UNSUPERVISED ANOMALY DETECTION OF HIGH DIMENSIONAL DATA WITH LOW DIMENSIONAL EMBEDDED MANIFOLD

A Dissertation

by

IMTIAZ AHMED

Submitted to the Office of Graduate and Professional Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Yu Ding
Committee Members,	Satish Bukkapatnam
	Xia Hu
	Li Zeng
Interim Head of Department,	Lewis Ntaimo

August 2020

Major Subject: Industrial Engineering

ABSTRACT

Anomaly detection techniques are supposed to identify anomalies from loads of seemingly homogeneous data and being able to do so can lead us to timely, pivotal and actionable decisions, saving us from potential human, financial and informational loss. In anomaly detection, an often encountered situation is the absence of prior knowledge about the nature of anomalies. Such circumstances advocate for 'unsupervised' learning-based anomaly detection techniques. Compared to its 'supervised' counterpart, which possesses the luxury to utilize a labeled training dataset containing both normal and anomalous samples, unsupervised problems are far more difficult. Moreover, high dimensional streaming data from tons of interconnected sensors present in modern day industries makes the task more challenging. To carry out an investigative effort to address these challenges is the overarching theme of this dissertation.

In this dissertation, the fundamental issue of similarity measure among observations, which is a central piece in any anomaly detection technique, is reassessed. Manifold hypotheses suggest the possibility of low dimensional manifold structure embedded in high dimensional data. In the presence of such structured space, traditional similarity measures fail to measure the true intrinsic similarity. In light of this revelation, reevaluating the notion of similarity measure seems more pressing rather than providing incremental improvements over any of the existing techniques.

A graph theoretic similarity measure is proposed to differentiate and thus identify the anomalies from normal observations. Specifically, the minimum spanning tree (MST), a graph-based approach is proposed to approximate the similarities among data points in the presence of high dimensional structured space. It can track the structure of the embedded manifold better than the existing measures and help to distinguish the anomalies from normal observations. This dissertation investigates further three different aspects of the anomaly detection problem and develops three sets of solution approaches with all of them revolving around the newly proposed MST based

similarity measure.

In the first part of the dissertation, a local MST (LoMST) based anomaly detection approach is proposed to detect anomalies using the data in the original space. A two-step procedure is developed to detect both cluster and point anomalies. The next two sets of methods are proposed in the subsequent two parts of the dissertation, for anomaly detection in reduced data space. In the second part of the dissertation, a neighborhood structure assisted version of the nonnegative matrix factorization approach (NS-NMF) is proposed. To detect anomalies, it uses the neighborhood information captured by a sparse MST similarity matrix along with the original attribute information. To meet the industry demands, the online version of both LoMST and NS-NMF is also developed for real-time anomaly detection. In the last part of the dissertation, a graph regularized autoencoder is proposed which uses an MST regularizer in addition to the original loss function and is thus capable of maintaining the local invariance property. All of the approaches proposed in the dissertation are tested on 20 benchmark datasets and one real-life hydropower dataset. When compared with the state-of-the-art approaches, all three approaches produce statistically significant better outcomes.

"Industry 4.0" is a reality now and it calls for anomaly detection techniques capable of processing a large amount of high dimensional data generated in real-time. The proposed MST based similarity measure followed by the individual techniques developed in this dissertation are equipped to tackle each of these issues and provide an effective and reliable real-time anomaly identification platform.

DEDICATION

To My Lovely Wife

Ineen Sultana

To My Dear Parents

Umme Kulsum and Jafar Ahmed

ACKNOWLEDGMENTS

Obtaining a Ph.D. is a long journey. No doubt, it requires an inquisitive mindset, strong determination and passionate pursuit of uncovering new knowledge. However, at the same time, it takes a lot of support, inspiration and dedication to complete one. My Ph.D. journey is no different. In this section, I would like to express my gratitude to all those people who helped me to climb this steep hill.

At first, I would like to thank Professor Yu Ding, my Ph.D. advisor. I consider myself very lucky to have him as my advisor. He takes very good care of his Ph.D. students. To me, his best part is his highly organized, timeliness and attention to detail nature. Whenever he hires a Ph.D. student, he sets a specific plan for the student over the next five years and makes sure that the student is always on track. He spends a big portion of his busy schedule behind his students. Initially, I was progressing very slowly in my research, absolutely sloppy in writing and used to make many small mistakes. He keeps his faith in me and relentlessly tried to correct all of these mistakes by going through each one of them.

Next, I would like to thank my dissertation committee members, Dr. Xia (Ben) Hu from the Department of Computer Science and Engineering. He helped in my research by providing ideas and necessary directions. I would also like to thank Dr. Satish Bukkapatnam and Dr. Li Zeng for their constant support and directions in all phases of my research work.

I would like to thank the industrial collaborators of my research work, Aldo Dagnino, Mithun P. Acharya and Travis Galoppo from ABB Inc. for providing me the industrial angle of the research problem and constant guidance to solve these problems. I would like to mention the former and current members of our research group: Dr. Hoon Hwangbo (now an assistant professor at UTK), Dr. Yanjun Qian (now an assistant professor at VCU), Erika Sy (now at Dow Chemical Company), Dr. Ahmed Aziz Ezzat (now an assistant professor at Rutgers), Shilan Jin, Abhinav Prakash,

Adaiyibo Kio, Jiaxi Xu, David Perez and Jason Lawley. It is always fun to discuss a wide variety of topics with them which includes all the way from research problems to Netflix movies!

I would like to express my gratitude to my parents for all the little sacrifices they made along the way to bring me up and help me to reach this stage of life. I would also like to mention my brother Jamil Ahmed who always keeps me in his prayers and wish me success. I would like to mention my undergraduate and M.Sc. thesis supervisor Dr. Abdullahil Azeem and my former colleague Dr. Sanjoy Kumar Paul for guiding me during the initial phase of my research career.

Lastly, I would like to express my heartfelt gratitude and love to the most important person of my life, my wife, Ineen Sultana. Being a Ph.D. student herself in the same department, she understands me in a way that no one does. Obtaining a Ph.D. is a stressful journey. Very often, you can doubt yourself and become depressed. But it was not that difficult for me as I had the privilege to have the most encouraging and jolly person on my side. She has the power to cheer me up in any situation whether it is academic or non-academic. I wish her the best for her Ph.D. as she is planning to defend her dissertation by the end of this year.

CONTRIBUTORS AND FUNDING SOURCES

This work was supported by a dissertation committee consisting of Professors Yu Ding, Satish Bukkapatnam and Li Zeng of the Department of Industrial & Systems Engineering and Professor Xia (Ben) Hu of the Department of Computer Science & Engineering.

All work conducted for the dissertation was completed by the student independently.

Graduate study was partially supported by the grants from NSF under grant no. CMMI-1545038, IIS-1849085 and ABB project contract no. M1801386.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Anomaly detection: An overview

Anomalies, also referred to as outliers, are loosely defined as data points or a cluster of data points which lie away from the neighboring points or clusters and are inconsistent with the overall pattern of the data. It is difficult to come up with a universal definition of anomalies as it often depends on the context.

Anomalies could be categorized into many different types [1, 2]. They can be *global* and distant from the majority of data points or can be *local* and homogeneously mixed with the regular data points unless properly separated. In Fig. 1.1, point A_1 and A_2 are referred to as the global point anomalies as they are far away from all of the existing clusters (C_1, C_2, and C_3) and data points. Point A_3 is referred to as a local point anomaly. At first look, it seems like a legitimate point as it lies close to the points in data cluster C_1. But, if we magnify the local neighborhood and compare its position only with respect to the points from cluster C_1, point A_3 becomes more visible as an anomaly. Anomalies can also form a cluster. In Fig. 1.1, for example, we can identify cluster C_3 as an anomalous cluster as it is different from the other clusters in size.

There are three broad categories of anomaly detection approaches, depending on the labels of the data in a training set. *Supervised anomaly detection* comes into play when we have appropriately labeled training data in advance (both normal and abnormal) so that we can train a model based on these labeled data and use it to decide the labels of future data. Support Vector Machine (SVM) [3] or Artificial Neural Network (ANN) [4] are the examples of this approach. But, when we have only normal instances and no anomalous data, we can still use the normal data to train a model and classify future observations as anomalies if they deviate from the normalcy. This normal-data-only approach is known as *semi-supervised anomaly detection*. One-class SVM [5]

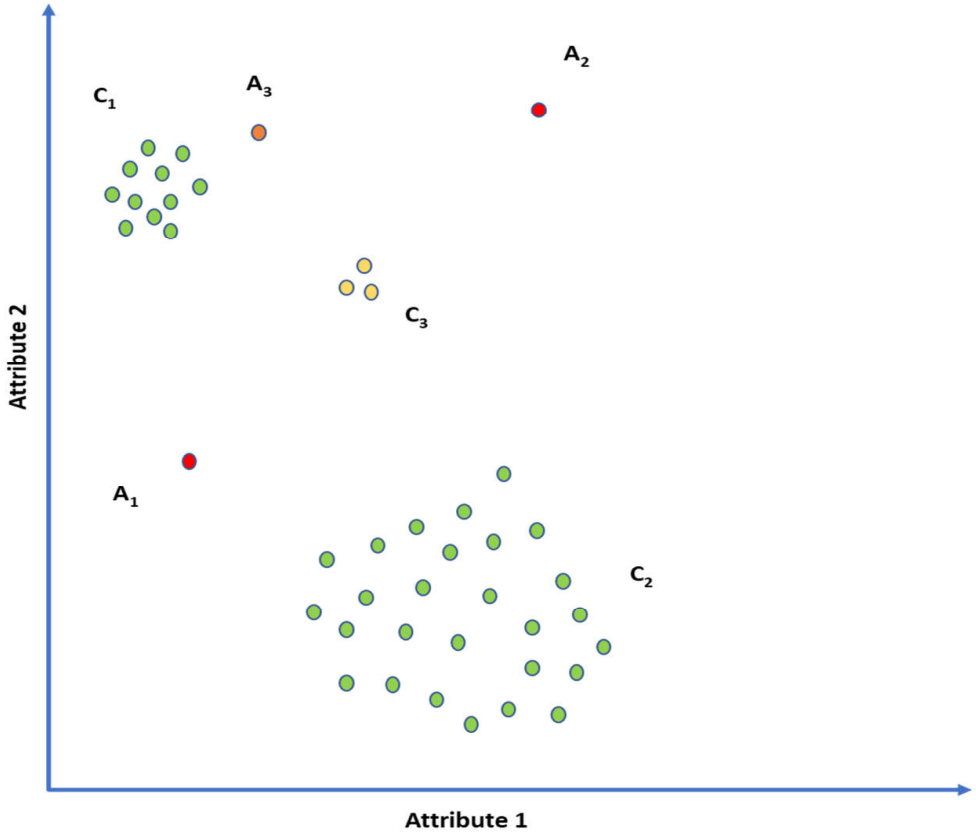


Figure 1.1: Types of anomalies. (Reprinted from [1])

falls under this category. The most difficult scenario is the absence of any label of the data. As a result, it is not possible to conduct a supervised training. One, therefore, has to rely entirely on the structure of the dataset and detect the anomalies, if any, in an unsupervised manner. This last category is known as *unsupervised anomaly detection*. Throughout this dissertation, we are going to work with anomaly detection problems falling into this latest category.

1.2 Motivation

Anomaly detection problem arises in many real-life applications including but not limited to credit card fraud detection, cybersecurity, medical image analysis, surveillance, and industrial process safety. In the current industrial setting, various multi-platform devices, sensors converge into one centralized, interconnected network and lead to the burst in the availability of unlabeled streaming data points. As a natural consequence, it creates the need for developing suitable analytic techniques to analyze these high dimensional data stream in an unsupervised fashion and detect unusual, anomalous behaviors in real-time.

To detect anomalies, using unlabeled data streams pushes us to the territory of unsupervised learning. Consider the example of a hydropower plant which works as a motivation behind this dissertation. It operates with turbine systems that are instrumented with dozens of sensors. Each turbine has subcomponents or functional areas such as several bearing systems, a generator, etc. Sensors collect various types of data in real-time such as the temperature of oil inside the bearing systems, vibrations in each functional area, a variety of harmonics in functional areas, and many more. In total, each turbine collects more than 200 attributes from its sensors. The sensor data is then stored in a central control system and kept as time stamped historical data points. Anomaly can be triggered from various sources and can cause a range of problems.

When a service/maintenance engineer suspects that there is a malfunction in a turbine, she/he extracts a dataset from the control system that contains the collected sensor data for that turbine for the selected period of time (few weeks to few months), and then stores this data in a relational database or simply in a .csv file for further analysis. Staring at a spreadsheet of data, a service/maintenance engineer often wonders if there is an automated, efficient way to distinguish the anomalies in the turbines. This problem falls under unsupervised anomaly detection because the historical dataset in the spreadsheet almost surely has both normal data and anomalies. It is just that the service/maintenance engineers do not know which is what. What makes this problem

more challenging is the number of attributes in the data space, amounting to a few hundreds and making a low-dimensional visualization difficult to carry out. An efficient offline anomaly detection approach can surely help a lot but still not enough to identify anomalies as they appear which is vital to protect the health of components. Equally important is to look at the streaming nature of the data. As data streams arrive in real-time for analysis, examining them one by one manually is not a practical approach. A pressing need is to have an automatic detection technique that could provide a list of potential anomalies by evaluating the stored high dimensional data points in small batches.

1.3 Problem with embedded manifold and a new solution

Now, as we have already laid out the motivation behind our work by illustrating the practical industrial need, let us move into the technical challenges of an unsupervised anomaly detection approach that we need to solve.

A fundamental issue in any anomaly detection approach is deciding on the similarity metric which will be used to distinguish the anomalous observations from the normal ones. Though a variety of approaches have been proposed in all these years, the vast majority of them rely on the Euclidean distance and some of its statistical variants. The concept is pretty straightforward: if $||A - B||_2 > ||A - C||_2$, it implies that A and C are more similar. When a minority of data points are dissimilar from a majority of data points, then the minority of data points are considered anomalies. However, in high dimensional spaces, pairwise Euclidean distances become similar and as a result Euclidean distance-based discriminative methods may not be able to differentiate among the relative position of data points accurately [6].

In a higher dimensional space, data points are also more likely to embed a complicated structure thus leading to a nonlinear manifold. A manifold is a topological space that locally resembles Euclidean space near each point but globally it may not. In the presence of such a complicated space structure, the direct Euclidean distance between two points does not represent their intrinsic

distance as it may no longer be possible to reach directly from one point to another point through a straight line. The later problem is more severe as it can occur even in a low dimension and we want to treat this issue first.

Let us elaborate on the problem of a structured data space and the embedded manifold through an example illustrated in Fig. 1.2. This example is inspired by the work of [7]. When the data forms a nonlinear structure in space, the distance measure between two data points on this structure is then constrained by such structure and should be accordingly construed. Consider the data points E, D, F in Fig. 1.2(a). With the structure, E-to-D is closer than E-to-F, meaning that E and D are more similar to each other than E and F. But if using the Euclidean distance, represented by the dotted lines, EF is shorter than ED, making the learning algorithm to believe E and F more similar to each other than E and D. Note that this example happens in a 3 dimensional space. It means that when the space structure exists, the Euclidean distance can break down even in a relatively low-dimensional space and lead us to the wrong conclusion. A similar real-life example illustrating a structured space is the flight route selection between two places on the earth's surface. It is not the direct Euclidean distance that governs the route selection, but the distance constrained by earth shape; this space-constrained distance is known as geodesic distance. In Fig. 1.2(a), the solid green line highlights a geodesic path between E and F.

As it is arguably more difficult to ascertain a priori whether a data space embeds a structure, we think it is safe to assume the presence of a structured space. This assumption raises a related question concerning the computation of the true intrinsic distance among data points without knowing the shape of the structure beforehand. In this dissertation, we devise a new similarity measure based on the concept of the minimum spanning tree (MST). What motivates us to choose MST as a similarity measure is the fact that as the number of data instances increases, the shortest path distances among data instances provide a good approximation to the geodesic distances.

Before going into further details, let us first look how this MST based similarity concept works. Suppose that one has a connected edge weighted undirected graph $G = (V, E)$, where V denotes

(a) Misrepresentation of true intrinsic distance.

(b) A possible solution using MST.

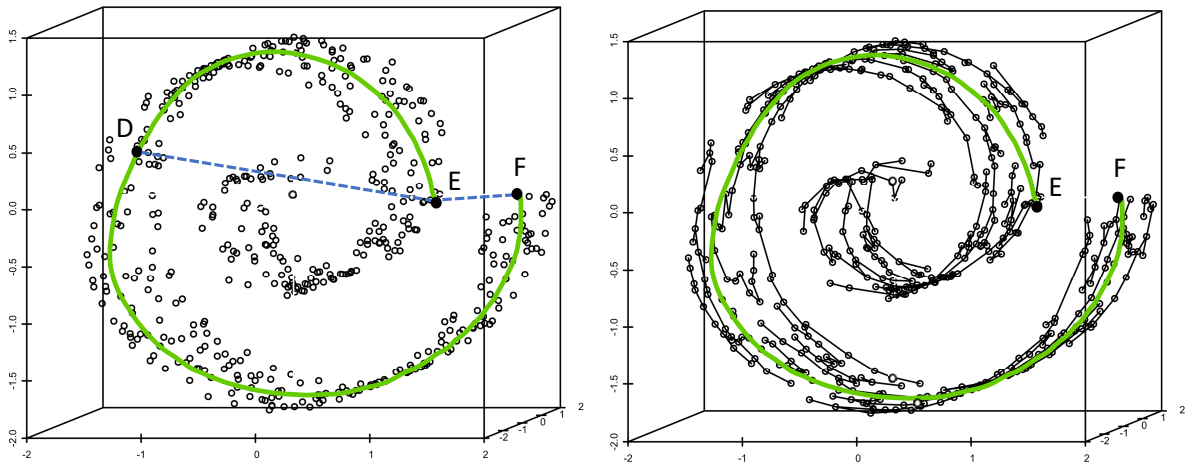


Figure 1.2: Shortcoming of Euclidean distance in a data space embedding complicated structures and a possible solution.

the collection of vertices and E represents the collection of edges with a real valued weight e_{ij} assigned to each of them, where i, j represent a pair of vertices from V. A minimum spanning tree is a subset of the edges in E of graph G that connects all the vertices in V, without any cycles and, with the minimum possible total edge weight. In other words, it is a spanning tree whose sum of edge weights is as small as possible. Let us define $\mathbf{D}_{ij}$ as the distance (Euclidean) between vertex i and vertex j connected by an edge. The weight, however, does not always mean physical distances. For example, the weight could represent the amount of flow between a pair of vertices or sometimes the cost of constructing this edge.

Consider a simple example in Fig. 1.3, left panel, where there are 10 vertices and 16 edges. Each of the edges has a unique edge length, which is represented by a numeric value. If we want to connect all the nodes using a subset of the given edges without forming a cycle, there could be many such combinations, but the one(s) having the minimum total edge length is the MST. MST may not be unique, but for this example, it is unique and shown in the right panel of Fig. 1.3. The edges in black color represent the selected nine edges from the 16 in total, forming the MST.

We see that MST compresses the original graph by reducing the total energy, and thereby,

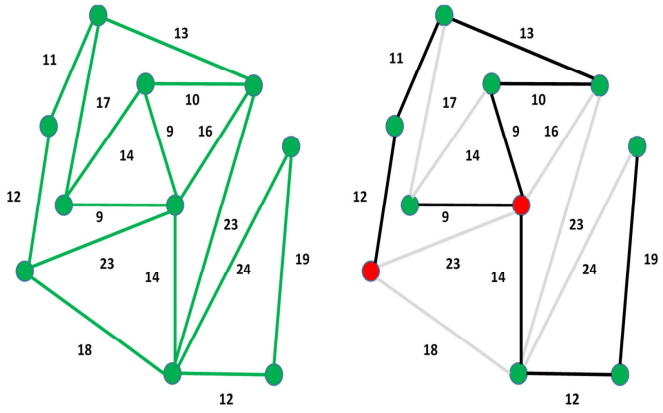


Figure 1.3: Formation of an MST: the left panel is the initial graph. In the right panel, the dark black edges form the minimum spanning tree. The total MST weight is 109.

provide a new measure of distance between the vertices. Although the distance between a pair of immediately connected nodes is still Euclidean, the distance between a general pair of nodes (i.e., data points) is not. Rather, it is the summation of many small-step, localized Euclidean distances hopping from one data point to another point. For example, the new distance between the two colored vertices is $12 + 11 + 13 + 10 + 9 = 55$ (right panel), while their original distance in the left panel is 23. We store in $\mathbf{M}_{ij}$ the new pairwise distance of vertices in the MST for future use instead of the typical Euclidean distances, $\mathbf{D}_{ij}$. As the MST reflects the connectedness among data points in a nonlinear manifold, the MST-based distance is the geodesic distance between two data points, which, according to [7], provides a better metric to differentiate them.

MST [8, 9] has the capability of approximating the geodesic distance in the presence of nonlinear manifold. To achieve that it only uses the knowledge of neighboring points, nothing more than that. MST can be applied to any dataset after the data points are represented by a graph object. We can do so by considering each observation as a vertex and the pairwise Euclidean distances among the vertices as the edge weights. Please refer to Fig. 1.2(b), where MST track the

geodesic path to provide an approximation of geodesic distance between E and F. There are three major algorithms [10, 11, 12] that can construct an MST for a given graph. The computational cost of constructing an MST is $O(n \log m)$, where n is the number of edges in the graph and m is the number of vertices. Here, n is $\binom{m}{2}$ in our applications (one edge for every pair of vertices, i.e., a fully connected graph), which suggest a time complexity of $O(m^2)$ in constructing an MST, and because of this, an MST can be constructed efficiently even for a large dataset.

1.4 Related works

Anomaly detection methods in the literature can be categorized into some major domains depending on the main theme of their detection mechanisms. The detection can be carried out both in the original data space and in a latent space or subspace. The number of variables in latent space or subspace is generally lower than the original data space. Whatever route we choose to follow, there are pros and cons associated with each of them which plays a significant role in the accuracy of the final detection outcomes. So, instead of grouping them based on their detection philosophy, we want to view the anomaly detection approaches from these two different perspectives as in our work we decide to contribute in both of these scenarios. For a more clear understanding of the flow of anomaly detection research, we summarize the major approaches accordingly in Fig. 1.4.

1.4.1 Anomaly detection in original data space

The commonality among the methods highlighted in this subsection is that they all detect anomalies using the original data points without transforming them into latent space or projecting into subspaces.

1.4.1.1 Distance-based methods

The first variant is the distance-based methods which consider a point as an anomaly if it lies further away from the majority of the data points [13]. The distance-based criterion entails a number of

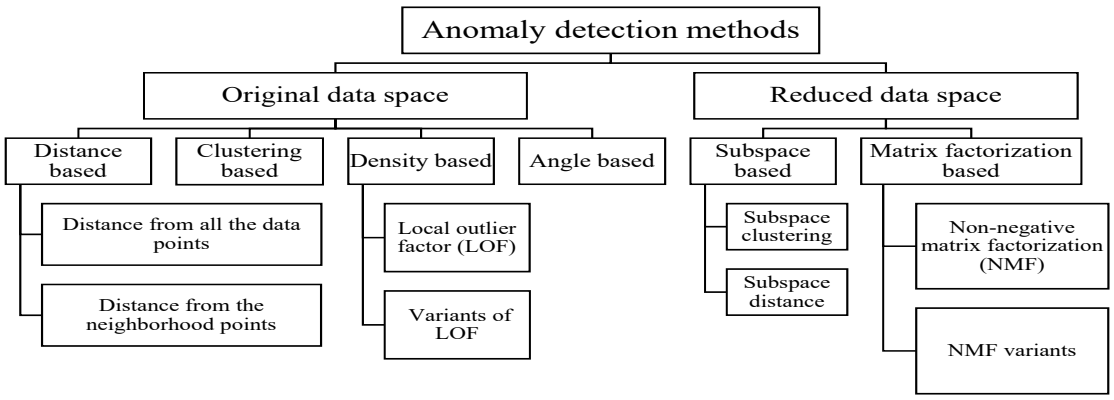


Figure 1.4: Major categorizations of anomaly detection methods.

variants to handle the complexity in real life. For instance, the k-nearest neighbor (k-NN) based methods compare a candidate data point with its k nearest neighbors, rather than all the data points, because it was believed doing so enhances the detection capability [14, 15]. Anomaly scores are also defined as the ratio of the average of the distances from the test point to its k-NN's and the average pairwise distance within the k-NN set [16]. One of the major downsides of these distance-based methods is: if the dataset has multiple clusters of varying density then they would not be able to separate local anomalies (i.e., anomalies only with respect to a single cluster) successfully.

