which describes a diffusion process (i.e., random walks) on the data set. The first few eigenvectors of this kernel can be used to obtain a low-dimensional representation of the data set, in which the Euclidean distances between data points correspond to random-walk distances, also called diffusion distances, between their original (high-dimensional) counterparts.

Usually, classes of similar data points appear, in the resulting low-dimensional space, as dense clusters separated by sparse areas [10]. By using a density function one can detect these clusters and the data points within them and thus achieve the desired analysis. This methodology was applied for classification and anomaly detection tasks [10]. In the case of anomaly detection, the classes were considered to represent normal behavior while data points that did not belong to any class were considered to be anomalous.

The DM methodology is based on similarities defined by a suitable distance metric. A Gaussian kernel is then used to give the notion of neighborhoods and, with proper normalization over each neighborhood, the diffusion kernel is obtained. When the data contains numeric measurements, there is a wide variety of distance metrics that can be used. Common metrics are the l_1 and l_2 metrics, which give good results in many practices. When, on the other hand, the data contains nominal values, finding a suitable distance metric is less obvious, as nominal values can be either equal or unequal with no notion of distance or proximity between different values.

One recent approach for handling nominal-valued data sets uses the Hamming distance as the metric that the diffusion kernel is based on. A method, which is based on it, to analyze mixed data sets containing both numeric- and nominal-valued parameters is presented in [13]. This approach can prove useful when there is a one-to-one correspondence between the rows in the data set and the analyzed items, and no bias is created by the dependencies between the parameters of the data set.

In this paper, we deal with a more general nominal-valued data set. We allow several data rows to be related to the same analyzed item. Also, we do not assume that the parameters of the data set are unrelated and we take into account possible bias due to dependencies between them. We define two possible distance metrics

that can be seen as an extension of the Hamming distance. We apply the DM method using these metrics to analyze items, each of which is represented by several rows from the data set.

The structure of the embedded space, which is achieved by the described method, is uniquely different from the ones that appear in many other studies. Instead of similar items being concentrated in dense clusters, these items form rays emanating from a common center near the origin. This unique geometry is a result of the discrete nature of the used distance metric and the resulting diffusion kernel. The clusters are thus identified as having common directions. The rays are not dense when represented by Cartesian coordinates. In polar coordinates, or at least those that correspond to angles, the data points on the same ray are very similar. This observation provides a clustering method to be applied to the embedded space and the method thus called polar clustering.

In addition to the new distance metric used in this paper, we also use a multi-view approach for analyzing the data in an unbiased manner. Multi-view techniques have been applied to many data analysis problems. In these problems, the studied samples consist of different subsets of parameters that, in some cases, even come from different sources. Each of these subsets contributes partial knowledge for the clustering process. Fusing them together can lead to an improved solution. This is done by utilizing the agreements among different views, each representing a single subset. The challenge in these techniques is to find the right parameter partition into subsets, and to understand the weight of each subset and its potential contribution to the learning process. Then, one needs to apply proper normalization and blending methods between the subsets while overcoming problems like cross-dependencies, normalization, repetition, and over- or under-weighting of parameters and subsets.

One common method for dealing with multiple sources (or subsets) of data parameters is to simply ignore the distinctions and concatenate parameters from all the sources into one vector. This represents an implicit assumption that all the parameters, from all the sources, are directly comparable, which is usually not true. Multi-view methods, on the other hand, consider the differences between the subsets and use them to better train the classifier that will be used to analyze the data. One method for applying such a technique is to design a special graph that is based on multiple sources and to use the kernel induced by the graph as the input for a kernel based clustering algorithm [15].

Other multi-view algorithms train two independent classifiers that bootstrap by providing each other with labels for the unlabeled data. The training algorithms tend to maximize the agreement between the two independent classifiers [6, 40]. It has also been shown that the disagreement between two independent classifiers is an upper bound for the error rate of a classifier achieved by uniting them together [9]. This could explain the recent success of multi-view learning in motivating clustering methods that are based on a multi-view approach.

Multi-view classification methods are sometimes called, in the literature, co-training or co-clustering. Under these names, they have been studied thoroughly in [5], where multi-view versions were presented for familiar partitioning methods, such as k-Means, k-Medoids, and EM. Another method that can be used is to

construct a specific kernel (similarity) matrix for each view, and then to blend the matrices into a single kernel matrix. This combined kernel can be used to apply further analysis to the data as a whole, e.g., by training a support vector machine that is based on it [14].

The application of multi-view approaches in conjunction with the DM methodology can be seen in [10, 12, 28]. These works use a hierarchy of views to provide a complete analysis of the data. Construction of each level in the hierarchy by pruning clusters in the previous level and determining the affinities between the pruned clusters is given in [10, 12] whose theoretical justification is given in [11]. This affinity is based on the relations revealed by examining small views, each of which contains samples in the two clusters compared by the view.

The other mentioned paper [28] is based on organizing the parameters in a hierarchical structure according to what they measure. Then, it works in a bottom-up fashion, each time executing the DM algorithm on a single view (i.e., a node in the hierarchy). The densities around the points in the embedded spaces of the children of a certain node are used as an input for the DM algorithm applied to that node.

The paper has the following structure. Section 14.2 describes the problem setup. Section 14.3 defines the distance metrics, describes the geometry of the resulting embedded space and explains the clustering and classification method. Section 14.4 demonstrates two applications to real-life data sets of the classification method.

14.2 Problem Setup

Assume that the data set X contains m observations where each observation details the values of l nominal parameters. Thus, X can be seen as a $m \times l$ matrix that contains nominal (i.e., categorical) values. The observations in the data set are not necessarily unrelated and several observations may refer to a single studied item or subject in the analysis. One example for such data sets is exception (i.e., software errors) analysis where several exceptions may relate to a single malfunction, which can be identified by the machine and time of these exceptions.