1.4.1.2 Density-based methods

- **Local outlier factor (LOF)**: The density-based approaches consider a point as an anomaly if the density around it is considerably lower than the density around its neighbors. The first of its kind and which is also by far the most popular method as well as the most cited work in the anomaly detection literature is known as local outlier factor (LOF) [17]. It is based on an idea called *local density* which is estimated by the distance at which a point can be reached by its neighbors. By comparing the local density of an object to the local densities

of its neighbors, one can identify the anomalies. A drawback of this popular method is that it works pretty well in the presence of the spherical shaped cluster but fails to identify the irregular geometric shaped clusters especially when linearly correlated points form a cluster. Connectivity based anomaly factor (COF) [18] claimed to solve this problem by introducing a shortest path based chaining approach. The main argument behind COF is that 'Low density does not always imply the anomalousness but the low connectivity to neighboring points surely does'. It differs from LOF by proposing a new incremental way of calculating the nearest neighbors.

- **Variants of LOF**: Due to the popularity of LOF method, variants of the LOF method was introduced within a short span of time to address the problems that could potentially limit the performance of the original LOF. Influenced outlierness (INFLO) [19] was proposed to compare the local density of an object to the average of its influence space. It is based on LOF, but it expands the neighborhood of the object to the influence space of the object and considers both the original neighborhood and the reverse neighborhood of an object. INFLO was introduced in order to handle the case where clusters with varying densities are in close proximity. Local density factor (LDF) [20] computes a kernel density estimation known as local density estimate (LDE), over a user-given number of k nearest neighbors. The LDF score is the comparison of LDE for an observation to its neighboring observations. Local outlier probabilities (LoOP) [21], was introduced which computes a local density based on the probabilistic set distance for observations, with a user given k nearest neighbors. The density is compared to the density of the respective nearest neighbors, resulting in the local outlier probability. The value ranges from 0 to 1, with 1 representing the greatest outlierness. The advantage of this probability based metric is that now anomaly scores are more interpretable and can be compared over datasets.

No matter what approach these methods adopt to calculate the local density, they all suffer from the problem of selecting a suitable value for k (number of nearest neighbors) like all other k-

NN based models. As we are working in an unsupervised setting it would not be possible to apply method like cross-validation to estimate a suitable k. In the original paper introducing LOF, it was suggested to use a range of k values and then for each observation select the k value that results a maximum anomaly score. In the literature when comparing methods, it is a common practice to use either the best k or the average result over a range of k values. However, in Section 2, we propose an adhoc policy for selecting the neighborhood size k.

1.4.1.3 Clustering-based methods

The density-based methods provide a different perspective for identifying anomalies, which is to consider the data clustering tendency. There are clustering approaches which are specifically developed for anomaly detection. There are three categories of clustering-based anomaly detection algorithms. Methods in [22, 23] fall under the first category, which identifies instances that do not belong to any regular cluster as anomalies. The second group of clustering technique is a variation of the first group and uses a clustering algorithm to detect clusters and then calculate an anomaly score by taking the distance from a point to its nearest cluster center. Both of these groups do not take into account that anomalies can also form clusters and in those cases such methods will fail to detect these anomalous clusters. The third category of clustering based algorithms [24, 25] was introduced to tackle this problem which assumes that normal observations belong to large and dense clusters, whereas anomalous observations belong to small and sparse clusters or lie further away from the cluster centroid.

1.4.1.4 Angle-based methods

Angle-based methods were introduced with the consideration that angles are a more stable measure in high dimensions compared to distances [26]. One major limitation of this method is the high computational time it requires to calculate the angles.

All of the methods discussed in this subsection have different ways of detecting anomalies,

but they are all similar in at least one aspect; which is the use of the Euclidean distance. We already argued that the use of Euclidean distance-based metric loses its effectiveness in a data space embedding inherent structures, which is most likely of high dimensionality. So, no matter what method we select to modify and develop further or even propose a new method, as long as we are using the Euclidean distance, we cannot capture the structure of the data and our detection would be wrong.

We believe that instead of making incremental improvements over any one of the existing methods, we need to rethink this fundamental issue of how we differentiate data instances in unsupervised learning settings. The current reliance on Euclidean distances appears to run out of steam. In Section 2, we develop a local MST (LoMST) based anomaly detection approach, capable of detecting anomalies by comparing the connectivity of each point to its neighbors.

1.4.2 Anomaly detection in reduced data space

As more and more high dimensional data comes into play, people start to realize the curse of dimensionality and how it reduces the effectiveness of the Euclidean metric. Consequently, the research effort in anomaly detection shifted towards evaluating data points in reduced dimensional space.

1.4.2.1 Subspace based methods

The key argument in favor of *subspace based methods* is that not all variables are relevant for discovering anomalies and it would be wiser to consider relevant subspaces rather than considering the entire feature set. One could devise different strategies in selecting the subspaces. For instance, principal components analysis (PCA) [27] renders the subspace that has the largest variances most relevant, while multidimensional scaling (MDS) [28] selects the subspace that preserves the interpoint distances in their low dimensional representation. They are both capable of preserving the original data space structure in linear vector spaces, but they tend to lose the data structure in

the presence of nonlinear manifolds [7].

A grid-based subspace clustering [29] was proposed for searching sparse, rather than dense, grid cells to report objects contained within those sparse grid cells as anomalies. High-dimensional outlying space (HOS) miner [30] ranks a point as an anomaly in any subspace if the sum of its distance from its k-nearest neighbors crosses a predetermined threshold. Subspace clustering [31] can be also helpful as anomalies are found in abnormally few clusters or low dimensional clusters. Subspace outlying degree (SOD) [32] detects an anomaly based on its deviation to the subspace spanned by a set of reference points. The subspace is formed with those features where the reference points vary very little. But the success of this method depends on the tuning of neighborhood parameters just like the k-NN method. HicS (High contrast Subspaces) [33] relies on finding those subspaces where attributes are correlated (statistically dependent). GLOSS [34] suggested that global neighborhoods should be considered when detecting anomalies locally in selected subspaces.

Subspace methods undoubtedly made progress in the unsupervised anomaly detection literature. But fundamentally, finding out the right subspaces to explore and how to compare anomalies from different subspaces are still a difficult problem to solve. However, autoencoder [35, 36, 37] came out as a rescue to solve the problem of finding a proper low dimensional subspace. It does so by reconstructing the data again from the subspace by minimizing the difference between reconstructed and original data. So, now, the reduced data space coordinates can be utilized for unsupervised learning. But, again, it suffers from the problem of embedded manifold due to the use of Euclidean distance-based similarity measure. Therefore, in Section 4, we propose a new graph regularized autoencoder which can overcome the problem of a complicated manifold and thereby generate a low dimensional representation useful for anomaly detection.

1.4.2.2 Matrix factorization based methods

- **NMF for anomaly detection**: Matrix factorization based methods have been applied successfully for a long time to represent the data in a reduced dimensional space for better learning and hence achieve the clustering benefits. These methods essentially factorize the original data matrix into two low rank matrices. One helps us to obtain the mapping of original data points to latent space and another depicts how the latent space is created from the original attributes. These low rank matrices help us to obtain clustering assignments and the attribute distribution of the clusters. However, these approaches have not gained sufficient popularity in the anomaly detection community. This may be due to the fact that traditional matrix factorization like the principal component analysis (PCA) produces low rank matrices consisting of negative values and positive and negative weights, which tend to cancel each other in reconstructing the original matrix and hence provide no intuitive meaning, while on the other hand, one would like to know the 'meaningful' attribute distribution of these clusters for anomaly detection, so as to evaluate the suspicious observations against them.

  For this reason, the nonnegative matrix factorization [38, NMF] method, which imposes the non-negativity constraint in matrix factorization and only allows additive linear combinations of components, comes across as a better candidate for the purpose of anomaly detection. NMF has the capability of generating both clustering assignments and meaningful attribute distribution in two separate matrices. Immediately after its introduction, NMF not only becomes a powerful tool for clustering [39], but it also shows enough potential as an anomaly detection approach [40, 41]. In the presence of complicated manifolds, however, researchers notice that NMF starts to lose its efficiency [42, 43] as it only tries to approximate the data without trying to mimic the similarity among observations in the latent space. In other words, the shortcoming of the original NMF in anomaly detection is attributed to that it has no provision to include the neighborhood structure information during the calculation of the factored matrices. So, we once again revisit the fundamental problem related to

capturing the inherent structure of the dataset.

- **Variants of NMF**: Two recent versions of NMF tried to tackle this problem. One of them is the graph regularized NMF [43, GNMF], which regularizes the original NMF formulation using a Laplacian matrix. But GNMF still constructs the neighborhood similarity matrix based on simple Euclidean distances. The second NMF variant is the symmetric NMF [42, SNMF], which uses only the similarity information while excluding the attribute information to generate the low rank matrices. In the absence of the attribute information, SNMF depends on a dense pairwise similarity measure which leads to a computational disadvantage. Also, by abandoning the original attribute information in its formulation, SNMF makes its detection outcomes less interpretable or 'meaningful' than NMF or GNMF. So, capturing and preserving the neighborhood structure in a computationally efficient way for meaningful anomaly detection in latent space is still an open problem to solve.

In Section 3, we propose a new neighborhood structure assisted NMF formulation for anomaly detection which takes into account the importance of neighborhood connectivity into account during factorization. To capture the neighborhood similarity, we make use of MST instead of the Euclidean distance-based similarity and thereby attain the advantage of geodesic similarity.

1.5 Performance metric

To evaluate the performance of the competing approaches in benchmark datasets, we use a *precision at* N[44, $P@N$] criteria. This is a widely used performance metric in anomaly detection. When observations are sorted in descending order according to their anomaly scores, the $P@N$ identifies the proportion of the correct anomalies in the top N ranks. In the circumstance where the true number of anomalies, $|O|$ is known, like in the benchmark datasets to be described in Section 1.6, we use that value as our choice of N and treat it as the same cut-off for all methods in comparison. When N is the number of true anomalies, the number of false alarms is implied once $P@N$ is calculated, which is $N - N \times P@N$. That is why we only present $P@N$ in the

benchmark studies. In reality, when the number of true anomalies is not known, the main objective in anomaly detection is still to increase $P@N$ for a fixed N, i.e., to have a higher detection rate within the cut-off threshold.

For a dataset DB of size m, consisting of anomaly set $O \subset DB$ and normal datasets $I \subseteq DB$, such that $DB = O \cup I$, $P@N$ can be formulated as

$$P@N = \frac{\#\{o \in O \,|\, rank(o) \leq N\}}{N}, \quad \text{where} \quad N = |O|. \tag{1.1}$$

When N is unknown in practical settings, a good practice is to choose N larger than the estimated number of anomalies but small enough so that it makes follow up identification operations feasible. The rationale behind this choice lies in the fact that the false positive rate for anomaly detection problems is generally high, especially compared to the standard used for supervised learning methods. Despite a relatively high proportion of false positives, anomaly detection methods can still be useful, particularly used as a pre-screening tool. By narrowing down the candidate anomalies, it helps human experts a great deal to follow up with each circumstance and decide how to take proper action or deploy a countermeasure.

1.6 Benchmark datasets

In this study, we use 20 benchmark anomaly detection datasets from [44] for the purpose of performance comparison. In Table 1.1, we summarize the basic characteristics of these 20 datasets. For all of these benchmark datasets, we know the total number of anomalies and the label of the individual observation, i.e., whether it is an anomaly or not. There are several versions of these datasets available depending on the data cleaning and preprocessing steps involved. For our analysis, we choose to use the normalized version of the datasets with all missing values removed and categorical variables converted into a numerical format.

Table 1.1: Public benchmark datasets used in the performance evaluation study.

Dataset	Number of observations (m)	Number of anomalies ($\|O\|$)	Number of attributes (p)
Annthyroid	7,200	347	21
Arrhythmia	450	12	259
Cardiotocography	2,126	86	21
HeartDisease	270	7	13
PageBlocks	5,473	99	10
Parkinson	195	5	22
Pima	768	26	8
SpamBase	4,601	280	57
Stamps	340	16	9
WBC	454	10	9
Waveform	3,443	100	21
WPBC	198	47	33
WDBC	367	10	30
ALOI	50,000	1,508	27
KDDCup99	60,632	200	41
Shuttle	1,013	13	9
Ionosphere	351	126	32
Glass	214	9	7
Pendigits	9,868	20	16
Lymphography	148	6	19

1.7 Hydropower dataset

Apart from the public benchmark datasets, a real-life dataset generated from a hydropower plant is also used in the performance comparison study. The hydropower data that was initially received was time-stamped (a total of seven months worth of data) and divided into different functional areas (turbines, generators, bearings etc.). The data was collected at 10-minute interval each day. But it was not always continuous and some days from each of these seven months were missing. After combining all data across all functional areas, there are 9,508 observations (rows in a data table) and 222 attribute variables (columns in a data table). Each row has a time stamp assigned to it. Attribute variables are primarily temperatures, vibrations, pressure, harmonic values, active power etc.

We conducted some basic preprocessing and statistical analysis [45] in order to clean the data. To maintain the similarity with the 20 benchmark datasets, we normalized the data, removed the duplicate rows as well as the rows with missing values. Additionally, we also did correlation analysis and plotted histogram, density and box plots. The data preprocessing did yield a small number of data records that are so far off from other data records. When confirming with the domain expert who provided the data, it was confirmed that those records were due to a recording mistake. After removing them, the total number of observations comes down to 9,219.

Here, different from the benchmark datasets, we no longer have the information of the number of true anomalies. After consulting the operating manager who provided this dataset, we are advised to report top 100 anomalous time stamps to the manager. The manager would follow up and check the status of the operation in detail, for instance, by manually examining operational logs and physical conditions of components, and then confirm us about the validity of the findings.

1.8 Organization of this dissertation

The dissertation proposes three unsupervised anomaly detection algorithms all revolving around our newly developed MST based dissimilarity metric to capture the geodesic similarity among observations. These algorithms have the potential to tackle the problem of high dimensional data with manifold embedding property better than the simple Euclidean distance. In addition, the dissertation also proposes the online version of these algorithms to handle the streaming data and enable real-time detections. All of these algorithms are tested on 20 public benchmark datasets and the real-life hydropower dataset to prove their worth.

The rest of the dissertation is organized as following. Section 2 develops a local MST (LoMST) based anomaly detection algorithm which detects both local and global anomalies by adopting a two step procedure. When compared with a wide variety of neighborhood based anomaly detection algorithms, it achieves superior and significantly better detection capability. Section 3 develops a neighborhood structure assisted NMF (NS-NMF) method to prove the fact that neigh-

borhood structure can essentially help the matrix factorization principles to achieve better anomaly detection capability and MST based similarity measure is an ideal candidate for such neighborhood approximation. It proposes a new NMF formulation by providing a detailed theoretical and empirical comparisons with some of the existing state of art formulations. Section 4 argues for the necessity of dimensionality reduction before applying any anomaly detection algorithm and the importance of autoencoder mechanism to do it in an unsupervised way. Then a graph regularized autoencoder is developed to deal with the nonlinear embedding problem which can be extended into today's powerful deep learning frameworks and thus applicable to data of any size and type. Finally, Section 5 summarizes the key contributions and highlights potential extensions beyond this dissertation.

2. LOMST: A LOCAL MST BASED ANOMALY DETECTION APPROACH AND ITS APPLICATION TO HYDROPOWER TURBINES*

In this chapter, we use the concept of MST on a local neighborhood level and develop a local MST (LoMST) based anomaly detection method. The merit of this method is that it provides a new distance measure among the observations in the local neighborhood which is capable of capturing the relative connectedness of data points/clusters in a complicated manifold. We also propose a strategy to detect global anomalies as a preprocessing step before detecting local point anomalies. The proposed method is compared with 13 popular anomaly detection methods on 20 benchmark datasets, demonstrating a considerable improvement in its ability of identifying anomalies. Furthermore, it is also applied to the dataset generated from a hydropower turbine and demonstrates remarkable detection competence.

2.1 Introduction

Our problem is motivated by the anomaly detection problem encountered in a hydropower generation plant as discussed in Section 1.7. The problem demands for an anomaly detection approach that can deal with high dimensional data in a streaming fashion. It was argued in [7] that the higher a data space's dimension, the more likely it embeds an inherent structure forming a nonlinear manifold. As a result, geodesic distance must be used instead of Euclidean distances in a nonlinear manifold to reflect accurately the distance between data points. We want to note that a complicated structure could happen to low dimensional spaces, too, so that the Euclidean-based distance metric could lose its effectiveness in low dimensional spaces as well. Empirically, however, people ob-

*Reprinted with permission from "Unsupervised Anomaly Detection Based on Minimum Spanning Tree Approximated Distance Measures and its Application to Hydropower Turbines" by Imtiaz Ahmed, 2019. IEEE Transactions on Automation Science and Engineering, 16 (2), 654-667, Copyright 2019 by IEEE.

*Reprinted with permission from "O-LoMST: An Online Anomaly Detection Approach And Its Application In A Hydropower Generation Plant" by Imtiaz Ahmed, 2019. 2019 IEEE 15th International Conference on Automation Science and Engineering (CASE), 762-767, Copyright 2019 by IEEE.

serve such instances happening more often in higher dimensional spaces and tend to associate the loss of effectiveness in Euclidean measures with the high dimensionality [46].

Geodesic distance is the minimum possible distance between two points in a curved surface like the surface of the earth. It was also shown [7] that as the number of data instances increases, the shortest path distances among data instances provide the best approximation to the geodesic distances. Driven by this insight, what we propose here is to use a minimum spanning tree (MST) to provide an approximation of geodesic distances in a structured space and then use it as the (dis)similarity metric. More specifically, we model the data observations as a network of nodes where edges represent the Euclidean distance from one another. An anomalous node would be the one which is less connected to its neighboring nodes. An MST is a measure that can capture the relative connectedness among nodes, while at the same time, approximates the geodesic dissimilarities among observations forming a nonlinear manifold. It has been shown in the literature [47, 9, 8] that MST is indeed a capable approximation of geodesic distances in a high dimensional data space embedding complicated structures.

There are several positive aspects associated with the proposed approach. First, the distance between any two nodes in an MST is no longer the direct Euclidean distance between them; rather it is the new dissimilarity metric which takes into account the overall connectivity among data points reflecting the complexity in a structured data space. So, the MST-based dissimilarity measure is a good candidate to approximate the geodesic distance and has the potential to overcome the limitation of direct Euclidean distance under such circumstances. Next, to take into account the presence of clusters of different shapes and densities, we develop MST locally and compare a node's connectedness with its neighboring cluster only. Doing so enhances the detection ability of the local, point-wise anomalies.

We are aware that MST has been used to find anomalies [48, 49, 50, 51, 52]. The objective of most of them [48, 49, 50, 52] is to isolate the clustered anomalies by removing the links of the global MST one by one. Our attention in this chapter is more about local, point-wise anomalies.

What was done in [48, 49, 50, 52] is similar to Stage 1 of the proposed method (more details later), which is really a preprocessing step in our method. Our main attention is on Stage 2, the local anomaly detection. The concept of [51], is a bit different as the author proposed to label any new test observation as an anomaly by comparing them to the most concentrated subset of points in the training sample. To find out the most concentrated subsets the author chooses to apply the k-point MST. So, this approach is applicable for semi supervised anomaly detection/novelty detection scenario where we know beforehand that the training sample is free from anomalies and can utilize it to detect any abnormality in the test sample. This is fundamentally different from the unsupervised problem we are dealing with. In our case we do not have any information about the label of the training data. Apart from that our approach utilizes neighborhood-based local MST which is entirely different from the concept of k-point MST.

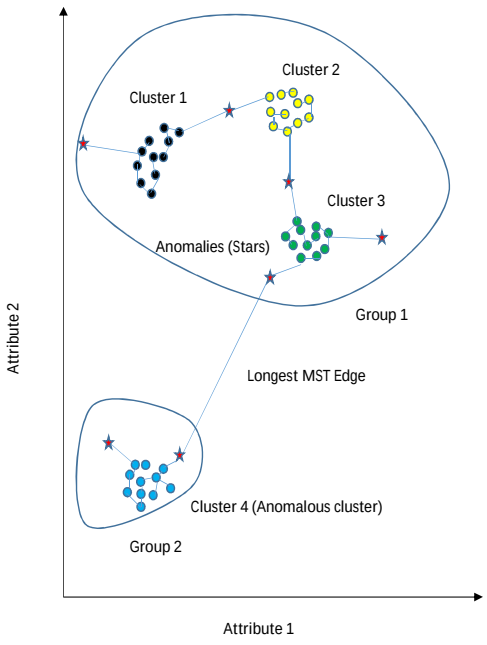
The rest of the chapter unfolds as follows: Section 2.2 describes the main idea of our proposed approach and the steps of the algorithm developed. Section 2.3 presents the comparative performance of our method with respect to other prevalent methods on 20 benchmark datasets. Section 2.4 describes the online version of the LoMST approach for streaming data. Section 2.5 analyzes the performance of the offline and online algorithms in hydropower dataset. Finally, we conclude the chapter in Section 2.6.

2.2 MST-based anomaly detection method

As mentioned earlier, our main focus is to come up with a minimum spanning tree-based distance metric which reflects the dissimilarity among anomalies and normal data points in structured data spaces. Please refer to 1.3 for a brief discussion of the MST's role in manifold approximation. Using MST helps address another complexity often encountered in anomaly detection, which is to detect pointwise anomalies in the presence of anomalous clusters. Fig. 2.1(a) presents an illustrating example in which there are well separated four clusters whose structures are not difficult to identify. The star shaped symbols represent the local anomalies relative to their nearest cluster.

Cluster 4 itself is an anomalous cluster whereas cluster 1, 2 and 3 are regular clusters. Existing anomaly detection methods such as LOF [17] adjust their view field on anomalies by setting different k's, the value of the nearest neighbors: when a small k is used, the local anomalies are detected but the anomalous cluster 4 will be unidentified, while when a large k is used, all the instances can be separated into two parts, namely, Group 1 and Group 2, in which Group 2 contains cluster 4. Under that circumstance, one has to pay the price of not detecting the local anomalies in Group 1. The reason that using MST can help is because MST can be used as a clustering tool [50, 48] to isolate the anomalous clusters first. Then, it can be refined to define the dissimilarity distances in a local setting. This thought points to a two-stage procedure, which is to remove the global anomalous clusters first and then detect the local point-wise anomalies later; both stages use MST as the common methodological foundation.

(a) Point-wise anomalies versus anomalous clusters.



(b) Local MST and LoMST score. The total edge weight of the local MST for $N0$ is its LoMST score, i.e., W_{N0} =E_{01} + E_{12} + E_{23} + E_{04} +E_{45} +E_{36}.

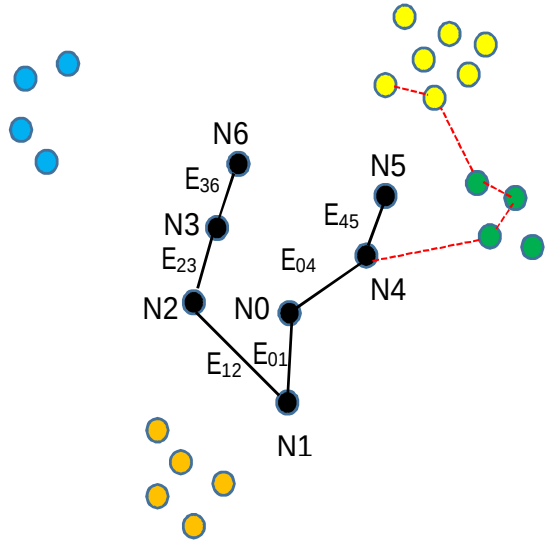


Figure 2.1: Anomaly detection using the proposed framework

Specifically, our detection algorithm proceeds as follows. The first stage is to identify the

anomalous clusters if they exist. The procedure is similar to the existing work of how MST is used [48, 49, 50]. First, we build a global MST using all the data points. The specific MST construction algorithm we use is in [10]. The reason we choose the algorithm in [10] is because it handles dense graphs well, while the other two algorithms [11, 12] handle sparse graphs better. And our case is a dense graph.

After the formation of the global MST, we then look for a long edge and treat it as the connecting edge between the anomalous cluster and the rest of the MST. Once this edge is disconnected, it separates the MST into two groups, and the smaller group is considered an anomalous cluster. One of the problems of deleting such edges is, in the absence of distant anomalies, normal points could be isolated and tagged as anomalies instead. So, we have to create a lower bound so that we can only delete the longest edge that crosses the bound. We decide to use μ+q*σ as the threshold where μ represents the average of the edge weight and σ represents their standard deviation. Now, the choice of q entirely depends on the data set and the distribution of anomalies. If we use a high q, then there will be no candidate edges to be deleted, on the other hand, too small a q will lead to a large chunk of points detected as anomalies. As we are not sure about the structure of the data set beforehand, it is not easy to specify the best value for q. In our case, we evaluate several candidate q values from 2-5 and we found that q =3 works reasonably well for most of the data sets.

This edge deletion procedure will be then iterated on the larger remaining group and see if there is another, less dominating anomalous cluster, until there is no anomalous long edge detected. This procedure is equivalent to a Phase I analysis in statistical quality control [53]. We have summarized the q selection policy for the *Arrhythmia* data set in Table 2.1 as an example.

Table 2.1: q selection policy for the *Arrhythmia* data set

Possible options	Number of iterations required	Number of true anomalies detected	Number of false detection
Mean + 3SD (q=3)	2	3	2
Mean + 4SD (q=4)	1	0	0
Mean + 5SD (q=5)	1	0	0

Once the clustering decomposition stops in the first stage, then we move on to the second stage of identifying pointwise anomalies which is also the main contribution of our work. In the second stage, we go into the neighborhood level for each data point to determine its possibility as an anomaly. The neighborhood is determined by the number of nearest neighbors and parameterized by k. We will come back later to discuss the procedure of selecting the value of k but for now let us assume we have a predetermined value for k.

Denote by R the rest of the data points after the anomalous clusters are removed in Stage 1. For any given data point in R, first, isolate its k nearest neighbors and treat them as this data point's neighborhood. Then, build an MST in this neighborhood. Considering the nature of these neighborhood MSTs, they are referred to as local MSTs (LoMST). The total edge length of the LoMST associated with the original data point is called the LoMST score for this data point and is considered the metric measuring its connectedness with the rest of the points in the neighborhood as well as how far away it is from its neighbors. For this reason, the LoMST is used as the differentiating metric to signal the possibility that the said data point may be an anomaly.

Consider the illustrating example in Fig. 2.1(b). Suppose that we have chosen $k = 6$ and start with data point $N0$. Then, we can locate its neighbors as $N1$, $N2$, $N3$, $N4$, $N5$ and $N6$. The MST construction algorithm connects $N0$ to its neighbors in the way as shown in Fig. 2.1(b). For $N0$, the total edge weight is W_{N0} =E_{01} + E_{12} + E_{23} + E_{04} +E_{45} +E_{36}, which is deemed the LoMST score for $N0$. This procedure will be repeated for other data points and Fig. 2.1(b) shows another MST, which is for $N5$ in the dotted edges.

The LoMST score for a selected observation will be compared with its neighbor's score. Comparison can be done in two ways. We can either compare W with the mean of the neighbor's scores or with the mean-to-standard deviation ratio of the neighbor's scores. Our analysis suggests that both comparison approaches have their own advantages depending on the structure of the dataset. When there are numerous anomalies, almost forming an anomalous cluster within a neighborhood, it would be better to use the mean-to-standard deviation ratio, as the mean of the neighbor's

LoMST scores are severely contaminated by other anomalies. But when the anomalies are very few, point-wise scattered around, the mean of the LoMST score works just fine. Considering that we have the first stage to remove some of the anomalous clusters, in this second stage, we then use the mean comparison of LoMST scores as our default approach.

The step of comparison will be repeated $|R|$ times covering all the nodes in R. Then the comparison score will be scaled between 0 and 1 using the maximum and minimum value of the scores. From now on, we call the normalized scores as the LoMST scores. After that, these LoMST scores will be sorted in a decreasing sequence, where a greater score implies a higher possibility to be an anomaly. To compile a complete list of anomalies, we follow the $P@N$ approach as described in Section 1.6, which is to select a prescribed cut off value N and flag the top N instances on our tank list as anomalies. One main reason behind such a detection procedure is that unsupervised detection methods tend to have a lower detection capability and higher false alarm rate, as compared to general supervised learning algorithms. As a result, unsupervised detection methods are typically used as a screening tool, flagging potential anomalies to be further analyzed by either a human operator or some more expensive procedure. A cut-off is therefore used to ensure the subsequent, more expensive or time consuming step practical and feasible.

For better technical understanding, the algorithm steps are summarized below. In addition, a flowchart of the proposed method separated into two stages is also highlighted in Fig. 2.2.

Now, to select an appropriate value for k, we adopt an approach based on the following observations, illustrated in Fig. 2.3. When we plot the average LoMST scores for a broad range of k (here 1-100), we observe that at small k values, the average LoMST score tends to fluctuate, but as we keep increasing k, the average LoMST score will become stable at a certain point. Our understanding is that when a proper k is chosen, the structure of the data is revealed and the label of the instances will become almost fixed, thus reflected in a less fluctuating LoMST score. If one keeps increasing k, there is the possibility that the data structure becomes mismatch with the assigned number of clusters and the current assignments of anomalies and normal instances become

ALGORITHM 1: LoMST algorithm for anomaly detection

Input : Dataset (rows represent observations and columns represent attributes), number of nearest neighbors, k and cut-off level for identifying anomalies, N.

Output: Anomaly index set, TO

Develop set of vertices V, where each vertex represent a separate observation from the dataset;
Construct edges by calculating Euclidean distance between each pair of vertices and store them in E;
Construct a global MST using V and E, let S be the set of edges of the resulting MST, where $S \subseteq E$;
Calculate the mean, μ, and the standard deviation, σ of the edges in S;
Initialize R, O_1, O_2, T, Nr and Lo = $\oslash$;
Calculate the longest edge from S and store its length in Lo;
if $Lo \geq \mu$ + 3σ **then**
    Remove this edge from the MST tree formed by edges in S;
    From the two disconnected trees, let A= {vertices contained in the smaller tree} and B= {vertices contained in the larger tree};
    Set S=B and O_1=O_1 U A;
    Go to step 6 ;
else
    Set R=S;
    Go to step 16 ;
end
for *each vertex* $r_i \in R$ **do**
    Determine its k nearest neighbors and save them in Nr_i;
    Construct a local MST using edges contained in E_{uv}, where $u, v \in N_i$ and $E_{uv} \subseteq E$;
end
for *each vertex* $r_i \in R$ **do**
    Calculate the total length of r_i's LoMST, W_{r_i};
    Calculate the mean ($\overline{W}_{Nr_i}$) of the total length of the LoMSTs associated with all vertices in Nr_i;
    Calculate the LoMST score for r_i as $T_i = W_{r_i} - \overline{W}_{Nr_i}$;
end
Normalize the scores stored in T between 0 and 1 and rank them in decreasing order;
Identify the top N scores and store the corresponding observations as point anomalies in O_2;
TO= O_1 U O_2;

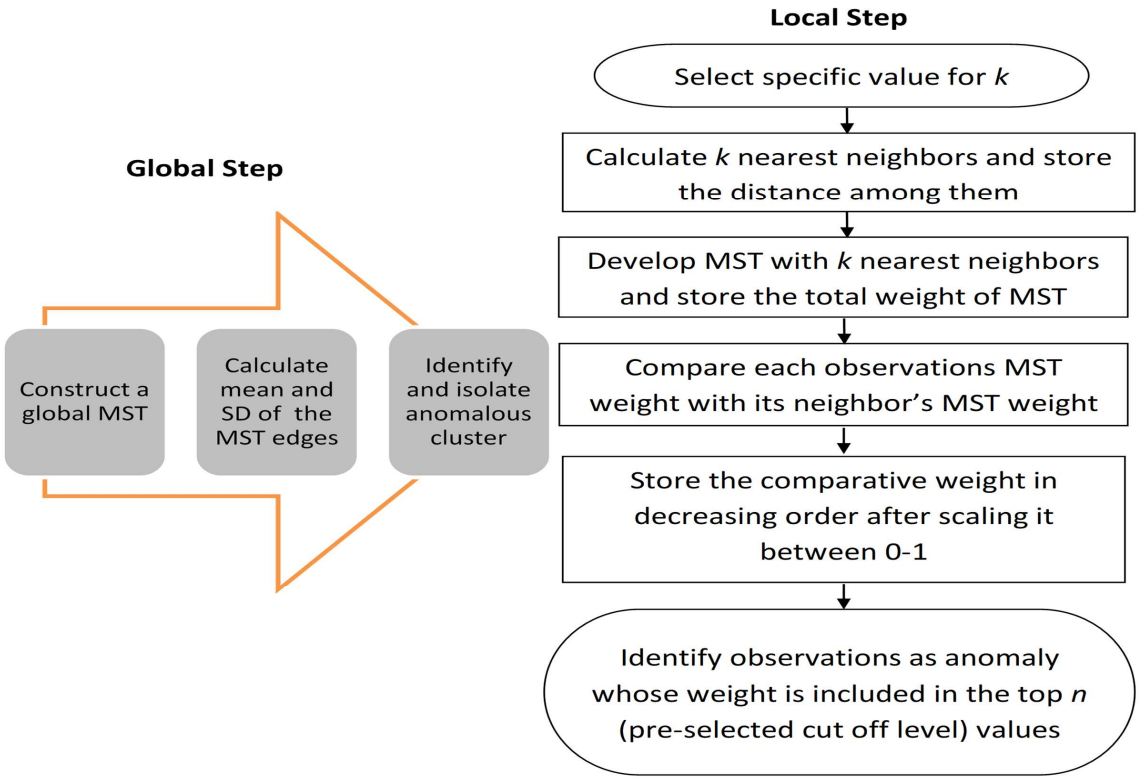
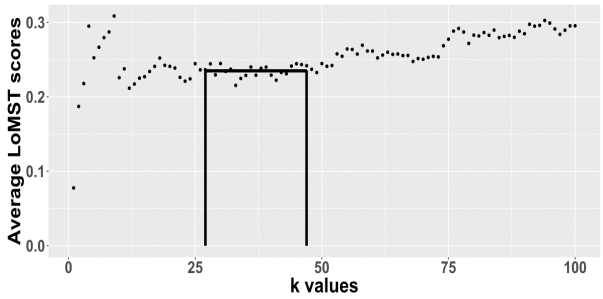


Figure 2.2: Flowchart of the proposed method.

destabilized again. Consequently, the average LoMST score could fluctuate once again. Based on this observation, our policy in choosing k is to select a range of k where the average LoMST scores are stable. If there are more than one stable range, we will then select the first one. Here, for the *Cardiotocography* dataset, we can choose a k range from 27-47 and for the *Glass* dataset we can choose a k range from 70-95. Within the identified stable range, which k to choose matters but matters less. What we suggest to select is the k value that returns the maximum standard deviation of the LoMST scores, because by maximizing the standard deviation among the LoMST scores, it increases the separation between the normal instances and anomaly instances and facilitates the detection mission.

(a) Cardioctography dataset.



(b) Glass dataset.

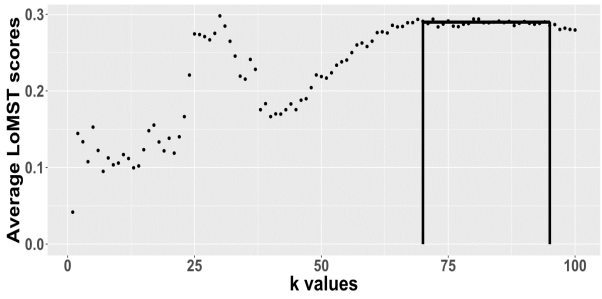


Figure 2.3: Select the range of k values.

2.3 Performance comparisons

In this section we wish to evaluate the performance of the MST-based method on a set of benchmark testing datasets listed in Table 1.1 compared to an array of well established methods in the literature.

As our method is dependent on the parameter k, we mainly focus on the nearest neighborhood based approaches for a fair assessment. [44] provided detailed experimentation on 12 popular nearest neighborhood approaches based on the 20 aforementioned data sets. These 12 methods are *Connectivity based Outlier Factor (COF), Local Density Factor (LDF), K-Nearest Neighbor (KNN), Outlier Detection using Indegree Number (ODIN), Local Outlier Factor (LOF), K-Nearest Neighbor Weight (KNNW), Simplified Local Outlier Factor (SLOF), Local Outlier Probabilities (LoOP), Influenced Outlierness (INFLO), Local Distance based Outlier Factor (LDOF) and Kernel Density Estimation Outlier Score (KDEOS)*. Traditional statistical process control (SPC) based approach could also be applied in the anomaly detection setting. We implemented one of the popular methods in SPC, the Hotelling T2 control chart [54]. We tested two versions while using the Hotelling T2 control chart: one with PCA that reduces the data dimension first and the other without PCA. It turned out that the T2 control chart without PCA performs slightly better than the PCA version. Hence, we only include the T2 result without PCA in the comparison tables to save space. In summary, we compare our MST-based approach with a total of 13 competing methods.

There is no guideline in the literature for how best to select k. [44] simply tried a range of k values (from 1 to 100) to obtain all the results and then choose the best k value for each method. In the first comparison, we follow the same approach, labeled as the "best k" comparison. The results are presented in Table 2.2. To better reflect the detection capability as they are compared to one another, we break down the comparative performance into four major categories, namely *Better*, *Equal*, *Close* and *Worse*, as explained in the table. Please note that the "best" k value in Table 2.2 may be different for respective methods.

LoMST shows superior performance and clearly outperforms other methods. In 13 of the 20 data sets, LoMST either exhibits a uniquely best detection performance or is tied with some other methods to achieve the best detection capability. In only 2 data sets, LoMST performs in the worse category, meaning that its detection capability is 20% lower than the best alternative. If we rank each of the 14 methods in a scale of 1 to 14 according to its actual performance in relative to others, then the average rank for the LoMST method is 2.2, while some of the closest competitors are *COF*(3.3), *LDF*(3.8), *KNNW*(4.5), *KNN*(5.0) and *LOF*(5.1).

Understandably, the "best k" is not practical, as in reality, people do not know the anomalies while selecting the best k. Since we have come up with a strategy to select a practical value of k, we use the same k value in the other 12 alternative methods that need the k (SPC does not need to know k). The performance comparison based on the practical k is presented in Table 2.3. We use the same performance breakdown as in Table 2.2. Our LoMST method continues to exhibit superior performance for being uniquely best in five of the 20 data sets and tieing other methods for achieving the best performance in another five data sets. The number of cases in the worse category is three. The average rank of the LoMST method is 2.8, slightly lower than that under the best k condition, while some of the closest competitors are *COF*(4.2), *KNNW*(4.3), *FBOD*(4.9), *KNN*(5.3), and *LDF*(5.7). The mean ranks are included in the table, as the last row. Please find in Table 2.4 the number of true positive detections of 14 methods under the best k setting, in which the best performance in every row is highlighted in boldface. To save space, we omit the same

Table 2.2: Performance comparison based on the best k value

Anomaly detection methods / Performance (number of data sets)	LoMST	COF	LDF	KNN	ODIN	LOF	KNNW	SimplifiedLOF	LoOP	INFLO	LDOF	FastABOD	KDEOS	SPC
Better (uniquely best result)	6	0	3	1	1	0	1	0	0	0	0	0	0	1
Equal (equal to the existing best result)	7	5	5	2	1	3	2	2	2	2	1	2	2	0
Close (within 20% of the existing best result)	5	10	6	7	8	8	7	6	6	5	7	5	1	1
Worse (not within 20% of the existing best result)	2	5	6	10	10	9	10	12	12	13	12	13	17	18
Mean relative rank	2.2	3.3	3.8	5.0	7.7	5.1	4.5	5.9	5.8	6.2	7.9	7.0	8.9	11.7

Table 2.3: Performance comparison based on the practical k chosen according to our selection policy

Anomaly detection methods / Performance (number of datasets)	LoMST	COF	LDF	KNN	ODIN	LOF	KNNW	SimplifiedLOF	LoOP	INFLO	LDOF	FastABOD	KDEOS	SPC
Better (uniquely best result)	5	2	1	1	2	0	0	0	0	0	0	2	0	1
Equal (achieving best result combinedly)	5	1	4	5	1	3	4	1	1	2	1	2	0	0
Close (within 20% of the best result)	7	11	4	6	8	7	9	8	7	7	6	6	3	3
Worse (not within 20% of the best result)	3	6	11	8	9	10	7	11	12	11	13	10	17	16
Mean Relative Rank	2.8	4.2	5.7	5.3	6.7	6.1	4.3	6.5	5.6	5.8	7.6	4.9	11.7	8.7

table under the practical k as it conveys the same message.

In Section 2.2, we mentioned both the mean-based comparison statistic and the mean-to-standard deviation based comparison statistic. Tables 2.2 and 2.3 present the comparison results using the mean-based statistic, which is our recommended default option. We also explore what if we use the mean-to-standard deviation ratio as the comparison statistic, and the results are presented in Table 2.5 and Table 2.6, respectively, depending on how k is selected. This time we included a smaller number of alternative methods in the comparison to save space because the performance of other methods lags too far behind the LoMST and the top competitors. Our comparison still shows that the LoMST performs the best among the alternatives, but its relative performance is slightly worse when using the mean-to-standard deviation ratio, as we expected. As explained in Section 2.2, our MST-based approach has the first stage of operation that removes the anomalous clusters, so it generally performs well when using the mean-based comparison statistic.

We further conduct some statistical test and see if the performance difference between the proposed method and its competitors is significant. For this purpose, we use a non-parametric method, the Friedman test [55]. Let n_a be the number of anomaly detection methods and n_d be

Table 2.4: Number of true positive detections of the 14 methods in the best k setting

	LoMST	COF	LDF	KNN	ODIN	LOF	KNNW	SimplifiedLOF	LoOP	INFLO	LDOF	FastABOD	KDEOS	SPC
WBC	**8**	**8**	**8**	**8**	4	**8**	**8**	6	4	4	4	7	1	5
Waveform	**35**	27	29	23	9	21	22	20	17	15	11	8	8	0
HeartDisease	**5**	4	4	4	4	4	4	4	4	4	2	2	1	0
WDBC	6	6	**7**	6	5	6	6	6	6	6	5	5	1	4
Glass	**3**	**3**	**3**	1	2	**3**	1	**3**	**3**	**3**	2	1	1	1
SpamBase	**78**	55	48	76	65	55	76	44	55	55	52	64	41	68
WPBC	**14**	11	13	10	12	10	9	10	10	11	13	10	**14**	5
Stamps	**6**	5	4	3	3	4	4	4	4	4	5	**6**	4	2
Parkinson	**4**	3	3	3	2	3	2	2	2	3	2	3	3	1
Lymphography	**6**	**6**	**6**	**6**	5	**6**	**6**	**6**	**6**	**6**	**6**	**6**	**6**	4
Ionosphere	108	107	104	107	99	105	108	104	101	101	101	108	97	**119**
PIMA	**8**	7	3	6	3	3	6	3	3	3	3	6	5	2
Shuttle	**7**	**7**	**7**	6	**7**	5	6	6	5	5	3	3	6	3
Cardiotocography	30	28	28	30	18	22	**31**	27	26	23	25	29	15	18
Arrhythmia	**5**	**5**	**5**	3	3	3	3	3	3	3	3	3	2	0
PageBlocks	46	45	**48**	46	40	45	47	47	45	43	44	45	12	38
Pendigits	**4**	3	1	0	0	1	1	1	1	0	1	1	1	0
KDDCup99	39	5	4	**87**	9	3	43	0	5	2	0	0	10	0
ALOI	314	346	226	261	**357**	319	265	316	336	345	0	0	193	69
Annthyroid	39	47	**60**	36	55	53	41	41	51	54	57	35	46	22

Table 2.5: Performance comparison based on best k value for alternative neighborhood comparison statistic

Anomaly detection methods / Performance (number of datasets)	LoMST	COF	LDF	KNN	KNNW
Better(uniquely best result)	7	1	3	1	1
Equal(equal to the existing best result)	4	5	5	4	2
Close(within 20% of the existing best result)	7	9	6	5	9
Worse(not within 20% of the existing best result)	2	5	6	10	8
Mean Relative Rank	3.0	3.1	3.5	4.8	4.5

the number of data sets. We define a matrix Ra whose entries in each row represent the detection method's rank for that specific data set. If there are tied values, we assign to each tied value the average of the ranks that would have been assigned without ties. For example, suppose we have two tied methods both with rank 7. If there is no tie, they should be assigned rank 7 and rank 8. A Friedman test then uses the average of them, which is 7.5, as the rank value for both of these methods instead of 7. Under the null hypothesis that all methods perform the same, the Friedman

Table 2.6: Performance comparison based on our k selection policy for alternative neighborhood comparison statistic

Performance (number of datasets) \ Anomaly detection methods	LoMST	COF	LDF	KNN	KNNW
Better(uniquely best result)	6	0	1	1	0
Equal(achieving best result combinedly)	2	2	5	5	3
Close(within 20% of the best result)	8	12	7	8	10
Worse(not within 20% of the best result)	4	6	7	6	7
Mean Relative Rank	3.8	4.4	5.6	5.0	4.2

statistic,

$$\chi_F^2 = \frac{12n_d}{n_a(n_a+1)}\left(\sum_{l=1}^{n_a}\overline{Ra_l^2} - \frac{n_a(n_a+1)^2}{4}\right), \tag{2.1}$$

follows a Chi-squared distribution with $n_a - 1$ degrees of freedom, where $\overline{Ra_l}$ is the average value of column l = 1, 2, ..., n_a. We have done the tests for both best k and practical k settings and found the p-values (1.27×10^{-12} and 1.07×10^{-10}, respectively) significant enough to reject the null hypothesis.

To find out whether our method is significantly different from other methods, we also conducted some post hoc analysis. Fig. 2.4 presents the post hoc analysis on the ranking data for the practical k setting, showing that the LoMST's ranking is significantly higher (lower in numeric sense) than other competing algorithms at the 0.05 level of significance. The detailed pairwise comparisons using respective p-values for both best k and practical k settings are presented in Table 2.7. The p-values are calculated using Conover post-hoc test [56]. We have used False Discovery Rate (FDR) approach [57] to adjust the p-values for multiple comparisons. Other than between COF and LoMST, which shows a marginal significance, all other pairwise comparisons shown a sufficiently significant difference, suggesting that the LoMST is superior and produces a better performance.