We begin to examine the data set X by defining the subjects of the analysis and relating each observation to the subject to which it refers. This can be done either by external labeling or by grouping the observation according to the values of (some of) their parameters. We denote the set of all subjects by S and its size by $n = |S|$. For each subject $s \in S$, the set of observations in X that refer to it is denoted by X_s.

We assume there is some relation between the parameter sets. Viewing them as a whole might be biased by the number of parameters relating to each perspective. For example, if there are five parameters describing the software components (e.g., process, class, thread) and two describing the thrown exception (e.g., error type), an analysis based on all these parameters would be biased towards the software perspective. To cope with these situations, we use a multi-view approach to analyze the behavior of the subjects. We divide the parameters of the data set to several perspectives, or views, and analyze each of them separately. This way, we provide a complete, unbiased analysis of the data set from several points of view.

The main goal of the analysis in this paper is to find structures and patterns in the data set. We do this by clustering the subjects to a set of classes in each perspective. An examination of the common categories (i.e., nominal values of the parameters) in each class, from each perspective, provides an understanding of the structure of the data set and the types of subjects in it. Also, one can deduce the relations between the subjects based on these understandings.

14.3 Classification Method

In this section, we present the classification method that is applied to each view (i.e., perspective) separately. We start by constructing the view, as a new data set, according to the subjects in S and the parameters selected for the view. Then, we describe the application of DM using a new distance metric. Finally, the structure of the embedded space and the clustering method applied to it are described.

14.3.1 Construction of a View

The analysis begins with selecting the parameters, from the original data set, to be used in the current view. Each observation in X combines nominal values of these parameters. Each subject $s \in S$ is related to some observations in X and so it is described by several combinations of values of the selected parameters. We will denote the set of these combinations for a subject $s \in S$ by V_s. We denote the set of all such combinations in the data set by $V = \cup_{s \in S} V_s$ and their number by $d = |V|$. From this point on, when we refer to combinations of parameter values, or just combinations, we mean the described combinations in V (or V_s for a subject s), unless specifically stated otherwise.

A single view is described by the subjects in S, the combinations in V, and the relations between them. We suggest two approaches to describe and handle this information: the boolean approach and the counter approach. We will describe them side by side in this paper. The boolean approach describes the relation between a subject $s \in S$ and a combination $v \in V$ by a boolean value stating whether or not v was reported for s in the data set, i.e. $v \in V_s$. The counter approach adds the information of how many times this combination reported for s. This describes the relation by a number that counts the observations related to s that contain the combination v.

Formally, each approach constructs an $n \times d$ matrix that describes the view. Each row in this matrix corresponds to a subject in $s \in S$ and each column corresponds to a combination $v \in V$. The boolean approach constructs the matrix B where each cell is defined as

$$[B]_{sv} = b(s, v) \triangleq \begin{cases} 1, & v \in V_s, \\ 0, & v \notin V_s, \end{cases} \quad s \in S,\ v \in V. \tag{14.1}$$

The counter approach constructs the matrix C where each cell is defined as

$$[C]_{sv} = c(s,v) \triangleq \begin{cases} |\{x \in X_s | x \sim v\}|, & v \in V_s, \\ 0, & v \notin V_s, \end{cases} \quad s \in S,\ v \in V, \tag{14.2}$$

where an observation $x \in X$ is similar to a combination $v \in V$ only if this combination of parameter values appears in x. For a subject $s \in S$, we denote its row in B by $b_s = b(s,\cdot)$ and its row in C by $c_s = c(s,\cdot)$. The constructed matrices provide a suitable presentation of the subjects for the analysis from the desired perspective, and whichever of the two described approaches we choose, we will refer to the constructed matrix for that approach as the current view's data set or simply the current view.

Finally, since the methodology we use for analyzing the current view is based upon the distances between data points, we must define a suitable distance metric between subjects for each approach. For the boolean approach, we define the following distance metric between the rows that represents two subjects $s, t \in S$ in the matrix B:

$$\|b_s - b_t\|_b \triangleq \frac{\sum_{v \in V} [b(s,v) \oplus b(t,v)]}{\sum_{v \in V} [b(s,v) \vee b(t,v)]}, \tag{14.3}$$

where the logical operators treat 1 and 0 as *true* and *false*, respectively, and the summation (and division) treat them as numbers. The counter approach defines the following distance metric between the rows that represent two subjects $s, t \in S$ in the matrix C:

$$\|c_s - c_t\|_c \triangleq \frac{\sum_{v \in V} |c(s,v) - c(t,v)|}{\sum_{v \in V} |c(s,v) + c(t,v)|}. \tag{14.4}$$

Both metrics measure the difference between the combinations related to the subjects s and t relative to the total number of combinations reported for any of them. They are similar to the *Jaccard Similarity Coefficient* [23] and the *Tanimoto distance* [29], which are used to compute the similarity and diversity between two data sets.

The rest of the analysis, which is presented in the next sections, does not depend on which approach we use and so we define general notations that will refer to the selected approach. B and C denote the constructed boolean and binary matrix, respectively, denoted by U. We denote by u_s the row of this matrix that represents the subject $s \in S$ (i.e., u_s is b_s or c_s depending on the selected approach). The distance between the rows of U, which represents two subjects $s, t \in S$, according to the selected approach metric, is denoted by $\|u_s - u_t\|_u$. With these notations and the represented constructed view, we are ready to apply the DM to this view and to analyze the lower dimensional representation provided by it.

14.3.2 Application of DM

The DM method analyzes the view's data set by exploring its geometry [8]. It is based on defining the isotropic kernel

$$k_\varepsilon(s,t) \triangleq e^{-\frac{\|u_s - u_t\|_u}{\varepsilon}}, \tag{14.5}$$

where $s,t \in S$ are two subjects and ε is a meta-parameter of the algorithm. This kernel represents the affinities between the two subjects from the perspective of the current view.

The kernel may be viewed as a construction of a weighted graph over the view. The subjects are used as vertices and the weights of the edges are defined by the kernel k_ε. The degree of each subject (i.e., vertex) $s \in S$ in this graph is

$$q_\varepsilon(s) \triangleq \sum k_\varepsilon(s,t). \tag{14.6}$$

Normalizing the kernel with this degree produces an $n \times n$ row stochastic transition matrix M whose cells are $[P]_{st} = p(s,t) = k_\varepsilon(s,t)/q_\varepsilon(s)$, $s,t \in S$, which defines a Markov process (i.e., a diffusion process) over the subjects.

The dimensionality reduction achieved by this diffusion process is a result of spectral analysis of the diffusion kernel. Thus, it is preferable to work with a symmetric conjugate to P that we denote by A and its cells are

$$[A]_{st} = a(s,t) = \frac{k_\varepsilon(s,t)}{\sqrt{q_\varepsilon(s)}\sqrt{q_\varepsilon(t)}} = \sqrt{q_\varepsilon(s)}\,p(s,t)\frac{1}{\sqrt{q_\varepsilon(t)}}, \quad s,t \in S. \tag{14.7}$$

The eigenvalues $1 = \lambda_0 \geq \lambda_1 \geq \ldots$ of A and their corresponding eigenvectors ϕ_i, $i = 0,1,\ldots$, are used to obtain the desired dimensionality reduction by mapping each subject s onto the point $\Phi(s) = (\lambda_i \phi_i(s))_{i=0}^{\delta}$ for a sufficiently small δ, which depends on the decay of the spectrum of A [8, 26]. This construction is also known as the Laplacian of the graph constructed by the kernel [7]. We denote the resulting low-dimensional vector representing a subject s by $\tilde{u}_s = \Phi(s)$, and the set of all such vectors by $\tilde{U}$. We also use the notations $\tilde{b}_s$ (and $\tilde{B}$) or $\tilde{c}_s$ (and $\tilde{C}$), when referring specifically to the boolean approach or the counter approach, respectively.

14.3.3 Construction of the Classes

In practice, for most data sets of the form dealt in this paper (see Sect. 14.2), the vectors in $\tilde{U}$ will have a unique geometry. They form rays emanating from a common center near the origin. This property is due to the discrete nature of the distance metric we used (14.3) or (14.4) and, in turn, the Gaussian kernel (14.5) and the diffusion kernel (14.7) constructed by it. Indeed, the inner product of two vectors $\tilde{u}_s, \tilde{u}_t \in \tilde{U}$

in the embedded space is

$$\langle \tilde{u}_s, \tilde{u}_t \rangle = \left\langle \Phi(s), \Phi(t) \right\rangle = \sum_{i=0}^{\delta} \lambda_i^2 \phi_i(s) \phi_i(t). \tag{14.8}$$

We recall that the eigenvalues of A^2 are $\lambda_0^2, \lambda_1^2, \lambda_2^2, \ldots$ [8, 26]. Thus, according to the spectral theorem and the fast decay of the eigenvalues of A, we get

$$\langle \tilde{u}_s, \tilde{u}_t \rangle = \sum_{i=0}^{\delta} \lambda_i^2 \phi_i(s) \phi_i(t) \approx a^2(s,t) = \left[A^2\right]_{st}. \tag{14.9}$$

Therefore, a small discrete set of values taken by the diffusion kernel leads to a small discrete set of inner products, which determines the angles between the vectors in the embedded space. Since there is a small variety of angles in the embedded space, similar vectors have approximately the same directions (from the origin) and unrelated ones have relatively wide angles between them. This approach is related to cosine similarity, which uses the cosine of the angle between two vectors to define the similarity between them, in the embedded space. The cosine similarity is used to compare documents in text mining [30] and to measure the similarity between clusters.

An examination of the used distance metrics presents a possible explanation for the described structure of the kernel. In both approaches (14.3) and (14.4), totally unrelated subjects have a distance of 1 between them while completely correlated ones have a 0 distance. The range of possible values between 0 and 1 (for two compared subjects) depends on the number of combinations reported for the compared subjects. As more combinations are reported for them, it leads to more possible values. In many cases, however, the maximal number of combinations reported for a single subject is no more than a few dozen combinations while the common number of them for a single subject is less than a dozen. Therefore, the range of possible values for the distance between two subjects is, in practice, fairly limited.

The geometry of the embedded space suggests a new clustering method to be applied to it. Instead of measuring density in Cartesian coordinates, we measure it in polar coordinates. Specifically, the vectors are clustered in this space according to their angle coordinates. First, we find the dominant directions of the rays where large concentrations of vectors lie. Then, we associate each vector with the closest ray. This method yields a set of classes, each of which contains vectors representing similar (i.e. correlated) subjects in the original data set of the currently analyzed view.

One issue that should be pointed out is the concentration of some points near the origin. The embedding process preserves only the principal components of the data. There are many cases in which some of the subjects are completely unrelated to any other subject in the view. Such data points should have a negligible affinity to every other data point; therefore they have an inner product of approximately 0 with all other vectors in $\tilde{U}$. If the dimensionality of the embedded space was large

enough we would see such vectors as almost orthogonal to all the other vectors, but since we deliberately use a low-dimensional embedded space, only their projection on this space is seen. The projections of such data points are thus seen as very close to the origin as they are almost orthogonal to the observed space. Therefore, before applying our analysis we clear a dense area around the origin, which contains all the unrelated vectors to the observed space.