We summarize our method's performance with respect to the data size in Table 2.8. It is ev-

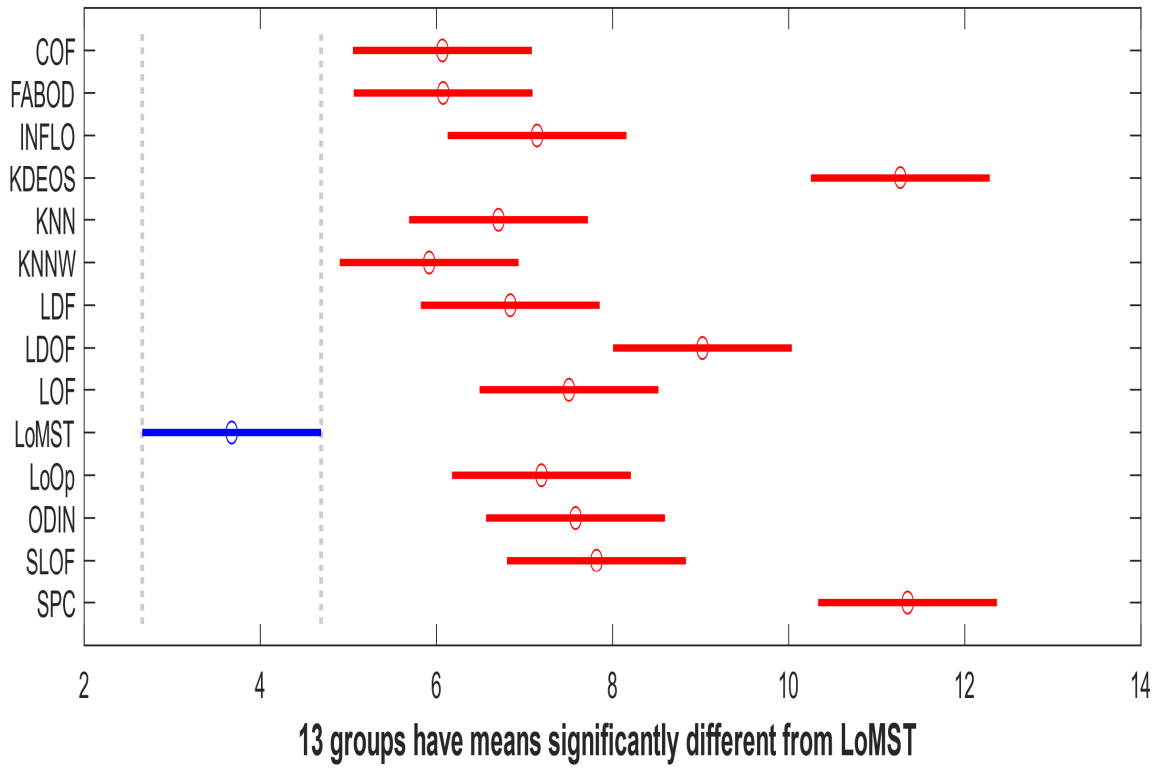


Figure 2.4: Post hoc analysis on the ranking data obtained by the Friedman test. This analysis is under the practical k setting.

ident from the table that our method performs well and came out as the best method in a wide range of scenarios. If we look at the two extreme considering highest number of observations (m=60,632, *KDDCup99* data set) and the highest number of attributes (p = 259, *Arrhythmia* data set), we see that our method performs reasonably well in both cases. So, we can at least guarantee the success of our method up to this point if not beyond. In addition, we think that our method will perform better when the number of anomalies is in the range of less than a hundred to few hundreds. Also, it is our understanding that our method will be more suitable for the big data extreme (number of observations>>number of attributes) rather than the high-dimension data (number of attributes>>number of observations).

In general, the proposed technique will perform better in the presence of nonlinear man-

Table 2.7: P-values of pairwise comparison of LoMST method with the competing methods.

	Best k	Practical k
COF	1.28×10^{-1}	6.47×10^{-2}
LDF	3.18×10^{-2}	2.96×10^{-4}
KNN	3.78×10^{-4}	2.24×10^{-3}
ODIN	2.03×10^{-8}	6.11×10^{-6}
LOF	5.66×10^{-5}	2.01×10^{-5}
KNNW	1.95×10^{-3}	4.02×10^{-2}
SLOF	2.16×10^{-6}	3.92×10^{-6}
LoOP	2.02×10^{-6}	2.77×10^{-4}
INFLO	8.13×10^{-7}	9.79×10^{-5}
LDOF	1.30×10^{-9}	1.44×10^{-8}
FABOD	1.02×10^{-7}	1.14×10^{-2}
KDEOS	1.28×10^{-11}	7.04×10^{-17}
SPC	3.08×10^{-18}	1.05×10^{-9}

ifolds in a data space. Unfortunately, there is no known efficient method to test the data space structure quickly, meaning that finding out whether the data space is a flat linear vector space or not usually depends on using different distance metrics to try a detection task. If using MST or other graph-based distances produces the same/similar outcome as using the Euclidean distance, then it implies that the data space is flat and linear; otherwise, it implies that the data space embeds a complicated structure. It is, therefore, safer to use the MST-based method, because even if there is no complicated structure in the data space, the MST can approximate the Euclidean distance between two points, too. The downside in such circumstance is that using MST costs more computationally. Still, since our primary concern now is to boost the detection capability in unsupervised anomaly detection, we think the computational expense is a necessary evil to bear with.

Another important point is, in practical sense we do not know the number of the true anomalies to set N. As explained earlier, N is usually chosen in practice to be larger than potential number of anomalies but small enough to make the subsequent identification operations feasible. For anomaly detection problems, the false alarm rate is generally high, in order to boost the detection

Table 2.8: Performance of the LoMST method

Dataset	Number of observations over number of attributes	Number of observations	Number of attributes	Number of anomalies	Rank (Best 'k' setting)	Rank (Practical 'k' setting)
Arrhythmia	2	450	259	12	1	1
SpamBase	81	4601	57	280	1	1
KDDCup99	1479	60632	41	200	3	2
WPBC	6	198	33	47	1	3
Ionosphere	11	351	32	126	2	3
WBC	51	367	30	10	1	1
ALOI	1852	50000	27	1508	7	7
Parkinson	9	195	22	5	1	1
Annthyroid	343	7200	21	347	11	10
Waveform	164	3443	21	100	1	1
Cardiotocography	102	2126	21	86	2	3
Lymphography	8	148	19	6	1	1
Pendigits	617	9868	16	20	1	2
HeartDisease	21	270	13	7	1	1
PageBlocks	548	5473	10	99	4	7
Stamps	38	340	9	16	1	6
Shuttle	113	1013	9	13	1	1
WDBC	13	454	9	10	2	2
Pima	96	768	8	26	1	1
Glass	31	214	7	9	1	1

capability. This is a common phenomenon in all anomaly detection methods. To see this point, consider the following example. In the WBC dataset, there are 10 true anomalies and 213 normal instances. When using $N = 20$ as the cutoff, meaning that the LoMST method flags 20 instances as anomalies, 10 of the 20 are truly anomalies and 10 are falsely tagged as anomalies. As such, the detection rate is 10/10 (100%), whereas the false alarms are 10/20 (50% of all alarms). We would argue that despite the relatively high proportion of the false alarms, the anomaly detection method is still practically useful, particularly as a pre-screening tool. By narrowing down the candidate anomalies from the whole set to 20, which is an order of magnitude decrease in data amount, it helps human experts a great deal to follow up with each circumstance and decide how to improve their processes. We believe that a fully automated anomaly detection is not yet realistic in the near future, due to the challenging nature of the problem and the relatively lack of advancement in the state-of-the-art. Therefore, a useful pre-screening tool, as the current anomaly detection offers, would be valuable in filling the void, while researchers strive for the ultimate, full automation goal.

2.4 O-LoMST: An online local MST based anomaly detection algorithm

With the increasing availability of streaming data, industries nowadays are striving for an automated online anomaly detection algorithm that can analyze data stream and detect anomalous patterns in real-time. Such an online algorithm should detect anomalies on the fly, without storing all, or a very long stretch of, the historical data. It should be able to update its control mechanism for anomaly detection upon receiving new data. Moreover, the algorithm must work in an unsupervised way; i.e., in the absence of class labeling information *a priori*. These fundamental requirements limit the application of traditional anomaly detection approaches in streaming scenarios. In this section, we want to propose an online anomaly detection method, based on the offline LoMST method we just developed. As the offline method specifically handle the issue of nonlinear manifold embedding in data spaces and use a minimum spanning tree to approximate and capture the manifold structures, leading to a much enhanced detection ability, the primary objective is now to make the offline method applicable to streaming data and address the aforementioned unique online issues.

Traditional anomaly detection algorithms detect anomalies through the comparison of candidate data points with all observations in the historical data set. However, the same cannot be done when data are arriving in the streaming form, i.e., the data appear as a single stream or in small batches. In streaming data, an algorithm can only see the current batch and has limited information about the past, because the algorithm cannot, or does not want to, store all the previous data. Instead the algorithm only “memorizes" a handful of the most recent historical data, in order to avoid large storage requirements or for other practical considerations. The offline LoMST [58] indeed needs the knowledge of the nearest neighbors based on the whole dataset. To reach a verdict about a current data points, the offline method uses the information of all the past and current data points; all these actions need to be avoided, should the algorithm be made online compatible to handle streaming data.

Instead of storing all past data points, a streaming algorithm should presumably devise and make use of an indicator variable that can summarize the common characteristics of the past observations. The algorithm should embody a provision to continually update this indicator to reflect information received via the arrival of new data, as well as update the decision threshold that will be used for deciding in real-time whether an observation is an anomaly or not. All these are precisely the goals of our research undertaking reported in this section, namely that we propose an online version of the LoMST, referred to as online LoMST (O-LoMST), which has the aforementioned online capabilities.

A number of online anomaly detection algorithms have also been proposed recently. The first variety includes those which are partially online, [59, 60] meaning that they have an initial offline phase to learn about the regular (normal) clusters and their characteristics, and then detect anomalies online using the knowledge from the offline phase. The online part is self adaptive and the control mechanism (e.g., cluster center) gets periodically updated according to the changes observed in incoming data. The second variety includes the online version of the subspace algorithms [61, 62] which have been introduced to deal with the difficulties associated with high dimensional data stream. There are also online algorithms [63, 64] that use statistical tests to detect anomalies in real-time. All these online methods have the same limitations as their offline counterparts, which is that they do not have the provision to deal with data with inherent nonlinear manifold structure. That is why we believe that converting the offline LoMST still warrants the research effort. The resulting online method is to be explained in the following subsection.

2.4.1 O-LoMST algorithm

To construct the regular, offline version of MST, one uses all the data points, which is not possible in the presence of the streaming data. In streaming scenarios, data points arrive either as a single entity or in small batches. Meanwhile, we do not want to store all the previous data points in memory while building and updating the MST, in order to be more efficient and save storage

space.

To solve this problem, we use a local version of the regular MST in a streaming fashion. By "local MST", we mean that instead of using all the data points, the MST is to be constructed using the nearby points only. To facilitate the subsequent presentation, let us define some notations first. Denote each streaming batch by the index, j, and their size as B_j. Denote by t_j the data point that is currently under evaluation. To reduce the search space for neighbors, we choose to use a fixed-size candidate set of neighbors. It consists of the data points which are in close temporal vicinity to the time stamp under consideration, decided by a user controlled parameter, c. Given a c, the closest available c data points with respect to t_j form the candidate set. Certain care needs to be taken to locate the c closest neighbors. Although the neighbors are generally in the $\pm c/2$ range around t_j, a temporal truncation can make the data spread asymmetrically before and after t_j. For example, suppose that we are evaluating the 50^{th} time stamp from a batch that runs the time stamp index from 1 to 100 and $c = 30$. Then the time stamps ranging from 51–65 and 35–49 form the candidate set. But if the batch size is 60, meaning that the time stamp index runs from 1 to 60, then the time stamps 51–60 and 30–49 would form the candidate set instead.

Given a candidate set, Q_{t_j}, a search is conducted to find the k nearest neighbors of t_j, so that, $\|t_j - x_1\|_2 < \|t_j - x_2\|_2 < \cdots < \|t_j - x_k\|_2 < \cdots < \|t_j - x_c\|_2$, where $x_1, \ldots, x_k, \ldots, x_c \in Q_{t_j}$. We refer to the k nearest neighbors (NN), i.e., from x_1 to x_k, as the temporal k-NN of t_j and store them in $\mathfrak{N}_{t_j}$, i.e., $\mathfrak{N}_{t_j} = \{x_1, x_2, \ldots, x_k\}$.

After identifying the neighbors, we use those neighbor points along with t_j to build a dense graph object, i.e., fully connected graph. To create the MST from this dense graph, there are many different approaches. We choose to use Prim's algorithm [10] due to its effectiveness in the case of the fully connected graph. The algorithm selects a total of k edges from a collection of $\binom{k+1}{2}$ edges stored in $E_{t_j} = \{e_{t_j,x_1}, \cdots, e_{t_j,x_k}, e_{x_1,x_2}, \cdots, e_{x_1,x_k}, \cdots, e_{x_{k-1},x_k}\}$, such that sum of the length of the selected edges are minimum. Here e_{t_j,x_k} denotes the edge between two data points, t_j and x_k; other similar notations follow this convention. The selected edges by the Prim's algorithm are the

edges in the resulting MST, connecting t_j and its neighbors. The total length of this MST is stored in W_{t_j}. This above step is then repeated for the remaining data points in batch j.

After a local MST is constructed for all time stamps in batch j, the connectedness of time stamp t_j is then compared with that of its neighbors, as expressed in (2.2), and the difference is denoted by $\mathfrak{L}_{t_j}$, such as

$$\mathfrak{L}_{t_j} = W_{t_j} - \overline{W}_{Nr_{t_j}}, \tag{2.2}$$

where $\overline{W}_{Nr_{t_j}}$ is the average of the total tree length associated with data points, other than t_j, in $\mathfrak{N}_{t_j}$. This comparison score, $\mathfrak{L}_{t_j}$, is to be used for the final anomaly evaluation.

The mean and standard deviation (SD) of all the comparison scores in a particular batch will be calculated, respectively, using (2.3) and (2.4), as:

$$\mu_j = \frac{1}{B_j} \sum_{t_j=1}^{B_j} \mathfrak{L}_{t_j}. \tag{2.3}$$

$$\sigma_j = \sqrt{\frac{1}{B_j} \sum_{t_j=1}^{B_j} (\mathfrak{L}_{t_j} - \mu_j)^2}. \tag{2.4}$$

These batch-specific mean and SD scores will be used to update the overall mean and SD from earlier batches, using the formula in (2.5) and (2.6), respectively, upon receiving new observations. Here μ_{old} and σ_{old} is the mean and SD value that are available to us prior to seeing the observations in the j^{th} batch.

$$\mu_{new} = \frac{1}{\sum_{i=1}^{j} B_i} \left[B_j \cdot \mu_j + \sum_{i=1}^{j-1} B_i \cdot \mu_{old} \right], \tag{2.5}$$

$$\sigma^2_{new} = \frac{1}{\sum_{i=1}^{j} B_i}\left[B_j \cdot \sigma_j^2 + \sum_{i=1}^{j-1} B_i \cdot \sigma^2_{old}\right] + \frac{\sum_{i=1}^{j-1} B_i \cdot B_j}{(\sum_{i=1}^{j} B_i)^2} \cdot (\mu_{old} - \mu_j)^2. \tag{2.6}$$

To reflect any major change in the data generation process, (e.g., seasonal changes, maintenance operations, installation of new equipment etc.), we choose to divide the detection process into blocks and restart the mean and SD update process at the beginning of each block. The duration of each block can be selected by the domain expert based on the process under consideration. To decide on whether a time stamp is an anomaly or not, we use $\mu_{new} \pm 3\sigma_{new}$, the typical 3-sigma control limits, as the threshold. If $\mathfrak{L}_{t_j}$ is greater than the threshold, the corresponding t_j is marked as an anomaly. After a batch of streaming points are evaluated, the mean and SD scores will be updated, and the same procedure will be repeated on the next available batch. Once all the batches from a block are completed, the detection process will restart again from the next block.

For a clear understanding, the algorithm steps are summarized in Algorithm 2.

2.5 Application to the hydropower plant data

Now we want to evaluate the performance of both offline and online version of the LoMST approach on hydropower data. Besides our own LoMST method, we have also applied two other popular anomaly detection methods on the same hydropower data. The two other methods are: LOF [17] and SOD [32] method. LOF represents the 12 neighborhood-based methods considered for comparison in Section 2.3 and is arguably the most popular method in the anomaly detection literature. SOD is a representative of the subspace methods and we are curious to see how a subspace method could do to the real application data. However, we want to mention here that finding the right subspace is usually even harder than finding the candidate anomalies, and as a result, the neighborhood-based methods often outperform the subspace-based methods. For instance, if we compare SOD based on the practical k approach with LoMST and the other 13 methods in Sec-

ALGORITHM 2: O-LoMST algorithm for anomaly detection

Input : Block size, Z, streaming batch size, B_j in a block, number of candidate for neighbor selection, c, number of temporal neighbors, k

Output: List of anomalous time stamps from batch j, denoted as a_j

for *each block of size* Z **do**
  Initialize μ and σ;
  for *each streaming batch* j **do**
    for *each time stamp* $t \in$ *batch* j **do**
      Form a set of candidate temporal neighbors, Q_{t_j};
      Determine the k nearest neighbors and save them in $\mathfrak{N}_{t_j}$;
      Construct a local MST using time stamp t_j and its neighbors in $\mathfrak{N}_{t_j}$;
      Calculate the total length of the resulting MST, W_{t_j};
      Calculate the mean, $\overline{W}_{\mathfrak{N}_{t_j}}$, of the total length of the local MSTs associated with all neighbors in $\mathfrak{N}_{t_j}$;
      Calculate the LoMST score for t_j as $\mathfrak{L}_{t_j} = W_{t_j} - \overline{W}_{\mathfrak{N}_{t_j}}$;
    end
    Calculate the mean, μ_j, and SD, σ_j, of the LoMST scores;
    Update the overall mean, μ, and SD, σ;
    List the time stamps as anomalies if $\mathfrak{L}_{t_j} \geq \mu + 3\sigma$ and store them in a_j;
    Store the recent $c/2$ time stamps of batch j and discard others;
  end
end

tion 2.3, its ranking in the four categories would be 0, 0, 8, 12 and its average ranking would be 6.4, much worse than the top competitors.

For all three methods, we need to specify the value of the nearest neighbors k. Selecting a suitable value for the neighborhood parameter, k, is not an easy task. This value is dependent on the size of the clusters present in the system. Ideally, we should select a value which is larger than a potential anomalous cluster but smaller than the regular cluster. In this case, we were lucky enough to get a suggestion from the domain expert about the possible size of an anomaly cluster. Based on the domain expert's suggestion, we decide to consider the value of k in a range of 10-20 and find the anomaly scores for each k in the range. Then we took the average of these scores as the final anomaly score for each of the instances. We followed this principle for both the LOF method and our offline LoMST method. For SOD, we need to select two parameters instead of one: one is k, while the other one is the number of reference points for forming the subspace. To maintain

the comparability with LoMST and LOF, we choose $k = 15$ for SOD, which is the middle point of the above-suggested range. Concerning the number of reference points, it should be smaller than k but not too small a value that may render instability in SOD. We explore a few options and finally settle on 10. Below 10, the SOD method becomes unstable.

We also include the online version of the LoMST we proposed in the last section. In order to evaluate the performance of the proposed approach for streaming data, we pass the time stamps from the hydropower dataset in small batches to mimic the streaming scenario. We choose a fixed batch size, $B = 100$ and a block size, $Z = 7000$, meaning that in total there are 92 batches divided into 2 blocks to cover all of the time stamps. We have used $k = 15$ as neighborhood size. For the temporal neighborhood size, c, we settle on the set of 50 candidate time stamps, i.e., $c = 50$, which is a half of the batch size.

After choosing these algorithmic parameters, we have applied our online detection method to all 92 batches and detected a total number of 207 anomalous data points. The anomaly detection procedure regarding a particular batch is shown in Fig. 2.5. After each streaming batch, analysts can carefully evaluate the anomalies detected, pinpoint the root cause of these anomalies, and take proper actions if necessary. We suggest the practitioners to stop the entire process if they detect too many anomalies, e.g., more than 60% in a batch. Because too many anomalies signal a major change in the process. Consequently, operators should carefully examine the process for any major malfunction.

We report the top 100 anomalies identified by each method according to their anomaly scores (the bigger the score, the more likely it is anomalous) in Table 2.9. Due to the space limitation, we skip some rows in the table. The performance of the offline methods is reasonably consistent as 14 out of the top 30 probable anomalies identified by these methods are common. This similarity continues even if we go beyond 30 time stamps. These three methods work differently, especially that SOD is from another family of methods and completely different than LOF and LoMST. In spite of their differences, they have returned similar results for the hydropower dataset. This serves

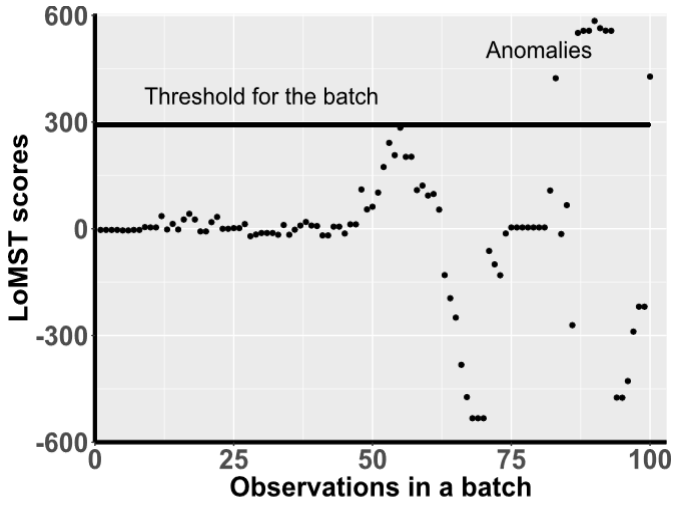


Figure 2.5: Anomaly detection in a streaming batch

as a way to cross validate the detecting outcomes, as the true anomalies are unknown. We report the top 100 time stamps as anomalies to the data provider. The domain expert checks the physical system and agrees that these present valid concerns and the method provides valuable tips for trouble shooting.

The three methods do have differences in their detection outcomes. LOF method completely missed the 4^{th} of July time stamps, although almost half of the 100 top anomaly prone timestamps returned by both SOD and LoMST method belong to this day. We investigate the issue and find that most of the timestamps in July corresponds to low active power, whereas the timestamps from 4^{th} of July are marked with abnormally high active power. The rest of the attributes behaves identically as other days of July. We know that when the number of attributes increases, nearest neighborhood methods usually fall short of detecting anomalies if abnormal values only happen to one or few dimensions. This is where the subspaces method can do better (if the abnormal value subspace is successfully identified). It is therefore not surprising to see that the SOD method detects these anomalies correctly, but it is truly encouraging to see that LoMST is capable of detecting these anomalies, even though LoMST a neighborhood-based method. It supports our claim that MST approximates the intrinsic distance among observations in a structured data space. On the other

hand, LOF and LoMST, being a local method, successfully identified point anomalies on the 16^{th} of April, while SOD method failed to identify them. In a nutshell, LoMST method attains the merit of subspace-based methods without losing the benefits of local neighborhood-based methods.

We also observe that findings from the online LoMST are very similar to those by the offline LoMST, which substantiates the success of the online LoMST approach. As this is an unsupervised problem, we do not have the luxury to have a training set containing a mix of normal and anomalous values. The only way to verify the sanity of the outcome is to ask the domain expert and the operator of the system to double check. They can analyze the detected anomalies one by one, inspect the particular components in the plant, and feedback to us with their understandings. We consider ourselves extremely lucky here as the domain experts confirm the reasonableness of the detection outcomes.

If we look very closely to the anomalous time stamps, we observe that anomalies from all four approaches have similar patterns and they can be conveniently grouped into some particular days. Even if the individual time stamps are not exactly the same, they fall into the same group. We have listed these anomaly prone days in Table 2.10, and we note that they are very similar irrespective of the detection methods used. If we compare the performance of online and regular LoMST approach, we can observe that, out of 11 anomaly prone days, they share 9 of them. Despite the online features employed by O-LoMST, for instance, using only the current batch combined with summary statistics from prior batches, the online algorithm does not seem to sacrifice the detection performance too much. Comparing LoMST with LOF and SOD, the online LoMST in fact outperforms offline LOF and SOD in terms of overall detection performance.

Anomaly detection does not immediately reveal the root causes causing the anomalies. Finding out which variables contribute to the anomalies provides a nice interpretability of each anomaly and helps the domain expert to verify the root causes and then fix them (if genuine). Given that the anomalies are identified now, the original unsupervised learning problem is translated into a supervised learning problem, and for that, we build a classification and regression tree (CART)

Table 2.9: Summary of the top 100 anomalies returned by the four approaches.

O-LoMST	LoMST	LOF	SOD
4/16/2015 23:10	4/16/2015 16:00	4/16/2015 16:00	7/4/2015 0:00
7/4/2015 4:20	4/16/2015 23:10	4/16/2015 23:10	7/4/2015 4:20
7/4/2015 4:30	7/4/2015 4:40	9/13/2015 7:00	7/4/2015 4:30
7/4/2015 4:40	7/4/2015 4:50	9/13/2015 19:30	7/4/2015 4:40
9/13/2015 19:00	7/4/2015 5:00	9/13/2015 19:40	7/4/2015 5:30
9/13/2015 21:40	7/4/2015 5:20	9/14/2015 0:40	7/4/2015 5:40
9/14/2015 0:40	7/4/2015 5:30	9/14/2015 1:00	7/4/2015 5:50
9/14/2015 1:00	7/4/2015 6:20	9/14/2015 1:10	7/4/2015 6:20
9/14/2015 1:10	7/4/2015 9:10	9/14/2015 2:00	7/4/2015 6:30
9/14/2015 1:50	7/4/2015 9:30	9/14/2015 2:10	7/4/2015 6:50
9/14/2015 8:00	7/4/2015 9:40	9/14/2015 8:00	7/4/2015 7:00
9/14/2015 8:10	7/4/2015 9:50	9/14/2015 8:10	7/4/2015 7:50
9/14/2015 8:20	7/4/2015 10:10	9/14/2015 8:20	7/4/2015 8:20
9/14/2015 8:30	7/4/2015 10:20	9/14/2015 8:30	7/4/2015 8:30
9/14/2015 8:40	7/4/2015 10:30	9/14/2015 8:40	9/13/2015 7:00
9/14/2015 8:50	7/4/2015 10:40	9/16/2015 10:50	9/13/2015 21:40
9/14/2015 9:00	7/4/2015 10:50	9/17/2015 11:30	9/14/2015 1:00
9/14/2015 9:10	7/4/2015 11:10	10/3/2015 14:40	9/14/2015 1:10
9/14/2015 9:20	7/4/2015 11:20	10/4/2015 3:10	9/14/2015 1:50
.....	.....	.....	.....
9/15/2015 20:50	7/4/2015 13:50	10/13/2015 5:45	9/14/2015 8:00
9/16/2015 10:30	9/13/2015 7:00	10/13/2015 6:35	9/14/2015 8:10
9/16/2015 10:50	9/13/2015 19:10	10/13/2015 8:15	9/14/2015 13:05
9/16/2015 11:00	9/14/2015 1:00	10/14/2015 7:55	9/16/2015 10:50
9/17/2015 11:30	9/14/2015 1:10	10/14/2015 8:15	9/16/2015 11:00
10/3/2015 14:40	9/14/2015 8:00	10/14/2015 23:15	10/3/2015 14:40
10/4/2015 3:10	9/14/2015 8:10	10/14/2015 23:35	10/4/2015 3:10
10/4/2015 3:40	9/14/2015 13:20	11/2/2015 9:56	10/4/2015 4:20
10/4/2015 4:10	9/14/2015 13:50	1/2/2016 9:10	10/4/2015 4:30
10/4/2015 4:20	9/14/2015 14:10	1/2/2016 9:20	10/13/2015 5:45
10/4/2015 4:30	9/16/2015 10:50	1/2/2016 9:30	10/13/2015 6:35
10/4/2015 9:00	10/13/2015 8:15	1/2/2016 21:40	10/13/2015 8:25
.....	.....	.....	.....
10/13/2015 5:25	10/14/2015 7:25	1/9/2016 6:50	10/14/2015 7:25
10/13/2015 5:35	10/14/2015 7:35	1/9/2016 18:00	11/2/2015 9:56
10/13/2015 5:45	1/2/2016 9:10	1/9/2016 18:10	1/2/2016 1:30
10/14/2015 7:25	1/2/2016 9:20	1/9/2016 18:20	1/2/2016 9:10
10/14/2015 7:35	1/2/2016 9:30	1/9/2016 18:30	1/2/2016 9:20
10/14/2015 7:55	1/9/2016 6:50	1/9/2016 18:40	1/2/2016 21:40
.....	.....	.....	.....
1/9/2016 18:00	1/9/2016 18:40	1/11/2016 11:30	1/9/2016 6:50
1/9/2016 18:10	1/11/2016 1:30	1/11/2016 11:40	1/9/2016 18:30
1/9/2016 18:20	1/11/2016 11:50	1/11/2016 11:50	1/11/2016 1:30
1/9/2016 18:30	1/11/2016 12:00	1/11/2016 12:00	1/11/2016 11:30
1/9/2016 18:40	1/11/2016 13:00	1/11/2016 13:00	1/11/2016 12:00
1/11/2016 13:40	1/11/2016 13:50	1/11/2016 13:50	1/11/2016 13:50
1/11/2016 13:50	1/12/2016 11:20	1/11/2016 14:40	1/11/2016 14:40
1/11/2016 14:40	1/12/2016 11:30	1/12/2016 11:20	1/12/2016 11:20
1/12/2016 11:20	1/12/2016 11:40	1/12/2016 11:30	1/12/2016 11:30
.....	.....	.....	.....

Table 2.10: Most anomaly prone days according to the four methods

O-LoMST	LoMST	LOF	SOD
April 16th	April 16th	April 16th	July 4th
July 4th	July 4th	September 13th	September 13th
September 13th	September 13th	September 14th	September 14th
September 14th	September 14th	October 3rd	October 3rd
October 3rd	October 4th	October 4th	October 4th
October 4th	October 13th	October 13th	October 13th
October 13th	October 14th	October 14th	October 14th
October 14th	January 2nd	January 2nd	January 2nd
November 1st	January 9th	January 9th	January 9th
January 9th	January 11th	January 11th	January 11th
January 11th	January 12th	January 12th	January 12th

[65] using the `R` package `rpart` with the package's default parameter values. We at first discard the July 4^{th} timestamps from the top 100 time stamps, as the reason for them was quite straight forward. We proceed with the remaining of the top 100 timestamps and assign them a response value of 1, and all other data records (outside the top 100 timestamps) in the dataset a response value of 0 (meaning normal condition). The resulting tree is shown in Fig. 2.6.

From this decision tree we can see that the variable *Oil Temperature of Bearing 4*, *Air Pressure*, *Turbine Vertical Vibration* and *Delta Oil temp - Air Temp of Bearing 1* can correctly classify 25 anomalies based on the right combination of their conditions. One such condition is when the oil temperature of bearing 4 is less than 27.216 degrees Celsius, the turbine generator almost surely behaves strangely, and this condition leads to eleven anomalous observations consistently. When we report this finding to the domain expert, he deems this a key finding. During the subsequent preventive maintenance operation of the said turbine, it is confirmed that bearing 4 indeed needs repair to avoid future damage or costly interruption of the turbine operation.

2.6 Summary

We proposed a new dissimilarity metric based on the concept of minimum spanning tree for isolating local, point-wise anomalies from the normal observations in a structured data space. Rather

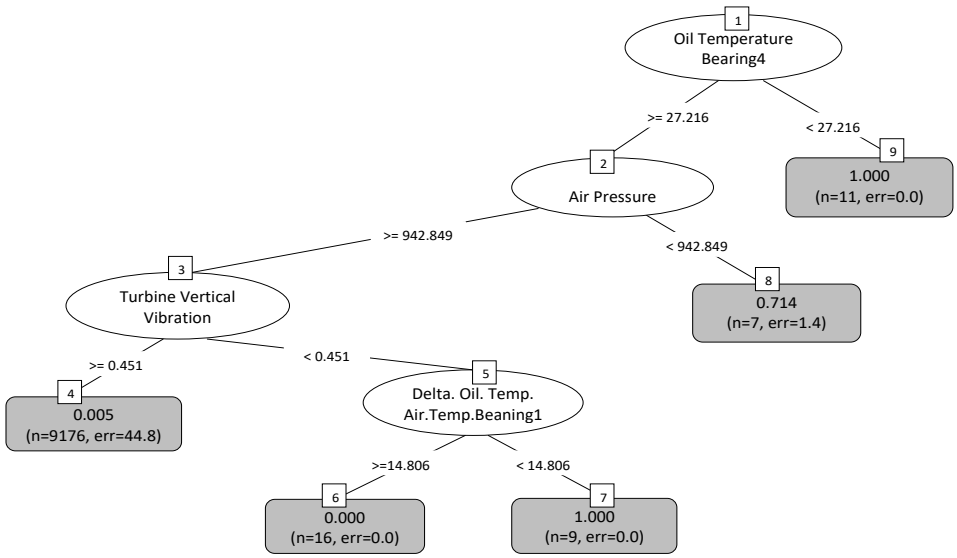


Figure 2.6: Decision tree based on the anomalies identified by LoMST method.

than applying MST to the entire dataset, we choose to follow a two-stage procedure. At first, a global MST is used to separate the distant anomalous clusters from the rest of the dataset. Then we build local MSTs for the remaining instances to detect point wise anomalies. The proposed MST-based method is effective and registers the best performance when it is compared with a wide array of methods on 20 benchmark datasets. The superiority of the proposed method inspires us to apply it to a real-life hydropower dataset. The MST-based method is successful in detecting different families of anomalies, achieving the merit of subspace-based methods without losing the benefits of local neighborhood-based methods. The validity of the anomalies detected is cross validated by two other anomaly detection methods and confirmed by the domain experts and maintenance operators who provided us the hydropower data in the first place. Root causes and threshold values for key attributes that contribute to the anomalies are determined in the form of a decision tree. The generated knowledge from the anomaly detection analyses helps service engineers monitor the turbine operation continuously, and potentially diagnose and predict the malfunctions of turbines in time. To keep up with the industry needs we also develop an online version of the LoMST method which proves to be equally effective when compared with the offline version.

3. NS-NMF: A NEIGHBORHOOD STRUCTURE ASSISTED NONNEGATIVE MATRIX FACTORIZATION APPROACH FOR UNSUPERVISED POINT ANOMALY DETECTION

Dimensionality reduction is considered as an important step for ensuring competitive performance in unsupervised learning such as anomaly detection. Nonnegative matrix factorization (NMF) is a popular and widely used method to accomplish this goal. But NMF, together with its recent, enhanced version, like graph regularized NMF or symmetric NMF, does not have the provision to include the neighborhood structure information and, as a result, may fail to provide satisfactory performance in presence of nonlinear manifold structure. To address that shortcoming, we propose to consider and incorporate the neighborhood structural similarity information within the NMF framework by modeling the data through a minimum spanning tree. We label the resulting method as the neighborhood structure assisted NMF. By comparing the formulation and properties of the neighborhood structure assisted NMF with other versions of NMF including graph regularized NMF and symmetric NMF, it is apparent that the inclusion of the neighborhood structure information using minimum spanning tree makes a key difference. We further devise both offline and online algorithmic versions of the proposed method. Empirical comparisons using twenty benchmark datasets as well as an industrial dataset extracted from a hydropower plant demonstrate the superiority of the neighborhood structure assisted NMF and support our claim of merit.

3.1 Introduction

Matrix factorization (MF) is one popular framework for finding the low dimensional embedding in a high dimensional dataset. MF based approaches have been employed successfully to represent and group high dimensional data for better learning capability. Methods which can be described using the traditional matrix factorization framework such as the principal component analysis (PCA) produces low rank matrices consisting of negative values and positive and negative weights, which tends to cancel each other in reconstructing the original matrix, and hence provides

no intuitive meaning. The nonnegative matrix factorization [38, NMF] method, which imposes the non-negativity constraint in matrix factorization and only allows additive linear combinations of components, comes out as a better candidate for finding the low dimensional representation of high dimensional data. NMF has the capability of generating both clustering assignments and meaningful attribute distribution in two separate matrices. Immediately after its introduction, NMF not only becomes a powerful tool for clustering [39], but it also shows enough potential in anomaly detection [66, 40, 41].

In the presence of complicated manifolds, however, researchers notice that NMF starts to lose its efficiency [42, 43] as it only tries to approximate the data without trying to mimic the similarity among observations in the latent space. In other words, the shortcoming of the original NMF is attributed to that it has no provision to include the neighborhood structure information during the calculation of the factored matrices and thus cannot approximate the manifold embedded in the data.

Our research finds it beneficial to include the structural similarity information of data, along with the original attribute information, in the objective function of an NMF-based method. Our specific approach is as follows. First, we convert the original data matrix into a graph object where each node represents an observation and each edge represents the virtual connection between a pair of data points. Then we apply a minimum spanning tree [10, MST] on the graph to build a similarity matrix which is sparse and thus leading to computational efficiency. We refer to the resulting method in this chapter as the neighborhood structure assisted NMF (NS-NMF). We demonstrate the benefit of the neighborhood structure assisted NMF in the task of anomaly detection. Our study will show that clustering benefits in low dimensional space and consideration of the local invariance property in obtaining those clusters make neighborhood structure assisted NMF a powerful anomaly detection method.

We want to note that two recent versions of the NMF method are closely related to what we propose in this chapter. One of them is the graph regularized NMF [43, GNMF], which regularizes

the original NMF formulation using a Laplacian matrix. Different from the proposed neighborhood structure assisted NMF, GNMF constructs the similarity matrix based on simple Euclidean distances. Our numerical testing shows furthermore that GNMF, when employed for anomaly detection, is rather sensitive to two of its tuning parameters: the number of nearest neighbors and a regularization parameter. By contrast, the neighborhood structure assisted NMF does not need the neighborhood parameter and its detection outcome appears to be much less sensitive to its regularization parameter. The second NMF variant is the symmetric NMF [42, SNMF], which uses only the similarity information while excluding the attribute information to generate the low rank matrices. In the absence of the attribute information, SNMF depends on a dense pairwise similarity measure which leads to computational disadvantage unlike MST. By abandoning the original attribute information in its formulation, SNMF makes its detection outcomes less interpretable or 'meaningful' than NS-NMF or GNMF. Comparing the neighborhood structure assisted NMF with other versions of NMF (plain NMF, GNMF and SNMF) over 20 benchmark datasets shows a clear and evident advantage of the proposed method.

To root our development in the background of anomaly detection, we would like to note that anomaly detection is related naturally to *clustering*, as researchers argue that detecting anomalies is to separate the data points into two classes—normal or regular versus abnormal, irregular, or anomalous [67, 22, 23, 68, 69, 25]. NMF based (or rather, all MF based) methods fall under the umbrella of clustering-based approaches as they also try to group the data in the low dimensional feature space, which appears to be in line with the subspace-based methods advocated for anomaly detection [30, 32, 46, 31, 33, 34].

The major contributions of our research reported here can be summarized as follows. First, we propose and design a new objective function, which incorporates neighborhood similarity information, to be used in the NMF framework for conducting data dimension reduction and data grouping. This neighborhood structure assisted NMF is the central piece in our subsequent anomaly detection procedure. Second, we compare the formulation and properties of the neighborhood structure

assisted NMF with other variants of NMF (GNMF and SNMF) and highlight the similarities and differences between them. This provides an in-depth understanding of how the neighborhood structure assisted NMF makes a difference in anomaly detection. Last but not the least, we provide both offline and online algorithmic implementations of the proposed neighborhood structure assisted NMF method to enhance its practical usefulness.

The rest of the chapter unfolds as follows. Section 3.2 describes the detailed formulation of the neighborhood structure assisted NMF. Section 3.3 discusses the similarities and differences among the proposed approach, GNMF and SNMF. Section 3.4 presents the proposed NS-NMF algorithm in a structured way for both offline and online versions. Section 3.5 compares the proposed NS-NMF method with other NMF variants on 20 benchmark datasets. We also apply these methods to a hydropower dataset. Finally, we summarize the chapter in Section 3.6.

3.2 Neighborhood structure assisted NMF and its application in anomaly detection

Anomaly detection is by and large an unsupervised learning problem as the class labels of data records are unknown in the training set and one has to depend on the structure of the data to flag a potential anomaly. To differentiate the anomalies from the normal background one may need to solve two problems. The first is to find the appropriate local contexts/communities as latent feature groups and their respective members, and the second is to extract the standard characteristics of these communities in terms of the original features so that they can serve as a basis for comparison. We use these communities to compare and judge all the data points for flagging and short listing potential anomalies. NMF apparently provides an effective solution to both of these problems. However, as we have pointed out earlier, the traditional NMF framework does not take into consideration the neighborhood structure of the data points while doing clustering in latent space and thereby produce unsatisfactory results. Also, approximating the neighborhood structure in the presence of nonlinear manifold is not straightforward either. In this section, we try to bridge this knowledge gap by proposing a graph-based approach to approximate the neighborhood

structure and develop a new neighborhood structure assisted NMF formulation for anomaly detection. The formulation considers the local invariance property while obtaining the low dimensional representation and thus should intuitively work better than the competing approaches.

3.2.1 Basic NMF framework

In NMF [38], a data matrix $\mathbf{A} \in \mathbb{R}_{+}^{m \times p}$, of which columns and rows represent the attributes and observations respectively, is factorized into two low rank matrices, namely, $\mathbf{W} \in \mathbb{R}_{+}^{m \times K}$ and $\mathbf{H} \in \mathbb{R}_{+}^{K \times p}$, such that the inner product of these factorized matrices approximate the original data matrix. Here, K represents the number of latent feature groups and it is required to be equal or less than the smaller of m and p. NMF tries to project the data with high dimensional features into a low dimensional latent space so that the original observations can be seen as a weighted linear combination of the newly formed basis vectors corresponding to each latent feature group. The rows of $\mathbf{H}$, each of which is a $1 \times p$ vector, represent the basis vectors, whereas the weights come from $\mathbf{W}$. For example, any row i from the original matrix can be reconstructed through (3.1) below:

$$\mathbf{a}_i = \sum_{k=1}^{K} W_{ik} \mathbf{h}_k, \tag{3.1}$$

where $\mathbf{a}_i$ represents the i^{th} row of $\mathbf{A}$, $\mathbf{h}_k$ represents the k^{th} row of $\mathbf{H}$, and W_{ik} is the $(i,k)^{th}$ element of $\mathbf{W}$.

To solve for the factored matrices, one needs to minimize the Frobenius norm of the difference between the original data matrix and the inner product of the factored matrices, as shown in (3.2).

$$\min_{\mathbf{W} \geqslant \mathbf{0}, \mathbf{H} \geqslant \mathbf{0}} \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 . \tag{3.2}$$

where $\| \cdot \|_F$ denotes the Frobenius norm and the constraints, $\mathbf{W} \geqslant \mathbf{0}$ and $\mathbf{H} \geqslant \mathbf{0}$, mean that both

$\mathbf{W}$ and $\mathbf{H}$ are nonnegative matrices.

3.2.2 Capturing neighborhood structure using MST

NMF finds its way in solving clustering problems when [70] show that NMF can be made equivalent to clustering approaches by formulating in a different way. In other words, it turns out that if properly formulated, NMF could have the advantages of regular clustering frameworks. As mentioned earlier, while doing clustering, NMF does not consider neighborhood structure information. But the neighborhood structure information should have been considered for the benefit of clustering or detection, as structurally similar observations in the original space ought to maintain the similarity in the latent space. To tackle this problem, we propose to extract the neighborhood structure information from the original data matrix via the modeling of a minimum spanning tree and then incorporate the structure information during the calculation of the NMF factored matrices, so as to make the resulting method more suitable for anomaly detection.

To discover the intrinsic structure in data, a popular undertaking is to form a graph object using the original data matrix $\mathbf{A}$. Each observation is represented by a node, which is connected with other nodes through a weighted edge with the weight being the pairwise Euclidean distance between them. A simple graph like this has its disadvantage—the similarity matrix thus generated would be too dense to be incorporated in the NMF setting for large datasets. In our treatment, we instead use the MST for constructing the similarity matrix among data points and doing so leads to a sparse similarity matrix.

Once we apply MST on the graph object resulting from the original data matrix $\mathbf{A}$ having m observations, what we get is a square matrix, $\mathbf{Y} \in \mathbb{R}_{+}^{m \times m}$, showing the pairwise connectedness and distance. A strictly positive value in $\mathbf{Y}$ represents the distance between two connected nodes in the resulting MST, whereas a zero indicates the disconnection between the two nodes. Different from the complete graph, $\mathbf{Y}$ of MST is supposed to be a sparse matrix, rather than a dense matrix. We further convert it into a pairwise similarity matrix, $\mathbf{S}$, by inverting only the positive entries of

$\mathbf{Y}$, as following:

$$S_{ij} = \begin{cases} \frac{1}{Y_{ij}}, & \text{if } Y_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases}$$

where S_{ij} and Y_{ij} are, respectively, the $(i,j)^{th}$ elements of $\mathbf{S}$ and $\mathbf{Y}$. This matrix is called a similarity matrix because a high value in $\mathbf{S}$ is the result of two nodes close to each other in the resulting MST, implying their similarity and likely the same cluster membership. On the other hand, a zero value means that two nodes are not directly connected and less likely similar to each other. Whether or not they may still belong to the same cluster depends largely on the two nodes' association with the common neighbors. Understandably, if two points have very low similarity and do not have connection through any common neighbors, they most probably belong to two separate clusters. Creating the similarity matrix is because we intend for $\mathbf{S}$ to guide the basic NMF process to group the similar observations into the same cluster, obtain the proper cluster centroids in the form of basis vectors, and subsequently use the cluster centroids in the action of anomaly detections.

3.2.3 Proposed NS-NMF formulation

By taking into account both the original attribute matrix $\mathbf{A}$ and the MST based neighborhood similarity matrix $\mathbf{S}$, we propose the following NS-NMF formulation to obtain the low rank factored matrices $\mathbf{W}$ and $\mathbf{H}$:

$$\min_{\mathbf{W}\geqslant\mathbf{0},\,\mathbf{H}\geqslant\mathbf{0}} \left\|\mathbf{S} - \mathbf{W}\mathbf{W}^T\right\|_F^2 + \alpha \left\|\mathbf{A} - \mathbf{W}\mathbf{H}\right\|_F^2 + \gamma(\|\mathbf{W}\|_F^2 + \|\mathbf{H}\|_F^2), \tag{3.3}$$

where $\|\mathbf{W}\|_F^2$ and $\|\mathbf{H}\|_F^2$ are the regularization terms added to the objective function to avoid overfitting, γ is the regularization parameter controlling the extent of overfitting. The first term is apparently the newly added structure similarity term, whereas the second term is the original NMF cost function, and the parameter α is used to tradeoff between these two cost functions and takes values between 0 and 1.

The merit of NS-NMF is to make sure that data points coming from the same background or from the same neighborhood are grouped into the same cluster. NMF in its original setting only tries to find the basis vectors that best approximate the data. While NMF can achieve successes by doing so in the presence of simple Euclidean structure, it does not appear effective in the presence of complicated intrinsic data structure. In those cases, NMF may achieve the data approximation objective but the clustering assignments may not be appropriate. But making proper clustering assignments is important for anomaly identification. As we see in (4.10), factoring out the right $\mathbf{h}_k$ is the key piece for computing the anomaly score and plays a decisive role in the subsequent anomaly detection.

The MST-based neighborhood structure approximates the geodesic distance between data instances via the multi-hop edge connection on the tree graph, which are considered a much better representation of complicated data structures and similarity between data instances than simple Euclidean distances [47, 9, 8]. The geodesic distance measures the minimum possible distance between two points in a curved space (like the surface of the earth), or more generally speaking, in a structured space. By incorporating the MST structural similarity measure in the NMF, the resulting method produces better clustering assignments with respective cluster centroids extracted from $\mathbf{H}$; doing so, we believe, helps substantially the mission of anomaly detection.

3.2.4 Local anomaly detection

Now, let us take a look at how the proposed NS-NMF can help us in distinguishing anomalies from the normal observations. The low rank factored matrices generated from NS-NMF provide us with the information we are seeking to solve the two problems discussed at the beginning of this section pertinent to anomaly detection. Each entry of $\mathbf{W}$ measures the extent of an observation's association with all K latent groups/clusters, whereas a row of $\mathbf{H}$ represents the average characteristics of one of the latent groups/clusters. To measure how an observation is deviating from its local context/group/community, we measure the Euclidean distance between the observation's original

attribute values and the average characteristics of that local community to which it belongs to. In other words, we create an anomaly score for observation i, which is calculated as in (4.10):

$$AS_i = \|\mathbf{a}_i - \mathbf{h}_k\|_2 \quad \text{when observation } i \text{ belongs to local community } k. \tag{3.4}$$

We repeat this process for all observations and rank the scores, $\{AS_i, i = 1, \ldots, m\}$, in descending order.