The vectors in the dense central area can be further explored in the same way as the original view. A second iteration might reveal some correlations between the subject corresponding to the vectors in the central area, which were masked by the rest of the vectors in the first one. This would specially be the case if the dimension of the embedded space in the first iteration was too low to encompass the nature of the examined view. If, on the other hand, it was sufficient to represent the view, the next iteration would show a clutter of uncorrelated vectors with no apparent relations between them.

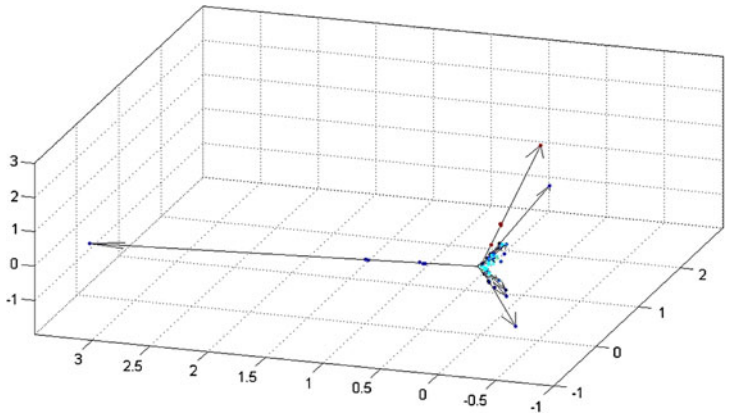
14.4 Empirical Results

In this section, we present two applications of the polar classification method for analyzing real-life data sets. The first example demonstrates the usage of this method to classify malfunctions from an error monitoring log. The second example shows tools for supporting management decisions during the testing phases (i.e., QA cycles) of a software development process that is based on the polar classification method.

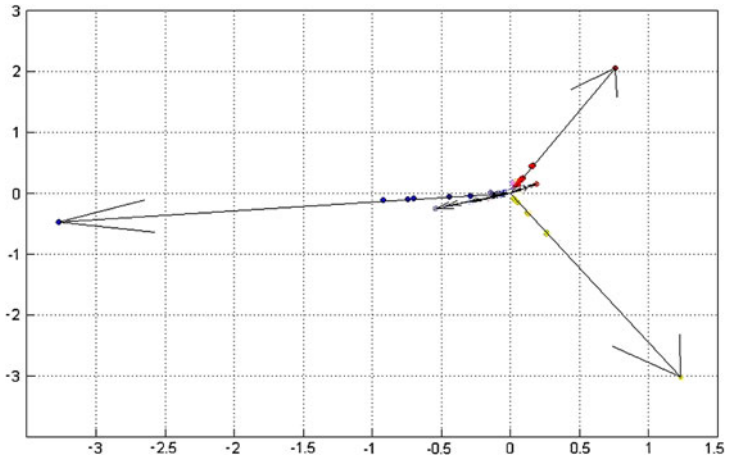
14.4.1 Error Monitoring

We applied the polar classification method to a data set that contains a log of errors that were recorded by a wide-scale distributed system. Each entry in the log records information about the malfunctioning server, the time of the error and the details of the error. We used the polar clustering and classification method to classify distinct events, which are identified by a server name and the time of the event according to the components that reported the malfunctions.

In order to achieve the desired clustering and classification we used the boolean approach to construct a 4018×719 flag matrix, which indicates the components that were malfunctioning in each event (i.e., specific server and time). Each row in this matrix corresponds to a single event and each column corresponds to a single component. Thus, there were 4018 distinct events and 719 distinct components in the analyzed log. Next, we constructed the boolean distance metric (14.3) between rows of this flag matrix and applied the DM method according to the calculated distances. We used the first 20 eigenvectors of the diffusion kernel (defined by (14.7)). Thus, our embedded space was 20-dimensional. This space is illustrated in Fig. 14.1.



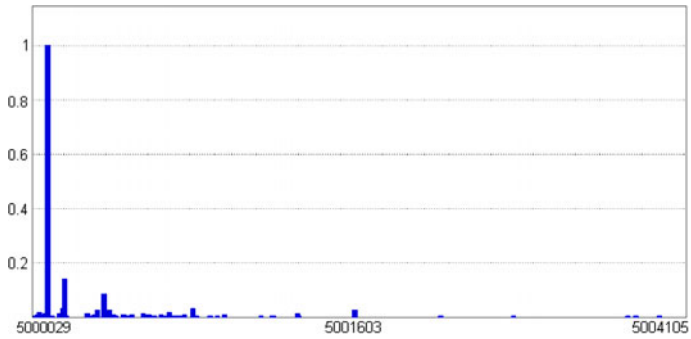
(a) First three axes of the embedded space



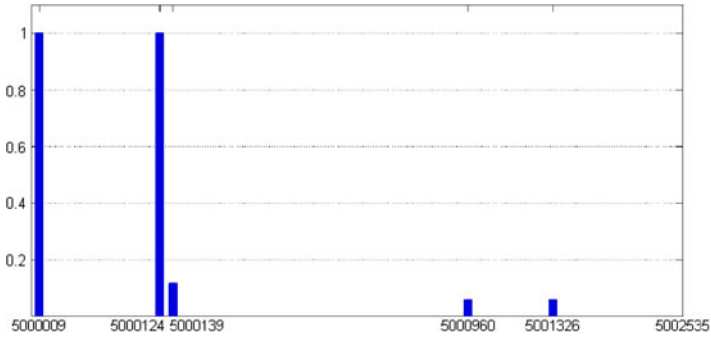
(b) Axes 7 and 8 of the embedded space

Fig. 14.1 An illustration of the 20-dimensional embedded space. The points correspond to the examined events, and they are colored according to the detected clusters (i.e., rays). The vectors display the detected rays on which the points are concentrated