In the practice of anomaly detection, it is common that once applied to the data, a method ranks the top N instances as potential anomalies and treats the rest of data instances as normal instances. One main reason for such a decision process is that unsupervised anomaly detection methods tend to have a lower detection capability and higher false alarm rate. As a result, unsupervised detection methods are typically used as a screening tool, flagging potential anomalies to be further analyzed by either a human operator or some more expensive procedure. A cut-off is therefore used to ensure the subsequent, more expensive or time consuming step practical and feasible. In this chapter also, we follow this practice to declare the observations with top N scores as anomalies where the cut-off threshold N is prescribed based on the cost/feasibility considerations.

3.3 NS-NMF relative to other NMFs

In this section, we want to highlight the similarities and contrast of the proposed NS-NMF with some of the related approaches, principally GNMF and SNMF so that the readers get a better understanding of how the proposed method made the difference.

First, we present the formulations of the three methods in (3.5)–(3.7). To capture the essence of the NS-NMF method, we rewrite our original formulation in (4.9) by ignoring the third component of the objective function, because the third term is a regularization term to avoid overfitting, and as such, having it or not does not change the essence concerning which piece of information is used in the matrix factorization. The new formulation of NS-NMF in (3.7) now has two terms. We also

change the position of α for easiness of comparison.

$$\text{GNMF} : \min_{\mathbf{W},\mathbf{H}\geqslant\mathbf{0}} \lambda \cdot \text{tr}(\mathbf{W}^T\mathbf{L}\mathbf{W}) + \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 . \tag{3.5}$$

$$\text{SNMF} : \min_{\mathbf{W}\geqslant\mathbf{0}} \left\|\mathbf{S} - \mathbf{W}\mathbf{W}^T\right\|_F^2 . \tag{3.6}$$

$$\text{NS-NMF} : \min_{\mathbf{W},\mathbf{H}\geqslant\mathbf{0}} \frac{1}{\alpha} \cdot \left\|\mathbf{S} - \mathbf{W}\mathbf{W}^T\right\|_F^2 + \|\mathbf{A} - \mathbf{W}\mathbf{H}\|_F^2 . \tag{3.7}$$

Conceptually and formulation-wise, GNMF is the closest to the proposed NS-NMF. We notice that both NS-NMF and GNMF formulations have the same second component. This second component comes from the original NMF formulation and is used to obtain two factor matrices from the original data matrix, one of which is the attribute matrix.

Admittedly, the authors of GNMF are the first to shed light on the necessity of considering neighborhood similarity information in the original NMF process. According to [43], the low rank approximation should be obtained as in (3.5), rather than in (3.2), in order to incorporate the neighborhood similarity information. The specific mechanism of incorporating the neighborhood similarity information in (3.5) is through the use of a graph Laplacian matrix, denoted by $\mathbf{L}$. The graph Laplacian matrix $\mathbf{L}$ can be obtained by $\mathbf{L} = \mathbf{T} - \tilde{\mathbf{S}}$, where $\mathbf{T}$ is a diagonal matrix also known as degree matrix [43, 70] and $\tilde{\mathbf{S}}$ is the adjacency matrix. The adjacency matrix is calculated using the Euclidean structured neighborhood information after converting the original data into a graph object. The model has two main parameters, namely, q, the number of the nearest neighbors to be specified in order to form the similarity matrix, and λ, the regularization parameter. The value of λ can take any nonnegative value. When it takes the value of zero, the formulation ignores the neighborhood similarity completely and GNMF reduces to the original NMF.

SNMF, on the other hand, promotes the idea that one should probably consider the neighborhood similarity information only while obtaining the factored matrices. The factorization of the similarity matrix $\mathbf{S}$ generates a clustering assignment matrix $\mathbf{W}$ that is also constrained to be non-

negative. The authors of SNMF believe that doing so captures the inherent cluster structure from the graph representation of the original data matrix. [42] argue, and present case studies in support of, that the traditional NMF is not ideal for handling datasets with nonlinear structures. Compared to the plain NMF, SNMF can deal with complicated patterns and generate more accurate clustering assignments. [42] also point out that the SNMF formulation is in fact equivalent to some graph clustering methods including spectral clustering [70]. [42] present an example to show that adding the non-negativity constraints help the clustering objective and also that SNMF performs more robustly compared to other clustering methods, as SNMF does not depend on the eigenvalue structure which tends to provide inaccurate results if certain conditions are violated.

SNMF formulation takes basically the first component of the NS-NMF formulation, because SNMF ignores the original NMF portion and uses the similarity information alone. SNMF focuses only on the accuracy of clustering assignments but for the mission of anomaly detection we need something more. What we need is to have a basis for comparison, so that we can pinpoint anomalies by looking at their deviation from the average characteristics of associated cluster. The second portion of the NS-NMF formulation helps us obtain a cluster centroid matrix which summarizes the attribute distribution of the latent feature groups, thereby providing the basis for detecting the anomalies from the clusters. For this reason, NS-NMF, which keeps the second term, is more meaningful for anomaly detection than SNMF. Even the first term, although appears the same in both SNMF and NS-NMF formulations, is not really the same, because the respective similarity matrix $\mathbf{S}$ used therein is different. SNMF uses the traditional Euclidean distance-based similarity metric considering the full graph, while that in NS-NMF comes from an MST. The similarity matrix in SNMF is too dense, making SNMF to suffer in case of approximating complex structures. Unsurprisingly, SNMF is computationally more expensive than NS-NMF.

Coming back to GNMF, which has the same second term as NS-NMF and poised to be more suitable for anomaly detection. The first term in both NS-NMF and GNMF formulations has a strong connection. It can be shown that when using the same similarity matrix $\mathbf{S}$, GNMF and

NS-NMF can be made (nearly) equivalent.

To facilitate the understanding of this connection, let us consider adding an orthogonality constraint on $\mathbf{W}$. This is not exactly required in the original formulations but [70] shows that minimizing $\left\|\mathbf{S}-\mathbf{W}\mathbf{W}^T\right\|_F^2$ retains the orthogonality of $\mathbf{W}$ approximately. Suppose that the symmetric normalized Laplacian matrix is used, i.e., the original $\mathbf{L}$ is pre- and post-multiplied by $\mathbf{T}^{-\frac{1}{2}}$, then we have

$$\mathbf{L}=\mathbf{I}-\mathbf{T}^{-\frac{1}{2}}\tilde{\mathbf{S}}\mathbf{T}^{-\frac{1}{2}},$$

where without ambiguity, we still use $\mathbf{L}$ to denote the symmetric normalized Laplacian matrix. Furthermore, denote by $\mathbf{S}=\mathbf{T}^{-\frac{1}{2}}\tilde{\mathbf{S}}\mathbf{T}^{-\frac{1}{2}}$ the newly generated normalized similarity/adjacency matrix. As such, the first term in GNMF can be made equivalent to that of NS-NMF by considering the following minimization formulation.

$$\begin{aligned}\min_{\mathbf{W}\geqslant\mathbf{0}}\mathrm{tr}(\mathbf{W}^T\mathbf{L}\mathbf{W}) &= \min_{\mathbf{W}\geqslant\mathbf{0}}\mathrm{tr}(\mathbf{W}^T(\mathbf{I}-\mathbf{T}^{-\frac{1}{2}}\tilde{\mathbf{S}}\mathbf{T}^{-\frac{1}{2}})\mathbf{W})\\ &= \min_{\mathbf{W}\geqslant\mathbf{0}}\mathrm{tr}(\mathbf{W}^T(\mathbf{I}-\mathbf{S})\mathbf{W})\\ &= \min_{\mathbf{W}\geqslant\mathbf{0}}\mathrm{tr}(\mathbf{W}^T\mathbf{W})-\mathrm{tr}(\mathbf{W}^T\mathbf{S}\mathbf{W})\\ &= \min_{\mathbf{W}\geqslant\mathbf{0};\mathbf{W}^T\mathbf{W}=\mathbf{I}}\mathrm{tr}(\mathbf{I})-\mathrm{tr}(\mathbf{W}^T\mathbf{S}\mathbf{W}).\end{aligned} \tag{3.8}$$

At the last expression, we use the orthogonality constraint on $\mathbf{W}$, as mentioned above. Minimizing (3.8) with respective to a nonnegative $\mathbf{W}$ does not change its minimization outcome if we add a term that does not depend on $\mathbf{W}$ and multiply the $\mathbf{W}$-depending term by a constant. Let us add one term, $\mathrm{tr}(\mathbf{S}^T\mathbf{S})$, to the last expression of (3.8) and multiply $\mathrm{tr}(\mathbf{W}^T\mathbf{S}\mathbf{W})$ by two. As such, we

end up with an equivalent minimization problem as follows:

$$\min_{\mathbf{W}\geqslant\mathbf{0}} \text{tr}(\mathbf{W}^T\mathbf{L}\mathbf{W}) \quad \text{is equivalent to} \quad \min_{\mathbf{W}\geqslant\mathbf{0};\mathbf{W}^T\mathbf{W}=\mathbf{I}} \text{tr}(\mathbf{I}) - 2\text{tr}(\mathbf{W}^T\mathbf{S}\mathbf{W}) + \text{tr}(\mathbf{S}^T\mathbf{S})$$

$$= \min_{\mathbf{W}\geqslant\mathbf{0};\mathbf{W}^T\mathbf{W}=\mathbf{I}} \text{tr}[(\mathbf{S} - \mathbf{W}\mathbf{W}^T)^T(\mathbf{S} - \mathbf{W}\mathbf{W}^T)] \qquad (3.9)$$

$$= \min_{\mathbf{W}\geqslant\mathbf{0};\mathbf{W}^T\mathbf{W}=\mathbf{I}} \left\|\mathbf{S} - \mathbf{W}\mathbf{W}^T\right\|_F^2 .$$

The above derivation makes it apparent that if one uses the same similarity matrix $\mathbf{S}$ in both GNMF and NS-NMF formulations, then GNMF is practically the same as NS-NMF. Of course, which $\mathbf{S}$ to use creates the difference between GNMF and NS-NMF. GNMF uses only a fixed small subset of the neighborhood/adjacency information to obtain a Euclidean distance based similarity matrix and then convert it to a graph Laplacian form. A prescribed neighborhood size, q, is one of the parameters used in GNMF. NS-NMF's similarity matrix, on the other hand, is based on an MST and differs from that of GNMF. NS-NMF does not need the neighborhood size parameter, due to its use of MST. Both methods use a regularization parameter, which is $\alpha\,(0-1)$ in NS-NMF and $\lambda\,(\geq 0)$ in GNMF; these regularization parameters are in fact equivalent. Our numerical analysis shows that GNMF is sensitive to both of its parameters, q and λ, while the NS-NMF is reasonably less sensitive to its parameter α. We believe that this is a benefit of using the MST-based similarity matrix.

3.4 Algorithmic implementation of NS-NMF

In this section, we discuss the implementation of the formulation proposed in Section 3.2 in details. We provide two algorithmic procedures, one is an accelerated offline implementation, i.e., which processes all the observations at the same time to evaluate their anomalousness and another is an online implementation, i.e., which processes observations one at a time for real-time anomaly detection.

3.4.1 Accelerated offline implementation

A number of algorithms have already been proposed to solve the original NMF problem and its variants. However, most of them are computationally expensive and not ideal for handling the big data scenario. In this chapter, we choose to utilize a distributed version of the stochastic gradient descent (SGD) algorithm [41, 71] which enables us to achieve an accelerated and parallel optimization scheme. In the traditional SGD, one updates the parameters at each round by going through a single training point at a time, whereas in the distributed version, we update the parameters by processing multiple independent blocks of training data in parallel and thereby taking the computational advantage. Here, independence means that parameter update of one block will not affect the parameter update of any other block, this property is also known as the interchangeable property [41, 71]. To design a distributed SGD, we define a loss function as in (4.4), which is just the block-wise summation of the objective function defined in (4.9)

$$\begin{aligned} L_{\mathbf{S},\mathbf{A}}(\mathbf{W},\mathbf{H}) &= \sum_{\{i,j,k\}} \Big(\left\| \mathbf{S}^{ij} - \mathbf{W}^i \mathbf{W}^{j^T} \right\|_F^2 + \frac{\gamma}{2B} \left\| \mathbf{W}^i \right\|_F^2 + \frac{\gamma}{2B} \left\| \mathbf{W}^j \right\|_F^2 \\ &+ \frac{\alpha}{2} \left\| \mathbf{A}^{ik} - \mathbf{W}^i \mathbf{H}^{k^T} \right\|_F^2 + \frac{\alpha}{2} \left\| \mathbf{A}^{jk} - \mathbf{W}^j \mathbf{H}^{k^T} \right\|_F^2 + \frac{\gamma}{B} \left\| \mathbf{H}^k \right\|_F^2 \Big) \\ &= \sum_{\{i,j,k\}} L_{i,j,k}(\mathbf{W}^i, \mathbf{W}^j, \mathbf{H}^k). \end{aligned} \tag{3.10}$$

where B represents the number of splits in each dimension and it controls the number of blocks created in $\mathbf{S}$ and $\mathbf{A}$. We essentially divide $\mathbf{S}$ and $\mathbf{A}$ into blocks and the position of each block is represented by the superscripts, i, j, k. Suppose that we have a 100×100 matrix and $B = 5$. Then we will have 5 blocks with each containing 400 (i.e., a 20×20 matrix) training points. To process and approximate a block $\mathbf{S}^{ij}$, we need to update parameter block $\mathbf{W}^i$ and $\mathbf{W}^j$. Likewise, to approximate a block of $\mathbf{A}^{ik}$, we need to update $\mathbf{W}^i$ and $\mathbf{H}^k$. As we have to process blocks from two separate matrices $\mathbf{S}$ and $\mathbf{A}$, with parameters to be updated connecting each other, we define

the blocks to be processed from these two matrices as an instance set $\{\mathbf{S}^{ij}, \mathbf{A}^{ik}, \mathbf{A}^{jk}\}$.

To achieve distributed and parallel processing, we need to process randomly generated instance sets at each round parallelly and it is possible only if they are interchangeable and as a result do not affect the resulting parameter updates $\{\mathbf{W}^i, \mathbf{W}^j, \mathbf{H}^k\}$ of one another. According to [71], the interchangeability occurs only when the superscripts of the blocks do not coincide. For example, $\{\mathbf{S}^{15}, \mathbf{A}^{13}, \mathbf{A}^{53}\}$ and $\{\mathbf{S}^{23}, \mathbf{A}^{24}, \mathbf{A}^{34}\}$ are two interchangeable instances as they have entirely different superscripts. Now, as we have defined both the loss function and the instance sets, parameter update can be calculated according to (3.11) where $\theta^{i,j,k} = \{\mathbf{W}^i, \mathbf{W}^j, \mathbf{H}^k\}$ and ϵ_t is the step size at current iteration.

$$\theta_{t+1}{}^{i,j,k} = \theta_t{}^{i,j,k} - \epsilon_t \Delta L_{i,j,k}(\theta_t{}^{i,j,k}). \tag{3.11}$$

The algorithm steps are summarized in Algorithm 3, including the construction of MST based similarity matrix, the NMF optimization procedure, and the detection of anomalies.

3.4.2 Online implementation

The offline implementation discussed above possess the advantage of parallel blockwise implementation but lacks the ability of instantaneous update and real-time anomaly score computing. It requires to see all the data at once and hence build the MST using all the instances even before the execution of the algorithm. To evaluate an observation in real-time, we need to find a way to update both weight matrix $\mathbf{W}$ and basis matrix $\mathbf{H}$ incrementally when new samples arrive in a streaming fashion. In addition, we need to make sure that, such an update will not require the entire data matrix and the MST weights to reside in the memory.

There have been quite a few efforts in developing the online version of the plain NMF [72, 73, 74, 75, 76], although comparatively fewer attempts have been made in terms of online GNMF [77]. It is so because adding geometric structures to guide the NMF process makes the online update more difficult as we need to calculate the geometric weights on the fly. In this section we layout

ALGORITHM 3: Offline implementation of NS-NMF algorithm for anomaly detection

Input : $\mathbf{A}$, α, N, γ, B, K
Output: List of anomalous nodes l_{final}

Generate a set of vertices V, where each vertex represent a separate observation from the dataset;
Construct edges by calculating Euclidean distance between each pair of vertices using their attribute values from the dataset and store them in E;
Construct a MST using V and E and generate the pairwise similarity matrix $\mathbf{S}$ from the resultant pairwise MST distance matrix;
Initialize $\mathbf{W}$ and $\mathbf{H}$ randomly;
Partition $\mathbf{S}$ and $\mathbf{A}$ and corresponding $\mathbf{W}$ and $\mathbf{H}$ into blocks;
while *not converged* **do**
    Randomly generate a collection of instance sets from blocked $\mathbf{S}$ and $\mathbf{A}$, $U = \{\{i_1, j_1, k_1\}, \{i_2, j_2, k_2\}, \{i_3, j_3, k_3\}, ...\}$ such that any two are interchangeable;
    for $(i, j, k) \in U$ *in parallel* **do**
        $\mathbf{W}'^{i} \leftarrow \mathbf{W}^{i} - \epsilon_t \Delta_{\mathbf{W}^i} L_{i,j,k}$;
        $\mathbf{W}'^{j} \leftarrow \mathbf{W}^{j} - \epsilon_t \Delta_{\mathbf{W}^j} L_{i,j,k}$ (if $i \neq j$);
        $\mathbf{H}'^{k} \leftarrow \mathbf{H}^{k} - \epsilon_t \Delta_{\mathbf{H}^k} L_{i,j,k}$;
        $\mathbf{W}^{i} \leftarrow \mathbf{W}'^{i}, \mathbf{W}^{j} \leftarrow \mathbf{W}'^{j}, \mathbf{H}^{k} \leftarrow \mathbf{H}'^{k}$;
        Non negativity projection for $\mathbf{W}^i$, $\mathbf{W}^j$ and $\mathbf{H}^k$;
    end
end
for $k = 1 : K$ *(number of latent features)* **do**
    Make the clustering assignment according to the largest entry in each row of $\mathbf{W}$;
    Calculate the local anomaly scores of the cluster members according to (4.10) and store them in AS_i ;
end
Store the accumulated list of nodes with anomaly scores from all the clusters as $l_{total} = \{AS_1, AS_2, \ldots, AS_m\}$;
Return the nodes associated with top N local anomaly scores as final anomalies in l_{final};

the online implementation of NS-NMF.

Let us assume that the observations, $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{d-1}, \mathbf{a}_d]$, are generated in a streaming fashion, where $\mathbf{a}_d$ represents the d^{th} data sample just arrived and its attribute information. Upon its arrival, the d^{th} component of the weight matrix and basis matrix need to be updated so that the instantaneous anomaly score can be calculated. For this d^{th} data sample we can write our NS-NMF objective function by ignoring the regularization component as following:

$$\mathbf{F}_d = \alpha \left\|\mathbf{A}_d - \mathbf{W}_d\mathbf{H}_d\right\|_F^2 + \left\|\mathbf{S}_d - \mathbf{W}_d\mathbf{W}_d^T\right\|_F^2,$$

which is equivalent to, (recall (3.9))

$$\begin{aligned}
&\alpha \left\|\mathbf{A}_d - \mathbf{W}_d\mathbf{H}_d\right\|_F^2 + \mathrm{tr}(\mathbf{W}_d^T\mathbf{L}_d\mathbf{W}_d) \\
&= \alpha \sum_{i=1}^{d}\sum_{j=1}^{p}((\mathbf{A}_d)_{ij} - (\mathbf{W}_d\mathbf{H}_d)_{ij})^2 + \sum_{k=1}^{K}\sum_{i=1}^{d}\sum_{j=1}^{d}(\mathbf{W}_d^T)_{ki}(\mathbf{L}_d)_{ij}(\mathbf{W}_d)_{jk} \\
&= \left[\alpha \sum_{i=1}^{d-1}\sum_{j=1}^{p}((\mathbf{A}_{d-1})_{ij} - (\mathbf{W}_{d-1}\mathbf{H}_d)_{ij})^2 + \sum_{k=1}^{K}\sum_{i=1}^{d-1}\sum_{j=1}^{d-1}(\mathbf{W}_{d-1}^T)_{ki}(\mathbf{L}_d)_{ij}(\mathbf{W}_{d-1})_{jk}\right] \\
&+ \left[\alpha \sum_{j=1}^{p}((\mathbf{a}_d)_j - (\mathbf{w}_d\mathbf{H}_d)_j)^2 + \sum_{k=1}^{K}\sum_{i=1}^{d-1}(\mathbf{W}_d^T)_{ki}(\mathbf{L}_d)_{id}(\mathbf{w}_d)_k + \sum_{k=1}^{K}\sum_{j=1}^{d-1}(\mathbf{w}_d^T)_k(\mathbf{L}_d)_{dj}(\mathbf{W}_d)_{jk}\right. \\
&\left. + \sum_{k=1}^{K}(\mathbf{w}_d^T)_k(\mathbf{L}_d)_{dd}(\mathbf{w}_d)_k\right] \\
&= \mathbf{F}_{d-1} + \mathbf{f}_d.
\end{aligned} \tag{3.12}$$

Apparently, the NS-NMF objective function in (3.12) is divided into two parts—$\mathbf{F}_{d-1}$ denotes the cost up to the $(d-1)^{th}$ sample and $\mathbf{f}_d$ denotes the cost of the d^{th} sample, whereas $\mathbf{w}_d$ and $\mathbf{a}_d$ denote, respectively, the last row of $\mathbf{W}_d$ and $\mathbf{A}_d$. This strategy is known as the incremental NMF in the literature [78, 79, 80]. As the number of samples increases, new observations will have minor influence on the basis matrix, so that updating only the weight vector of the last sample will suffice.

In light of this thought, we can consider the first $d-1$ rows of $\mathbf{W}_d$ equal to $\mathbf{W}_{d-1}$. To compute the cost of d^{th} sample, i.e., $\mathbf{f}_d$, we need to establish an updating policy for the basis matrix component $(\mathbf{H}_d)_{kj}$ and weight matrix component $(\mathbf{w}_d)_k$. We can utilize the gradient descent approach to derive the update policy as following:

$$(\mathbf{w}_d)_k^{t+1} = (\mathbf{w}_d)_k^t - \delta_k \frac{\partial \mathbf{F}_d}{\partial (\mathbf{w}_d)_k^t}, \tag{3.13}$$

$$(\mathbf{H}_d)_{kj}^{t+1} = (\mathbf{H}_d)_{kj}^{t} - \theta_{kj}\frac{\partial \mathbf{F}_d}{\partial (\mathbf{H}_d)_{kj}^{t}}. \tag{3.14}$$

In (3.13) and (3.14), δ and θ denote the step sizes, t denotes the iteration number, $k = 1, 2, \ldots, K$, and $j = 1, 2, \ldots, p$. The step sizes are chosen as in (3.15) and (3.16), following the work in [43] and [72]:

$$\delta_k = \frac{(\mathbf{w}_d)_k^t}{2(\alpha((\mathbf{w}_d)^t \mathbf{H_d}^t (\mathbf{H_d}^t)^T + \sum_{i=1}^{d}(\mathbf{T}_d)_{id}\mathbf{w}_i^t)_k}, \tag{3.15}$$

$$\theta_{kj} = \frac{(\mathbf{H}_d)_{kj}^t}{2\alpha(\mathbf{W}_{d-1}^T \mathbf{W}_{d-1}\mathbf{H}_d^t + (\mathbf{w}_d^{t+1})^T \mathbf{w}_d^{t+1}\mathbf{H}_d^t)_{kj}}. \tag{3.16}$$

Substituting (3.15) and (3.16) into (3.12) and (3.13), we obtain the updating policy as follows:

$$(\mathbf{w}_d)_k^{t+1} = (\mathbf{w}_d)_k^t \frac{(\alpha a_d (\mathbf{H}_d^t)^T + \sum_{i=1}^{d-1}(\mathbf{S}_d)_{id}\mathbf{w}_i^t)_k}{(\alpha(\mathbf{w}_d)^t \mathbf{H}_d^t (\mathbf{H}_d^t)^T + (\mathbf{T}_d)_{dd}\mathbf{w}_d^t)_k}, \tag{3.17}$$

$$(\mathbf{H}_d)_{kj}^{t+1} = (\mathbf{H}_d)_{kj}^t \frac{(\mathbf{W}_{d-1}^T \mathbf{A}_d + (\mathbf{w}_d^{t+1})^T \mathbf{a}_d)_{kj}}{(\mathbf{W}_{d-1}^T \mathbf{W}_{d-1}\mathbf{H}_d^t + (\mathbf{w}_d^{t+1})^T \mathbf{w}_d^{t+1}\mathbf{H}_d^t)_{kj}}, \tag{3.18}$$

where $\mathbf{S}_d$ and $\mathbf{T}_d$ represent, respectively, the MST based weight matrix and degree matrix.

If we look at (3.18), it requires all the data samples before the d^{th} one to compute the update but doing so will significantly increase the memory requirements. To overcome the problem, we can use the strategy of cumulative summation. Let us first introduce two variables $\mathbf{U}_d$ and $\mathbf{V}_d$ with initial values $\mathbf{U}_0 = \mathbf{V}_0 = \mathbf{0}$. Then using (3.19) and (3.20), we can rewrite the update policy for $\mathbf{H}_d$ as in (3.21) which now no longer needs to memorize all the samples.

$$\begin{aligned}\mathbf{U}_d &= \sum_{i=1}^{d} \mathbf{w}_i^T \mathbf{w}_i \\ &= \mathbf{U}_{d-1} + \mathbf{w}_d^T \mathbf{w}_d\end{aligned} \tag{3.19}$$

$$\begin{aligned}\mathbf{V}_d &= \sum_{i=1}^{d} \mathbf{w}_i^T \mathbf{a}_i \\ &= \mathbf{V}_{d-1} + \mathbf{w}_d^T \mathbf{a}_d\end{aligned} \tag{3.20}$$

$$(\mathbf{H}_d)_{kj}^{t+1} = (\mathbf{H}_d)_{kj}^{t} \frac{(\mathbf{V}_d)_{kj}}{\mathbf{U}_d (\mathbf{H}_d^t)_{kj}} \tag{3.21}$$

Another problem with (3.17) is that in order to update $(\mathbf{w}_d)_k$, we need to reconstruct the MST with all the data samples every time as a new sample comes in. Again, doing so slows down the online operation. To address this problem, we adopt the combination of local MST [81] and buffering strategy [82, 77]. In a local MST, for each observation we construct an MST with its neighbors only. These neighbors can be chosen in a temporal fashion. In other words, the neighbors of the d^{th} sample (which just arrives) are the samples having arrived in a specified time window before it, say, in the window of dating back to time instance $d - z$, where z is the size of the time window. If z =50, it means that the MST will be constructed based on the d^{th} sample and the most recent 50 samples arrived before the d^{th} sample. The buffering strategy states that instead of discarding all the old data samples, one maintains a buffer of specified size, Q. After the buffer is full for the first time, any new sample will be added to the buffer while the buffer drops the oldest one and thereby keeps its size the same. To connect the two approaches, we set the time window size the same as the buffer size, i.e., $Q = z$. Consequently, we can rewrite the updating policy for $(\mathbf{w}_d)_k$ as

$$(\mathbf{w}_d)_k^{t+1} = (\mathbf{w}_d)_k^{t} \frac{(\alpha \mathbf{a}_d (\mathbf{H}_d^t)^T + \sum_{i=d-z}^{d-1} (\mathbf{S}_d)_{id} \mathbf{w}_i^t)_k}{(\alpha (\mathbf{w}_d)^t \mathbf{H}_d^t (\mathbf{H}_d^t)^T + (\mathbf{T}_d)_{dd} \mathbf{w}_d^t)_k}. \tag{3.22}$$

The algorithm steps are summarized in Algorithm 4. There are three phases of the algorithm, after the initialization, we proceed to these phases. Steps 5–8 summarizes the first phase, i.e., $d < z$, where the new samples are added along with initialization of the weight vectors. When the buffer is full for the first time, i.e., $d = z$, it starts the second phase, in which an MST is constructed and the offline NS-NMF algorithm helps obtain the weights and basis vectors. Step 9–13 summarizes this phase. After that, the algorithm enters the third and final phase, i.e., $d > z$, where the update of the weight and basis vectors and calculation of the anomaly scores are carried out from Steps 14–23. In this phase, a gradient descent approach is used to obtain the updated weight and basis values for each new sample. Steps 16–20 summarizes the iterations on t, which are required for the convergence of gradient descent approach.

Similar to the offline counterpart, the assignment of an observation to the k^{th} cluster is made according to the largest entry in $\mathbf{w}_d$. After that, we can compute the anomaly score of the d^{th} observation as in (3.23). Now, we can either choose a threshold and mark the observation as anomaly on the fly if its anomaly score crosses the threshold or we can store the anomaly scores to do the evaluation later. In this work, we test both of the options. For the benchmark datasets, which do not have any timestamps marking, we decide to go for the second option. What this means is that while we run the online algorithm to get the anomaly scores on the fly from the streaming data, the declaration of anomaly is again based on selecting the top N scores as anomalies, same as we do in our offline scenario. On the other hand, for the hydropower dataset, we do have the associated timestamps and therefore we know the order of the observations. So, we decide to go with the first option and mark anomalies on the fly.

$$AS_d = \|\mathbf{a}_d - (\mathbf{H}_d)_k\|_2 \quad \text{when observation } d \text{ belongs to local community } k. \tag{3.23}$$

ALGORITHM 4: Online implementation of NS-NMF algorithm for anomaly detection

Input : Current observation, $\mathbf{a}_d$, trade-off parameter, α, buffer size, z, and the number of clusters, K

Output: Current basis matrix, $\mathbf{H}_d$, and anomaly score, AS_d

Initialize $\mathbf{H}_0$ with random values;
Initialize $\mathbf{U}_0 = \mathbf{V}_0 = \mathbf{0}$;
Initialize $\mathbf{A}_0 = \mathbf{W}_0 = \mathbf{S}_0 = \phi$;
while *a new observation* $\mathbf{a}_d$ *arrives* **do**
    Draw the current sample $\mathbf{a}_d$;
    Initialize weight coefficient $\mathbf{w}_d$ with random values;
    Append $\mathbf{a}_d$ to $\mathbf{A}_{d-1}$ and assign it to $\mathbf{A}_d$;
    Append $\mathbf{w}_d$ to $\mathbf{W}_{d-1}$ and assign it to $\mathbf{W}_d$;
    if $d = z$ **then**
        Construct MST using the observations in $\mathbf{A}_d$ and store the weights in $\mathbf{S}_d$;
        Apply offline NS-NMF to obtain $\mathbf{W}_d$ and $\mathbf{H}_d$;
        Calculate $\mathbf{U}_d$ and $\mathbf{V}_d$ using (3.19) and (3.20);
    end
    if $d > z$ **then**
        Construct MST using the observations in $\mathbf{A}_d$ and obtain $\mathbf{s}_d$;
        repeat
            Use (3.22) to update $\mathbf{w}_d^{t+1}$ using $\mathbf{s}_d$ and $\mathbf{H}_d$;
            Use (3.19) and (3.20) to update $\mathbf{U}_d$ and $\mathbf{V}_d$ with $\mathbf{w}_d^{t+1}$;
            Use (3.22) to update $\mathbf{H}_d^{t+1}$;
        until *Convergence*;
        Delete the first row vector from both $\mathbf{A}_d$ and $\mathbf{W}_d$;
        Calculate the anomaly score for the d^{th} observation using (3.23) and store it in AS_d
    end
end

3.5 Comparative performance analysis of NS-NMF

We want to evaluate the performance of the proposed NS-NMF method for anomaly detection compared to the original NMF as well as GNMF and SNMF. In Section 3.5.1, we apply the competitive methods to benchmark datasets. In Section 4.4.1, we apply the competitive methods to hydropower plant.

First, let us take a look at the parameter selection policies for the competing methods. The number of latent features or clusters, K is needed for all of the NMF-based methods. In this

study we use $K = 5$. We have also explored the possibility of using $K = 2, 10, 15, 20$ and 25 and found that the relative performance of the competing methods remains the same in all cases. Based on our experiments with different K's, we observe that the NMF-based anomaly detection methods perform better when K is in the range of $5 - 15$. We settle for $K = 5$ because it results in overall good detection performances for all competing methods. The cut-off value, N, required to generate the final anomaly list for both offline and online version, is taken as the number of true anomalies in the benchmark dataset studies, as discussed above. Other than these two parameters that are common to all methods, the rest of the parameters are algorithm-specific. We summarize the parameter choices in Table 3.1. For SNMF and GNMF, we adopt the best settings described in their original papers. For NS-NMF, as mentioned earlier, its performance is not sensitive to the choice of α. We settle at $\alpha = 0.8$ by conducting a few simple trials.

Table 3.1: Parameter values and settings used for NS-NMF, GNMF and SNMF.

Competing methods	Number of latent factors, K	Other settings
Offline NS-NMF	5	Parameter controlling the influence of NMF, $\alpha = 0.8$ Regularizer for controlling overfitting, $\gamma = 0.2$
Online NS-NMF	5	Parameter controlling the influence of NMF, $\alpha = 0.8$ Buffer size, $z = B = 20$
GNMF	5	Manifold regularizer, $\lambda = 100$ Neighborhood graph construction parameter, $q = 5$ Weighting scheme for adjacency matrix: $0 - 1$
SNMF	5	Gaussian similarity measure for constructing $\mathbf{S}$

3.5.1 Performance comparison using publicly available datasets

We present the comparison of detection performance of the four offline methods on the 20 benchmark datasets in Table 3.2. For this comparison, we only consider the offline version of NS-NMF because the competitors are offline, so it is a bit unfair if we compare the online version of NS-NMF. For this reason, NS-NMF means the offline NS-NMF in the comparison shown in Table 3.2,

Table 3.2: Performance comparison among the NMF approaches.

Anomaly detection methods / Performance (number of datasets)	NS-NMF	NMF	GNMF	SNMF
Better (uniquely best result)	12	0	0	2
Equal (equal to the existing best result)	6	3	4	4
Close (within 20% of the best result)	0	4	2	4
Worse (not within 20% of the best result)	2	13	14	10
Mean relative rank	1.1	3.0	2.6	2.2

Table 3.3, and Figure 3.1.

To better reflect the methods' comparative edge, we break down the comparison into four major categories in Table 3.2, namely *Better*, *Equal*, *Close* and *Worse*, as explained in the table. NS-NMF outperforms other methods by showing uniquely best detection performance on 12 out of 20 datasets and tying for the best in another six cases. Only in two cases, NS-NMF is obviously worse than the best performer. SNMF achieves the uniquely best performance in those two cases, while NMF and GNMF tie for some best performance but never beat others outright. If we rank each of these four methods in a scale of 1 (best) to 4 (worst), then the mean rank for NS-NMF is 1.1, which is far ahead of other methods, with SNMF at 2.2 mean rank, GNMF 2.6 mean rank, and NMF 3.0 mean rank.

We apply the Friedman test [55] as we did in Section 2.3. We find the p-value (4.11×10^{-6}) significant enough to reject the null hypothesis.

We also perform a post hoc analysis of the four methods in terms of their ranking performance. Fig. 3.1 presents the post hoc analysis on the ranking data and it indicates that the ranking of NS-NMF is significantly better than the other three approaches at the 0.05 level of significance. The detailed pairwise comparisons between NS-NMF and each of the three methods are presented in Table 3.3. The p-values are calculated using the Conover post-hoc test [56]. We employ the Bonferroni correction [83] to adjust the p-values for multiple comparisons. All the pairwise com-

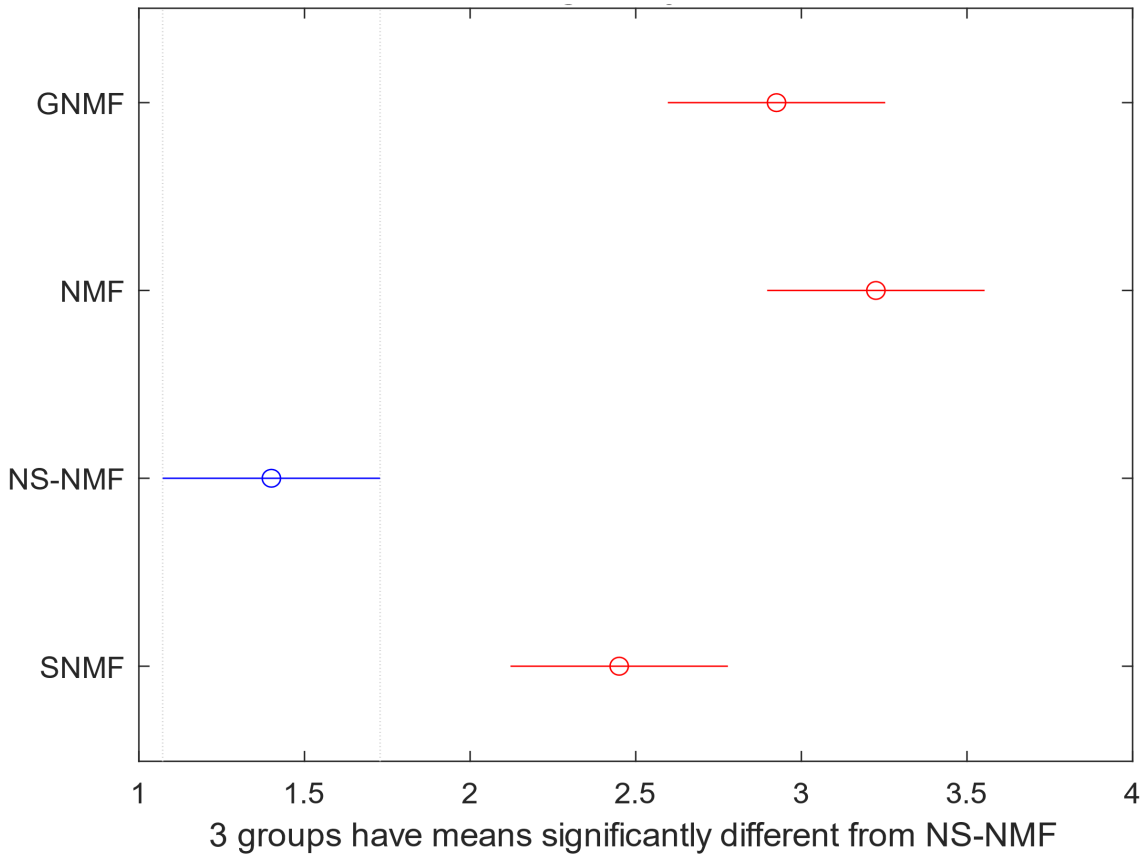


Figure 3.1: Post hoc analysis on the ranking data obtained by the Friedman test.

parisons show a sufficiently significant difference.

In Table 3.4, we summarize the number of true detections by the competing methods. We notice that the offline NS-NMF either outperforms or is no worse than NMF and GNMF in every single case. GNMF performs worse than the regular NMF in four cases. We use the best parameter setting for GNMF as recommended by its authors. We do observe the sensitive nature of GNMF to its parameters and acknowledge the possibility that some other parameter combinations might produce a better outcome. However, parameter selection in unsupervised learning settings is a difficult task, as the common approaches working well for supervised learning like cross validation does not work in the unsupervised circumstances. Therefore, the sensitivity of GNMF is certainly

Table 3.3: The p-values of pairwise comparisons between the NS-NMF method with each of the three competing methods.

Competing methods	p-value
NS-NMF vs NMF	1.64×10^{-18}
NS-NMF vs GNMF	2.86×10^{-15}
NS-NMF vs SNMF	1.20×10^{-9}

a shortcoming. SNMF's performance is better than that of NMF and GNMF, although still overall worse than the offline NS-NMF. SNMF is the slowest in terms of computational time and it does not appear scalable on big datasets. For example, in the case of the ALOI dataset that has 50,000 observations, SNMF takes almost eight hours to generate the index of the anomalous observations, whereas offline NS-NMF takes one quarter of that time to complete the task, with the majority of its time spent on MST construction. NMF and GNMF are much faster and take around 12 mins and 35 mins, respectively, in processing the same dataset.

Here we also include the results from the online version of NS-NMF because we would like to draw a comparison between the offline and online NS-NMF and see how much efficiency the online NMF maintains while using a small subset of data to compute the anomaly scores. Unsurprisingly, the offline version comes out superior in 14 out of the 20 cases. On the other hand, for most of the cases with the exception of the case of `Annthyroid`, the online version does not perform that much worse than the offline version and never comes out as the uniquely worst performer among the methods. A bit surprisingly, the online version even beats its offline counterpart in three cases. It seems to suggest that in some cases, having a longer memory and global data connection may hurt the detection. Overall, we deem the online NS-NMF a competent and promising online anomaly detection algorithm.

Table 3.4: Number of true positive detections of the competing NMF methods. Bold entries represent the best detection performance in a respective dataset.

Dataset	Offline NS-NMF	Online NS-NMF	NMF	GNMF	SNMF
WBC	**9**	7	8	8	7
Waveform	6	**31**	0	1	26
Heart	**4**	**4**	3	**4**	**4**
WDBC	**8**	6	**8**	6	7
Glass	**4**	1	0	0	1
Spambase	36	**38**	21	23	28
WPBC	**16**	11	11	11	13
Stamps	**5**	2	1	2	4
Parkinson	**3**	**3**	2	**3**	2
Lymphography	**5**	3	**5**	3	**5**
Ionosphere	**82**	68	51	38	65
PIMA	**9**	4	8	3	2
Shuttle	**4**	1	1	1	3
Cardiotocography	19	11	13	19	**33**
Arrhythmia	**5**	3	3	3	3
Page blocks	**38**	**38**	24	**38**	**38**
Pendigits	0	**1**	0	0	0
KDD	**87**	81	56	82	78
ALOI	**151**	121	91	97	84
Annthyroid	**40**	18	13	12	24

3.5.2 Application to a power plant example

The industry dataset used in this study comes from a hydropower plant. The top 100 anomalies identified by the three methods are shown in Table 3.5. We use both online and offline version of NS-NMF. To detect anomalies on the fly, we use the threshold update policy similar to [84] for the online algorithm. To save space we skip some rows in the table. We observe that altogether these three methods have 33 common anomalous time stamps among the top 100 anomalies, whereas offline NS-NMF, online NS-NMF and GNMF produce similar outcomes and share 49 common time stamps. These common findings serve as an indirect way of cross validating the sanity of the detection outcomes.

Table 3.5: Summary of the top 100 anomalies.

Offline NS-NMF	Online NS-NMF	GNMF	SNMF
7/4/2015 11:30	7/4/2015 11:50	7/4/2015 11:30	4/15/2015 16:50
7/4/2015 11:40	7/4/2015 12:00	7/4/2015 11:40	4/15/2015 18:30
7/4/2015 11:50	7/4/2015 12:20	7/4/2015 11:50	4/15/2015 20:10
7/4/2015 12:00	7/4/2015 12:30	7/4/2015 12:00	4/15/2015 22:30
7/4/2015 12:10	7/4/2015 12:50	7/4/2015 12:10	4/15/2015 22:50
7/4/2015 12:20	7/4/2015 1:00	7/4/2015 12:20	4/16/2015 2:50
7/4/2015 12:30	7/4/2015 1:30	7/4/2015 12:30	4/16/2015 4:30
.....	.....	.....	.....
9/13/2015 19:00	9/14/2015 7:40	7/4/2015 14:40	4/18/2015 23:10
9/14/2015 2:40	9/14/2015 7:50	7/8/2015 12:20	4/19/2015 4:00
9/14/2015 8:00	9/14/2015 8:00	7/8/2015 18:00	4/19/2015 21:40
9/14/2015 8:10	9/14/2015 8:10	9/15/2015 21:20	4/19/2015 23:40
9/14/2015 8:20	9/14/2015 8:20	9/15/2015 21:40	4/20/2015 8:20
9/14/2015 8:30	9/14/2015 8:30	10/3/2015 18:50	7/4/2015 11:30
9/14/2015 13:00	9/14/2015 8:50	10/3/2015 19:10	7/4/2015 11:50
9/14/2015 13:10	9/14/2015 13:00	10/3/2015 19:20	7/4/2015 12:00
.....	.....	.....	.....
10/3/2015 20:50	10/3/2015 20:20	10/3/2015 22:20	7/4/2015 15:00
10/3/2015 21:00	10/3/2015 20:30	10/3/2015 22:30	7/4/2015 15:10
10/3/2015 21:10	10/3/2015 20:40	10/3/2015 22:40	7/4/2015 15:20
10/3/2015 21:20	10/3/2015 20:50	10/3/2015 22:50	7/4/2015 15:30
10/3/2015 21:30	10/3/2015 21:30	10/3/2015 23:00	7/4/2015 15:40
10/3/2015 21:40	10/3/2015 21:50	10/3/2015 23:10	7/4/2015 15:50
10/3/2015 21:50	10/3/2015 21:20	10/3/2015 23:20	7/4/2015 16:00
.....	.....	.....	.....
10/4/2015 0:00	10/3/2015 23:40	10/4/2015 17:20	7/4/2015 18:10
10/4/2015 0:10	10/3/2015 23:50	10/4/2015 17:30	7/8/2015 12:20
10/4/2015 0:20	10/4/2015 0:00	10/4/2015 17:40	7/8/2015 15:50
10/4/2015 23:40	10/4/2015 0:10	10/4/2 015 17:50	7/8/2015 18:00
10/4/2015 23:50	10/4/2015 0:20	10/4/2015 18:00	9/17/2015 4:40
10/5/2015 1:00	10/4/2015 0:30	10/4/20 15 18:10	9/17/2015 4:50
10/5/2015 1:30	10/4/2015 22:50	10/4/2015 18:20	9/17/2015 5:00
.....	.....	.....	.....
10/5/2015 4:30	10/13/2015 17:25	10/5/2015 3:30	10/3/2015 20:40
10/5/2015 4:40	10/13/2015 17:30	10/5/2015 3:40	10/3/2015 21:50
10/5/2015 4:50	10/13/2015 17:35	10/5/2015 3:50	10/3/2015 22:00
10/13/2015 16:35	10/13/2015 17:45	10/5/2015 4:00	10/3/2015 22:10
10/13/2015 16:45	10/13/2015 18:20	10/5/2 015 4:10	10/3/2015 22:20
10/13/2015 16:55	10/13/2015 18:30	10/5/2015 4:20	10/3/2015 22:30
.....	.....	.....	.....
1/2/2016 21:20	1/11/2016 18:10	10/13/2015 17:05	10/13/2015 17:25
1/2/2016 21:30	1/12/2016 11:20	10/13/2015 17:15	10/13/2015 17:35
1/2/2016 21:40	1/12/2016 11:30	10/13/2015 17:25	10/13/2015 17:45
1/12/2016 11:20	1/12/2016 11:40	10/13/2015 17:35	10/14/2015 18:35
1/12/2016 11:30	1/12/2016 12:10	10/13/2015 17:45	1/3/2016 7:00

Apart from the common detection outcomes, we find that the anomalous time stamps belong to a few anomaly-prone days, which are listed in Table 3.6. It is also noticeable that anomalies occur in chunks, and in most cases, the observations in the close time vicinity of an anomalous time stamp are also returned as anomalies. When we report these time stamps to the operating manager, he agrees, after his own verification, that most of these findings present valid concerns and indeed require trouble shooting.

While there is a good common ground shared by these methods, the competing methods do perform differently at certain aspect. Despite the close performance between GNMF and NS-NMF, GNMF entirely misses the anomalous stamps on September 13^{th}, September 14^{th}, January 2^{nd}, and January 12^{th}. SNMF likewise misses those dates, but does detect one time stamp on January 12^{th}. The offline NS-NMF successfully identified all of those time stamps, but misses some potential anomalous time stamps in the month of April, and so does the online NS-NMF. The online NS-NMF also misses the anomalies on September 13^{th} and January 2^{nd}. SNMF successfully detects the April dates. All of them misses the January 9^{th} anomalies which are deemed as abnormal by the operating manager and his team. The operating manager also indicated that July 8^{th}, September 17^{th}, January 3^{rd} time stamps do not appear to be anomalies, after their extensive closer-look that yields no intelligible outcomes. These dates are identified as anomalous by SNMF but not by NS-NMF or GNMF. The operating manager registers the offline version of NS-NMF as the most competitive method followed by the online counter part among the competitors.