The events in the embedded space form distinct rays emanating from a mutual central point. These rays were detected and the events were classified according to the ray on which they lie. Next, we examined the events in each of the resulting classes. Every class had a few components that were reported in almost all of the events in the class. We refer to such components as the dominant components of the examined class. The bar plot in Fig. 14.2(a) demonstrates a class with a single dominant component and the one in Fig. 14.2(b) shows an example of a class with two dominant components. Each bar in these plots represents a single component. The height of the bar indicates how many of the events, in the examined class, reported it.



(a) A class with one dominant component (DB adapter in this case)

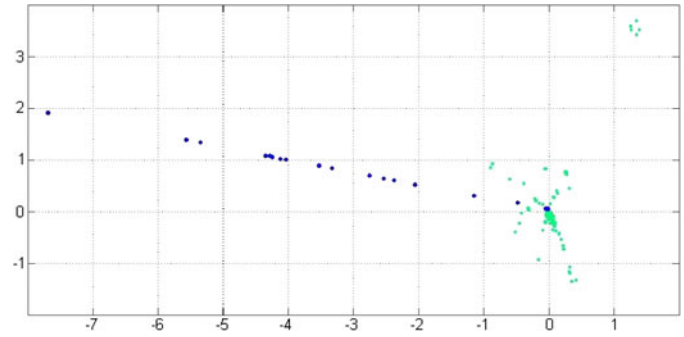


(b) A class with two dominant components (input fetcher and data parser in this case)

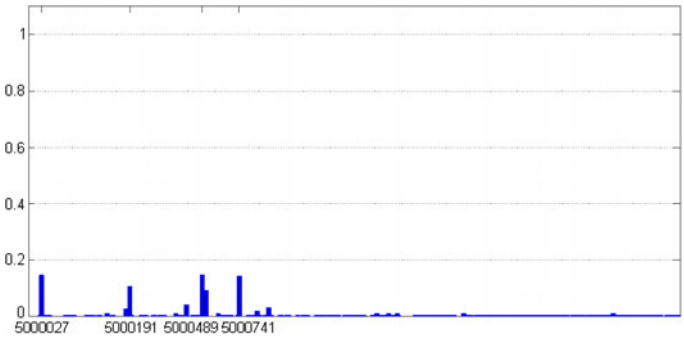
Fig. 14.2 Examples of classes with one and two dominant components. The bar plot shows for each component how many of the events in this class reported it (e.g., 0 means none and 1 means all)

The embedded space in this case also had a dense central area, which contains 1539 events (out of the original 4018) that were unrelated to the detected classes. We examined this central area by applying the same analysis to the events in it. The classes that resulted from this analysis showed more subtle patterns than these from the first iteration. An example of such a pattern is shown in Figs. 14.3, 14.4, and 14.5, which presents three sections of a single class (i.e., ray).[1] When all the events in the class are considered (Fig. 14.3(a)), the dominant components of the class are not apparent (Fig. 14.3(b)). If, on the other hand, we only consider events, which are very far away from the central area (Fig. 14.4(a)), then only five components, which are reported for all of these events, are left as dominant (Fig. 14.4(b)). Finally, by filtering out only the events that are very close to the central area and

[1]The dominant components are clear when points that are too close to the central area are not considered. The dominant components in this case have various interrelated functions specific to the analyzed system.



(a) Eigenvectors 5 and 10 of the embedded space of the second iteration



(b) Dominant components

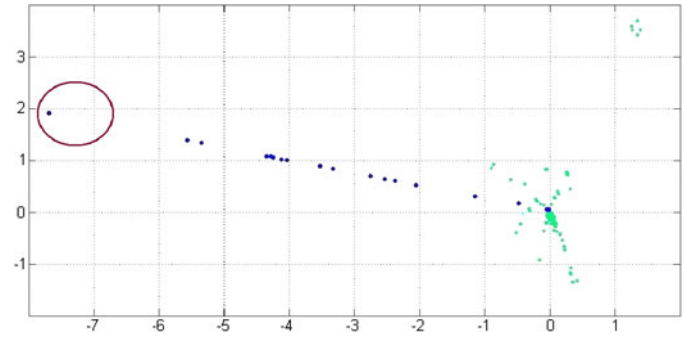
Fig. 14.3 A class with a few dominant components in the second iteration: An entire class

considering the remaining events (Fig. 14.5(a)), we get the bar plot in Fig. 14.5(b), which still indicates about three dominant components of this class and two less dominant ones.

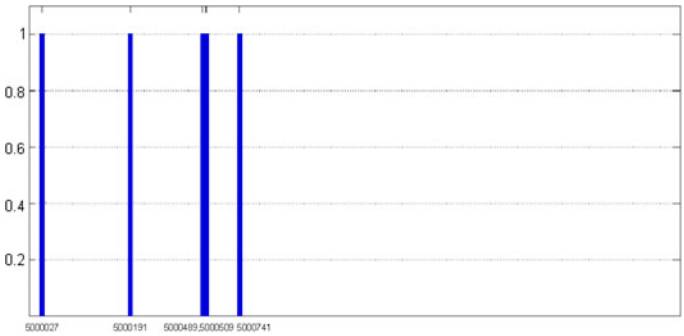
We used 25 dominant classes from each of the conducted iterations. For each class we identified its dominant components. Therefore, the original 4018 events were clustered into 50 different classes that covered 3292 events. The remaining 726 events lied in the central area of the embedded space of the second iteration and did not show any special correlations. These results provide vital information about the common combinations of malfunctioning components, which cover over 75 % of the events in the log. This information can be used both for root-cause analysis in order to find the programmatic defects that cause these problems and as a guidance tool for the development of future versions of the system.

14.4.2 Quality Assurance

The term quality assurance (QA) in software development refers to a phase in the development life cycle, in which the developed product is tested for inherent oper-



(a) Eigenvectors 5 and 10 of the embedded space of the second iteration

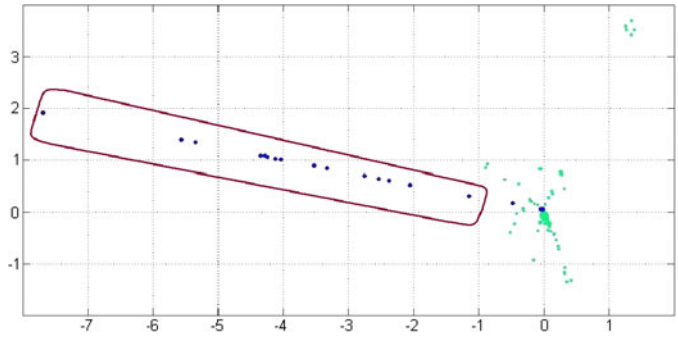


(b) Dominant components

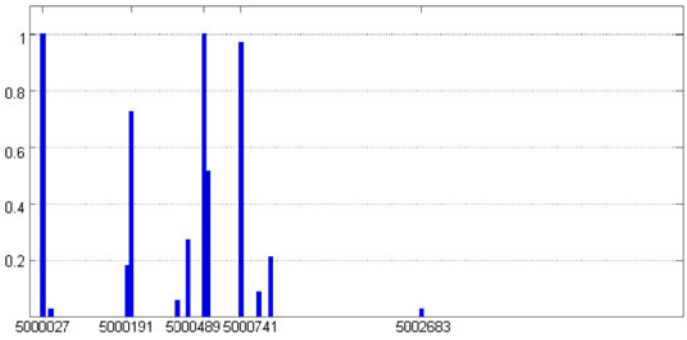
Fig. 14.4 A class with a few dominant components in the second iteration: The farthest points from the central area

ational errors. Usually, several testing cycles are applied and the defects found in each cycle are fixed before the next cycle begins. In most development teams, a defect management tool is used to log all the detected bugs and prioritize them based on their impact and severity. When developing and maintaining multiple configurations and several different versions of the software, it is not trivial to determine the priority of a given bug. In order to do so, potential benefits across all configurations and versions have to be considered. For example, a small set of defects can, in fact, be the root cause of a possible instability in several configurations.

We applied the polar classification method to the analyzing software defects tasks, then classifying the configurations and the versions of a project based on the detected defects. We used a data set that contains information about defects that were detected in several configurations and in several versions. This information was recorded during several testing cycles of a software project. The detail that were recorded for each defect are the defected features, the defect type, its detection, the configuration on which it occurred and the software version in which it occurred.



(a) Eigenvectors 5 and 10 of the embedded space of the second iteration

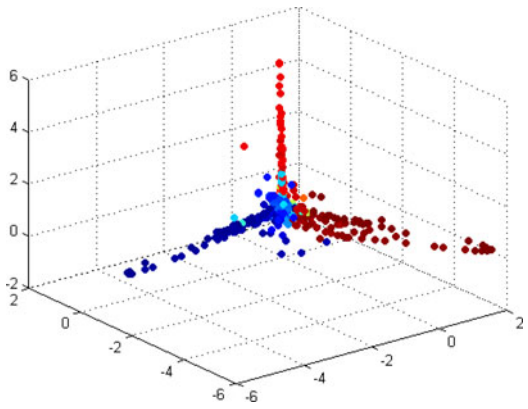


(b) Dominant components

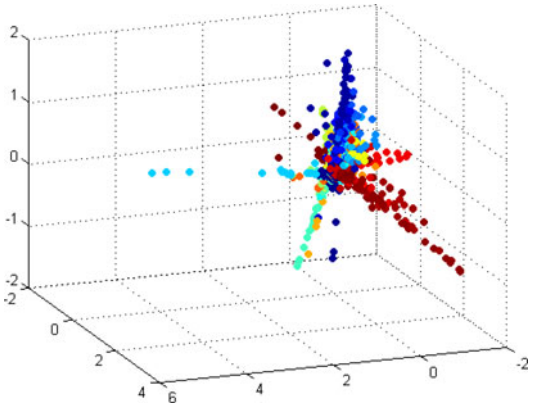
Fig. 14.5 A class with a few dominant components in the second iteration: Points not too close to the central area

We defined the subjects of the analysis, in this case, as pairs of the version (i.e., cycle) and configuration of the software. They are clustered by the defected features detected in each of these pairs. We used the counter approach to construct a counter matrix, where each row corresponds to a unique pair of version and configuration and each column corresponds to a single software feature. The data set contained 1426 distinct version-configuration pairs and 2275 different defected features (i.e., the counter matrix was a 1426×2275 matrix). Then, we used the counter distance metric (14.4) to compute the distances between rows of the counter matrix where the DM method is applied based on the computed distances. The resulting 20-dimensional embedded space is illustrated in Fig. 14.6.

The embedded space in this example is similar in shape to the one in the previous example. Again, the embedded data points, which correspond to version-configuration pairs, are organized on rays emanating from a mutual center. The version-configuration pairs are clustered according to the rays on which they lie in the embedded space. We performed a second iteration of the analysis, similar to the one explained in the previous example, on those that lie in the central area (i.e., the points that are unrelated to any cluster). In the first iteration, 981 of the 1426



(a) First three eigenvectors of the embedded space

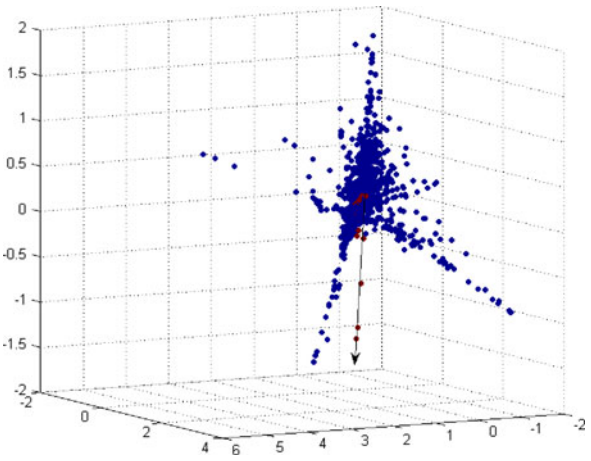


(b) Eigenvectors 4, 5, and 6 of the embedded space

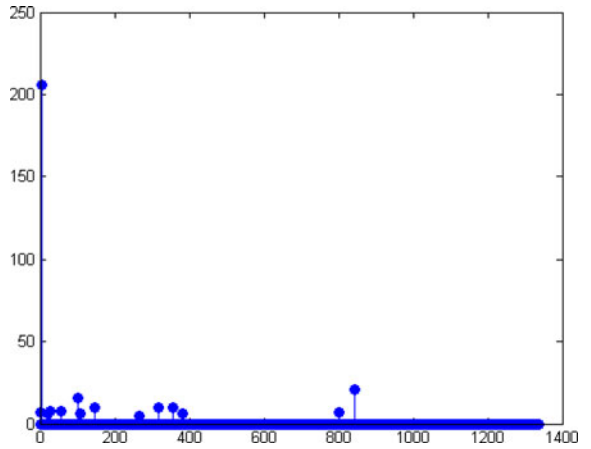
Fig. 14.6 An illustration of the 20-dimensional embedded space. The points correspond to the software versions. They are colored according to the detected clusters (i.e., rays)

data points were clustered into 86 different clusters. The second iteration, which was performed using the remaining 535 data points, detected 30 additional clusters that encompassed 219 data points (out of the 535 analyzed). Overall, 1207 version-configuration pairs were clustered into 116 classes, leaving 219 unclassified pairs, which were not correlated with the rest of the pairs.

The detected classes show a situation similar to the one in the previous example; i.e., each class has a small set of dominant defected features that occur in almost every version-configuration pair in the class. Figure 14.7 demonstrates this result by showing a single class with one dominant defected feature reported by most of the data points in it. The achieved versions and configuration clustering by using the polar classification method, can be used to find similar behaviors between different setting in the system. Defects (specifically defected features) can be prioritized according to the classes in which they are detected. There are classes that contain many configuration and many versions should indicate wide and long standing problems.



(a) A class with one dominant component



(b) A dominant component of the separated class

Fig. 14.7 An example of a class with one dominant component. The bar plot shows the sum of reports made by all software versions in this class for each component

14.5 Conclusions

We presented a distance metric that utilizes the DM methodology for analyzing nominal data sets. We used a multi-view approach to analyze a data set from several perspectives instead of examining it as a whole. From each perspective, a diffusion kernel was constructed and the analyzed items of the perspective were mapped to Euclidean space using spectral analysis of this kernel. The embedded items formed rays in the embedded space that were emanating from a common central area, which is the new origin. These rays indicate a dominant pattern in the data set, and can be used to cluster and classify the analyzed items. The results of this clustering can also be used as a basis for further analysis of the data, e.g., by further analyzing the similarities between the clusters and rating each of them according to its impact on the other clusters.

References

1. Agrawal R, Gehrke J, Gunopulos D, Raghavan P (1998) Automatic subspace clustering of high dimensional data for data mining applications. In: SIGMOD '98: proceedings of the 1998 ACM SIGMOD international conference on management of data. ACM, New York, pp 94–105
2. Ankerst M, Breunig MM, Kriegel HP, Sander J (1999) OPTICS: ordering points to identify the clustering structure. In: SIGMOD '99: proceedings of the 1999 ACM SIGMOD international conference on management of data. ACM, New York, pp 49–60
3. Babuška R (1998) Fuzzy modeling for control. Kluwer, Norwell
4. Berkhin P (2006) A survey of clustering data mining techniques. Grouping Multidimensional Data Cl(c):25–71
5. Bickel S, Scheffer T (2004) Multi-view clustering. In: ICDM '04: proceedings of the fourth IEEE international conference on data mining. IEEE, Washington, pp 19–26
6. Blum A, Mitchell T (1998) Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on computational learning theory, Madison, WI, 1998. ACM, New York, pp 92–100
7. Chung F (1997) Spectral graph theory. CBMS regional conference series in mathematics, vol 92. AMS, Providence
8. Coifman RR, Lafon S (2006) Diffusion maps. Appl Comput Harmon Anal 21(1):5–30
9. Dasgupta S, Littman ML, McAllester D (2001) PAC generalization bounds for co-training. Technical report, AT&T Labs-Research
10. David G (2009) Anomaly detection and classification via diffusion processes in hyper-networks. PhD thesis, School of Computer Science, Tel Aviv University
11. David G, Averbuch A (2012) Hierarchical data organization, clustering and denoising via localized diffusion folders. Appl Comput Harmon Anal 33(1):1–23
12. David G, Averbuch A (2011) Localized diffusion. Part II: Coarse-grained process (submitted)
13. David G, Averbuch A (2012) SpectralCAT: categorical spectral clustering of numerical and nominal data. Pattern Recognit 45(1):416–433
14. de Diego IM, Munoz A, Moguerza J (2010) Methods for the combination of kernel matrices within a support vector framework. Mach Learn 78:137–174
15. de Sa VR, Gallagher PW, Lewis JM, Malave VL (2010) Multi-view kernel construction. Mach Learn 79(1):47–71
16. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD '96: proceedings of the 2nd international conference on knowledge discovery and data mining. AAAI, New York, pp 226–231
17. Everitt B, Landau S, Leese M (2001) Cluster analysis, 4th edn. Arnold, London
18. Guha S, Rastogi R, Shim K (1998) CURE: an efficient clustering algorithm for large databases. In: SIGMOD '98: proceedings of the 1998 ACM SIGMOD international conference on management of data. ACM, New York, pp 73–84
19. Guha S, Rastogi R, Shim K (2000) ROCK: a robust clustering algorithm for categorical attributes. Inf Syst (Oxf) 25(5):345–366
20. Hinneburg A, Keim DA (1998) An efficient approach to clustering in large multimedia databases with noise. In: KDD '98: proceedings of the 4th international conference on knowledge discovery and data mining, pp 58–65
21. Huang Z (1997) A fast clustering algorithm to cluster very large categorical data sets in data mining. In: SIGMOD-DMKD '97: workshop on research issues on data mining and knowledge discovery
22. Huang Z (1998) Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min Knowl Discov 2(3):283–304
23. Jaccard P (1901) Étude comparative de la distribution florale dans une portion des Alpes et des Jura. Bull Soc Vaud Sci Nat 37:547–579
24. Jain AK, Murty MN, Flynn PJ (1999) Data clustering: a review. ACM Comput Surv 31(3):264–323

25. Karypis G, Han EH, Kumar V (1999) Chameleon: hierarchical clustering using dynamic modeling. Computer 32(8):68–75
26. Lafon S (2004) Diffusion maps and geometric harmonics. PhD thesis, Yale University
27. MacQueen J (1967) Some methods for classification and analysis of multivariate observations. In: Proceedings of the 5th Berkeley symposium on mathematical statistics and probability. Statistics, vol I. Univ California Press, Berkeley, pp 281–297
28. Rabin N (2010) Data mining dynamically evolving systems via diffusion methodologies. PhD thesis, School of Computer Science, Tel Aviv University
29. Rogers DJ, Tanimoto TT (1960) A computer program for classifying plants. Science 132(3434):1115–1118
30. Salton G, Buckley C (1988) Term-weighting approaches in automatic text retrieval. Inf Process Manag 24(5):513–523
31. Sebban M, Nock R (2002) A hybrid filter/wrapper approach of feature selection using information theory. Pattern Recognit 35(4):835–846
32. Sheikholeslami G, Chatterjee S, Zhang A (2000) WaveCluster: A wavelet-based clustering approach for spatial data in very large databases. VLDB J 8(3–4):289–304
33. Stanfill C, Waltz D (1986) Toward memory-based reasoning. Commun ACM 29(12):1213–1228
34. Strehl A, Ghosh J (2000) A scalable approach to balanced, high-dimensional clustering of market-baskets. In: HiPC '00: proceedings of the 7th international conference on high performance computing. Springer, London, pp 525–536
35. Wang K, Xu C, Liu B (1999) Clustering transactions using large items. In: CIKM '99: proceedings of the 8th international conference on information and knowledge management. ACM, New York, pp 483–490
36. Wang P (2008) Clustering and classification techniques for nominal data application. PhD thesis, City University of Hong Kong
37. Wang W, Yang J, Muntz R (1997) STING: a statistical information grid approach to spatial data mining. In: VLDB '97: proceedings of the 23rd international conference on very large data bases. Morgan Kaufmann, San Francisco, pp 186–195
38. Wang W, Yang J, Muntz R (1999) STING+: an approach to active spatial data mining. In: ICDE '99: proceedings of the 15th international conference on data engineering. IEEE, Los Alamitos, pp 116–125
39. Yang Y, Guan X, You J (2002) CLOPE: a fast and effective clustering algorithm for transactional data. In: KDD '02: proceedings of the 8th ACM SIGKDD international conference on knowledge discovery and data mining. ACM, New York, pp 682–687
40. Yarowsky D (1995) Unsupervised word sense disambiguation rivaling supervised methods. In: ACL '95: proceedings of the 33rd annual meeting on association for computational linguistics. Association for Computational Linguistics, Stroudsburg, pp 189–196
41. Yun CH, Chuang KT, Chen MS (2001) An efficient clustering algorithm for market basket data based on small large ratios. In: COMPSAC '01: proceedings of the 25th international computer software and applications conference on invigorating software development. IEEE, Washington, pp 505–510
42. Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: an efficient data clustering method for very large databases. In: SIGMOD '96: proceedings of the 1996 ACM SIGMOD international conference on management of data. ACM, New York, pp 103–114
43. Zhao Y, Song J (2001) GDILC: a grid-based density-isoline clustering algorithm. In: ICII '01: proceedings of the international conferences on info-tech and info-net, vol 3. IEEE, New York, pp 140–145