Detecting the anomalies does not tell us directly the root cause behind the abnormal behaviors. But anomaly detection outcomes can be used to assign class labels to the respective data records. A simple follow-up is to build a classification and regression tree on the labeled datasets, which can reveal which variable, or variable combination, is actually leading to these anomalous conditions. Doing so injects the interpretability to an unsupervised learning problem and can advise proper actions to address the anomalous condition and prevent future damage, disruption, or even catastrophe. An example of such exercise can be found in [81, 45] and we will not repeat it here.

Table 3.6: Most anomaly prone days identified by the three methods.

July 4th, 2015
September 13th, 2015
September 14th, 2015
October 3rd, 2015
October 5th, 2015
October 13th, 2015
October 14th, 2015
January 2nd, 2016
January 12th, 2016

3.6 Summary

In this chapter, we propose a neighborhood structure assisted nonnegative matrix factorization method and demonstrate its application in anomaly detection. We argue that in the absence of the similarity information, the original NMF basis vectors are not enough to represent and separate complicated clusters in the reduced feature space. To represent and summarize the complex data structure information in a similarity matrix, we use a minimum spanning tree to capture the neighborhood connectivity information and to approximate the geodesic distance between data instances. The current approaches that use the Euclidean distance based similarity metric is not capable of approximating the true data space structure and those approaches using the complete graphs become computationally expensive. We develop a joint optimization framework to obtain the clustering indicator and the attribute distribution matrix, and then, we devise an anomaly score to flag potential point-wise anomalies. We use a parallel block stochastic gradient descent method to compute these factored matrices for fast implementation. We also design an online algorithm to render the proposed method applicable for analyzing streaming data. The specific action of modeling the neighborhood structure appears to make an appreciable impact, as NS-NMF demonstrates clear advantage in the task of anomaly detection in an extensive empirical study using 20 benchmark datasets and one hydropower dataset.

4. GRAPH REGULARIZED AUTOENCODER FOR ANOMALY DETECTION

Dimensionality reduction is a crucial first step for many unsupervised learning tasks including anomaly detection. Autoencoder is a popular mechanism to accomplish the goal of dimensionality reduction. In order to make dimensionality reduction effective for high-dimensional data embedding nonlinear low-dimensional manifold, it is understood that some sort of geodesic distance metric should be used to discriminate the data samples. Inspired by the success of neighborhood aware shortest path based geodesic approximators such as ISOMAP, in this work, we propose to use minimum spanning tree (MST) to approximate the local neighborhood structure and generate structure-preserving distances among data points. We use this MST-based distance metric to replace the Euclidean distance metric in the embedding function of autoencoders and develop a new graph regularized autoencoder, which outperforms, over 20 benchmark anomaly detection datasets, the plain autoencoder using no regularizer as well as the autoencoders using the Euclidean-based regularizer. We furthermore incorporate the MST regularizer into two generative adversarial networks and find that using the MST regularizer improves the performance of anomaly detection substantially for both generative adversarial networks.

4.1 Introduction

Autoencoder [35, 36, 37] is a widely used tool in many unsupervised learning tasks such as clustering and anomaly detection [85, 86]. It is an efficient dimensionality reduction mechanism, converting a data matrix $\mathbf{X} \in \mathbb{R}^{m \times p}$, of which columns and rows represent the attributes and observations respectively to a dimension-reduced output $\mathbf{Z} \in \mathbb{R}^{m \times l}$, such that $l < p$, but preferably $l \ll p$. Autoencoders frame the unsupervised problem of dimensionality reduction through the use of a pair of encoder and decoder—while the encoder reduces $\mathbf{X}$ to $\mathbf{Z}$, the decoder reconstructs $\mathbf{Z}$ to an $\mathbf{X}' \in \mathbb{R}^{m \times p}$, which is of the same dimension as $\mathbf{X}$. The goal of finding the optimal low-dimensional representation $\mathbf{Z}$ is to be accomplished by designing the encoder/decoder pair to

minimize the reconstruction error between $\mathbf{X}$ and $\mathbf{X}'$. Please see the illustration in Fig. 4.1.

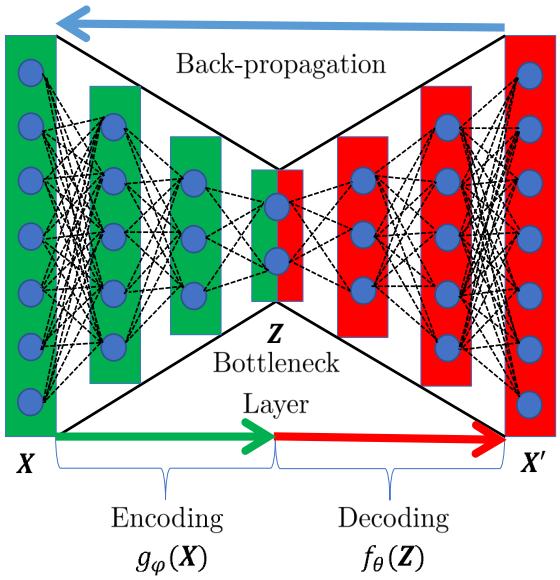


Figure 4.1: Autoencoder framework using feed-forward neural networks. Blue circles represent nodes/neurons and dotted lines represent neuron connections.

By producing the low-dimensional $\mathbf{Z}$, an autoencoder does not automatically perform clustering or anomaly detection. Yet, people argue that once a good low-dimensional representation of the high-dimensional original data is obtained, the subsequent tasks become much easier and manageable. For this reason, autoencoders are considered a key technique to address one of the challenges in unsupervised learning, especially when dimensionality reduction is inevitably necessary for achieving good performances. While the general idea of autoencoders can be materialized by various choices of encoder and decoder, the practical ones in use are almost invariably artificial neural networks (ANN), as depicted in Fig. 4.1.

Researchers recognize important benefits in using the autoencoder approach for tasks like anomaly detection. Autoencoder can handle complex data in large quantity, thanks to the current advancement in deep neural networks. With deep neural networks serving as an encoder/decoder, one can reach to the latent space through multiple steps of nonlinear transformation, which can

help unfold data with complex intrinsic structure and greatly facilitate the subsequent detection objective.

It is not surprising that the loss function characterizing the reconstruction error between $\mathbf{X}$ and $\mathbf{X}'$ plays a crucial role in a successful autoencoder design. The choice of loss function often depends on the ultimate learning tasks, e.g., binary classification, multi-class classification, or regression, and the typical choices include *cross-entropy*, *Kullback-Leibler divergence*, *mean squared error*, and *mean absolute error*. These loss functions, however, do not have any provision to preserve the neighborhood structure in the reduced representation. But preserving the neighborhood structure while reducing the data dimension is demonstrably critical and in fact a key requirement studied in the nonlinear embedding research [87, 88, 7, 89, 90, 43].

Dimensionality reduction problems are closely related to the manifold approximation or intrinsic structure recovery problem. According to the manifold hypothesis [6, 91], high dimensional data tend to lie on a low-dimensional manifold embedded in the high-dimensional space. An autoencoder, if properly designed can help us to find the proper low-dimensional manifold embedding. In doing so, autoencoders need an additional embedding component in the loss function to ensure that data points maintain the structural similarity in the low dimensional space as they are in the original, high-dimensional space.

The first question to address is how to measure the similarity between data points given the possibility of an embedded manifold structure. We cannot use traditional distance metrics such as the Euclidean distance [7]. The consensus is that one would need to use some sort of geodesic distance metrics in the presence of nonlinear manifold structure, and for that, there are already a few efforts made to incorporate the nonlinear embedding methods in the autoencoder framework [92, 93, 94, 95, 96, 97]. The current embedding approaches choose to utilize either the Laplacian eigenmap (LE) [89] or the locally linear embedding (LLE) [88] methods and combine both reconstruction and embedding loss functions. But they still have one major limitation in common, which is the use of Euclidean distance as a measure of similarity during the calculation of

the embedding representation.

We propose to use a minimum spanning tree (MST) [10], a graph-based approach, to approximate the geodesic distance among data points in lieu of Euclidean distances. Then, it raises a second problem, which is, how to preserve the MST approximated intrinsic structure in the latent space created by the autoencoder. Here, we see an opportunity to provide an integrated solution to both the intrinsic structure recovery and the preservation problems. We devise a graph regularizer based on MST and plant it inside the autoencoder framework as an extra loss-function component, in addition to the original reconstruction loss. We provide two alternative formulations for the graph regularizer, both of which guide the autoencoder to preserve the original data structure when generating the latent features. These latent features in turn help detect the anomalies or rare events from the data.

We apply the resulting graph regularized autoencoder to 20 benchmark datasets to demonstrate its merit in terms of enhanced capability and robustness in anomaly detection. In order to highlight the efficacy of using MST, we compare our graph regularized autoencoder with the autoencoders using no regularizer at all, as well as with the autoencoders of which the regularizer is built on Euclidean distance. To further demonstrate the superiority of our graph regularizer we incorporate it in two generative adversarial network (GAN)-based anomaly detection methods [98, 99]. We find that adding the MST based graph regularizer significantly improves the detection capability of the existing GAN based methods.

The main contribution of this work is that we advance the understanding and practice of designing an autoencoder mechanism for performing dimensionality reduction as well as enhancing its ability for subsequent anomaly detection. The resulting new design is a graph regularized autoencoder using MST to approximate the manifold structure and guide the generation of latent features.

The rest of this chapter unfolds as follows. Section 4.2 discusses the basic concepts and the

working mechanism of autoencoders. Section 4.3 describes the formulation and design of the proposed graph regularized autoencoder. Section 4.4 compares the proposed graph regularized autoencoder with other autoencoder variants on 20 benchmark datasets for the purpose of anomaly detection. The section also demonstrates the performance enhancement when the proposed graph regularizer is added to the GAN-based anomaly detection approaches. Finally, we provide a summary in Section 4.5.

4.2 Autoencoder framework

An autoencoder framework consists of three basic components, as listed below:

- Encoder,
- Decoder, and
- Loss function

We provide a concise summary in this section explaining how autoencoder works. We also explain the idea of embedding loss functions and highlight possible choices of loss function that we could borrow from the literature of nonlinear embedding based methods.

4.2.1 Basic setup

In its simplest form, an encoder is a feed-forward neural network which reduces the dimension of the original data and maps it to a latent space. This funneling process can be done in one step or through multiple steps which depends on the number of layers chosen for the architecture. A simple one layer encoding mechanism (g_ϕ) is summarized in (4.1),

$$\mathbf{Z} = g_\phi(\mathbf{X}) = h(\mathbf{W}_{enc}\mathbf{X} + \mathbf{b}_{enc}), \tag{4.1}$$

where $\phi = (\mathbf{W}_{enc}, \mathbf{b}_{enc})$ contains two sets of parameters: the weight ($\mathbf{W}_{enc}$) and the bias ($\mathbf{b}_{enc}$) parameters of the encoder. They are to be learned during the training process. Now, in the presence of a multi-layer encoder, one needs to repeat the encoding process multiple times until reaching the bottleneck layer. In that case, each layer will have its own weight and bias parameters. In (4.1), $h(\cdot)$ is known as the activation function. The nonlinear mapping of the data is induced through this function. There are many available activation functions to choose from, including but not limited to, the sigmoid function, hyperbolic function, and ReLU function. In (4.2), a sigmoid function is shown, which is differentiable and monotonic.

$$h(\mathbf{y}) = \frac{1}{1 + e^{-\mathbf{y}}}. \tag{4.2}$$

The sigmoid function maps the data onto the real line between 0 and 1 and is the most commonly used when one intends to predict probabilities or the response is scaled between 0 and 1. Instead of choosing one common activation function for all layers, we can also choose different activation functions for different layers.

Decoder is essentially another neural network, trying to reconstruct the original data from the compressed $\mathbf{Z}$ in the latent space. The number of nodes or dimension in the output layer of the decoder must be same as the original input data. Like encoding, decoding can also be done in a single step or through multiple steps. If one chooses to encode through multiple steps, or in other words, to gradually reduce the dimension of data to reach the bottleneck, the decoder should also follow the same philosophy, but it will be done in a reverse order, i.e., to gradually increase the dimension back to that of the original data space. The decoding mechanism, f_θ, is summarized in (4.3).

$$\mathbf{X}' = f_\theta(\mathbf{Z}) = h(\mathbf{W}_{dec}\mathbf{Z} + \mathbf{b}_{dec}). \tag{4.3}$$

Similar to the encoder, there are two parameters in this decoding process, $\theta = (\mathbf{W}_{dec}, \mathbf{b}_{dec})$, and they are also to be learned during the training process. Concerning the activation function in the decoder, we can opt for the same one we choose for encoder layers or we can try a different one.

Fig. 4.1 presents an example, where layer 1 is the input layer, layer 4 is the bottleneck layer, and layer 7 is the output layer. The weight matrices, $\mathbf{W}_{enc} = (\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3)$, are the corresponding to the connections between layers 1 and 2, layers 2 and 3, and layers 3 and 4, and $\mathbf{W}_{dec} = (\mathbf{W}_4, \mathbf{W}_5, \mathbf{W}_6)$ to the connections between layers 4 and 5, layers 5 and 6; and layers 6 and 7, respectively. Here, $\mathbf{W}_1 \in \mathbb{R}^{5\times 7}$, $\mathbf{b}_1 \in \mathbb{R}^{5\times 1}$, and other weight matrices and bias vectors can be likewise constructed.

To achieve an effective representation in the latent space, the autoencoder design is to minimize a loss function, $L(\cdot)$, that quantifies the reconstruction error between $\mathbf{X}$ and $\mathbf{X}'$. One of the most widely used loss functions is the squared error loss, as in (4.4).

$$\min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') = \|\mathbf{X} - \mathbf{X}'\|_F^2 . \tag{4.4}$$

The autoencoder concept can be materialized using many different types of neural networks such as the feedforward neural networks, convolutional neural networks (CNN), recurrent neural networks (RNN), and most recently, GAN. In this chapter, we limit ourselves mostly in the regime of feedforward neural network.

4.2.2 Embedding loss function

To unearth meaningful, effective latent representations in the presence of nonlinear manifold, autoencoders need to have an additional loss component to take care of the embedding problem. The plain autoencoder and the nonlinear embedding approaches can be combined together, and researchers have used a joint loss framework as following [92]:

$$\min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') + \sum_{1\le i<j\le m} G(\mathbf{z}_i, \mathbf{z}_j, \mathbf{D}_{ij}), \tag{4.5}$$

where the first loss component, $L(\cdot)$, reflects the autoencoder's reconstruction loss, and the second component, $G(\cdot)$, captures the embedding loss. In the embedding loss function, $\mathbf{z}_i, \mathbf{z}_j$ are the hidden representation of any two points and $\mathbf{D}_{ij}$ summarizes the Euclidean distance between the same two points in the original space. The embedding loss function actually tries to minimize the differences between an original pairwise distance and the corresponding pairwise distance in the hidden space. There are two popular nonlinear embedding functions used:

1. **Multidimensional scaling (MDS)**

$$G(\mathbf{Z}) = \sum_{i<j} (\mathbf{D}_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|_2)^2. \tag{4.6}$$

In this approach, the main objective is to preserve the inter-point distances in the hidden space [100]. In other words, MDS tries to minimize the difference between the pairwise Euclidean distances, $\mathbf{D}_{ij}$, in the original space and their counterpart in the hidden space. In practice, the minimum of this $G(\cdot)$ function is given by the eigen-decomposition of the Gram matrix of high dimensional data in $\mathbf{X}$ after double centering it.

2. **Laplacian eigenmap (LE)**

In a Laplacian eigenmap, the local properties are generated based on the pairwise similarities among data points [89]. The low dimensional representation is calculated in such a way so that data points closer in the original space should maintain the relative closeness compared to other pairs of data points in the hidden space. It means that if two points are highly similar, then the reward of minimizing the distance between them will be higher compared to another pair of points whose extent of similarity is comparatively lower. In practice, the Gaussian similarity measure is one of the most widely used form of similarity measure. The LE uses

the following formulation as its embedding function:

$$\min G(\mathbf{Z}) = \frac{1}{2} \sum_{i<j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \, \mathbf{W}_{ij} = \mathbf{Z}^T \mathbf{L} \mathbf{Z}, \tag{4.7}$$

where $\mathbf{W}_{ij}$ is the adjacency matrix of the graph, used as the weight to the distances in the hidden space. The second expression is a result by invoking the spectral graph theory. In above equation, $\mathbf{L}$ is the graph Laplacian matrix, which can be obtained by $\mathbf{L} = \mathbf{S} - \mathbf{W}$ where $\mathbf{S}$ is a diagonal matrix, also known as the degree matrix [43, 70].

4.3 Graph regularized autoencoder

In this section, we propose the graph regularizer and show how it can be incorporated in the autoencoder framework. We also discuss the relevant design issues of the graph regularized autoencoder framework and how it can be used to detect anomalies. As the graph regularizer is based on MST, please refer to 1.3 for a brief discussion of the MST's role in manifold approximation.

4.3.1 Proposed graph regularized autoencoder

To design the graph regularizer, we decide to stick with the two embedding loss functions we introduce in Section 4.2.2 but our proposal is to incorporate the MST distance in place of Euclidean distance. For the MDS framework, we replace the pairwise Euclidean distances, $\mathbf{D}_{ij}$ with the MST-based distances $\mathbf{M}_{ij}$ in (4.6). For the LE framework, unlike the traditional LE embedding, we define our similarity measure, $\mathbf{W}_{ij}$ through the inverse of MST-based distances $\mathbf{M}_{ij}$. Specifically,

$$W_{ij} = \begin{cases} \frac{1}{M_{ij}}, & \text{if } M_{ij} > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{4.8}$$

The proposed graph regularized autoencoder framework is expressed as a minimization of the

joint loss function of the reconstruction error and the MST-based embedding function, as in (4.9) below:

$$\min_{\phi,\theta} L(\mathbf{X}, \mathbf{X}') + G(\mathbf{Z}), \tag{4.9}$$

where

$$G(\mathbf{Z}) = \begin{cases} \sum_{i<j} (\mathbf{M}_{ij} - \|\mathbf{z}_i - \mathbf{z}_j\|_2)^2 \\ or \\ \frac{1}{2} \sum_{i<j} \|\mathbf{z}_i - \mathbf{z}_j\|_2^2 \, \mathbf{W}_{ij} = \mathbf{Z}^T \mathbf{L} \mathbf{Z}, \end{cases}$$

$L(\cdot, \cdot)$ is the same as in (4.4),

$Z = g_\phi(\mathbf{X}) = h(\mathbf{W}_{enc}\mathbf{X} + \mathbf{b}_{enc})$,

$\mathbf{X}' = f_\theta(\mathbf{Z}) = h(\mathbf{W}_{dec}\mathbf{Z} + \mathbf{b}_{dec})$,

$\phi = (\mathbf{W}_{enc}, \mathbf{b}_{enc}), \quad \theta = (\mathbf{W}_{dec}, \mathbf{b}_{dec})$.

Using this joint loss, we guide the autoencoder reconstruction mechanism using a graph regularizer, so that it maintains the manifold structure in the low-dimensional space. We, therefore, refer to this proposed autoencoder as the *graph regularized autoencoder*. We want to note that this is a general framework, as one can involve in this framework different types of autoencoding mechanism and choose various neural network architectures.

4.3.2 Anomaly detection

As mentioned earlier, an autoencoder does not necessarily produce an outcome for anomaly detection right away. To flag data points, one assigns the data points an anomaly score to signify how much it is different from normal observations.

To generate anomaly scores, we can follow two routes. The first option is to use the recon-

struction error associated with a data point, defined below:

$$AS_i = \|\mathbf{x}_i - \mathbf{x}_i'\|_2^2 . \tag{4.10}$$

We repeat this process for all observations and rank the scores, $\{AS_i, i = 1, \ldots, m\}$, in descending order, where a higher value suggests more likely to be anomalous.

The second option is to extract the low dimensional representation and then feed it to some existing anomaly detection approaches to detect the anomalies. Using this option, we in this work make use of two existing anomaly detection approaches. One is our local minimum spanning tree (LoMST) [81] and another is the connectivity outlier factor (COF) [101].

In LoMST, a local MST is formed for each observation and its k-nearest neighbors. A LoMST score is calculated for each node, which is the total edge length of the local MST associated with the node. The anomaly score for a node is then calculated as the difference between this node's LoMST score and the average of its k-nearest neighboring nodes' LoMST scores. For this method, a local neighborhood size, k, needs to be specified *a priori*.

COF introduces a new distance measure known as the *average chaining distance* to reflect the isolation of a data point from other points. Chaining is defined as a way to connect the nearest neighbors of an observation by calculating the shortest path incrementally starting from the observation itself without producing a cycle. The length of this chain is known as the chaining distance. The corresponding anomaly scores are calculated by comparing the individual average chaining distance to its neighbors' chaining distances. For this method, again, the neighborhood size, k, needs to be specified *a priori*.

Regardless of which option we choose to produce the anomaly scores, we have to select a cut-off value to flag a data point as an anomaly. For this purpose, we choose the simplest approach, which is to flag the top N scores as anomalies, with the value of N pre-determined. Unsuper-

vised anomaly detection methods are typically used as a first-step screening tool, flagging potential anomalies to be further analyzed and authenticated by more complex and expensive procedure. The choice of N is usually a trade-off between the goal of covering all possible anomalies and the desire to make the authentication of anomalies manageable, i.e., make the more expensive or time consuming subsequent steps practical and feasible.

4.3.3 Design of graph regularized autoencoder framework

To reach an efficient design for our graph regularized framework, we have to settle on some important parameters and components essential for an autoencoder:

4.3.3.1 Hidden layer dimension

For an autoencoder, the most important parameter is arguably the number of nodes in the hidden or bottleneck layer, $\mathbf{Z}$. The intrinsic dimension of the embedding layer is also a critical parameter from the manifold approximation perspective. If the dimension is chosen too small, useful features might be collapsed onto each other and then become entangled, while if the dimension is too large, the projections might become noisy and unstable [102].

Unfortunately, after many years of research in this area, there is still no consensus on how to choose this hidden layer dimension. After much investigation, we in this work choose to follow the procedure established in [103]. The main steps for the intrinsic dimension estimation method are as follows:

1. For each data point, i, calculate the ratio of the distance of the nearest and second nearest neighbors from this point, $\mu_i = \frac{r_2}{r_1}$.

2. Calculate the empirical distribution of μ_i by sorting them in ascending order, $F_{emp}(\mu_i) = \frac{i}{m}$.

3. Fit a straight line through the origin and the points $(\log \mu_i, -\log(1 - F_{emp}(\mu_i)))$.

4. Estimate the slope of this line. Round the estimated slope value to the nearest integer and use it as the intrinsic dimension, l

There are a couple of advantages to this method leading to our choice. The method uses minimal neighborhood information to find out the intrinsic dimension, and because of that, it runs rather efficiently as compared to other approaches. Moreover, we find that the method also saves us from adverse impact of dataset inhomogeneity in the estimation process.

4.3.3.2 Additional components and hyperparameters

Apart from the three main components discussed in Section 4.2.1, we use two auxiliary components in our graph regularized autoencoder. The auxiliary components are not specific to an autoencoder but play important roles in a neural network's training process. The first one is ***batch normalization*** [104], which is to normalize the data before passing to the autoencoding process. Once the input features are normalized to be on the same scale, the weights associated with them would also be on the same scale. Doing so helps avoid an uneven distribution of weights during the training process and prevent the learning algorithm from spending too much time oscillating in the plateau looking for a global minimum. Generally speaking, normalization helps train the network faster.

The second component is known as ***dropout*** [105]. Dropout is a regularization method that helps in reducing the chance of overfitting. When applied to a layer, it means some nodes of that layer will be randomly dropped off, along with all of their incoming and outgoing connections. If dropout is applied, then the layers will look like consisting of a different number of nodes and connectivity to the prior layer. Dropout, since what it does is to make the nodes on a layer have a random probability of being ignored, ensures that the resulting neural network does not rely on any particular input node. As a result, the resulting neural network will not give too much attention, which is often unwarranted, to any particular group of features. The dropout probability needs to be assigned during the training process. Generally, it could be set as 0.5 [105], which is proved to

be optimal for varieties of network architectures and tasks.

As autoencoders follow neural network architectures, one also needs to choose the values of a few standard hyperparameters. They include the number of layers, the choice of activation function, how to initialize the weight and bias values, optimization strategy and the number of epochs during the optimization etc. We summarize in Table 4.1 our choices regarding parameters and components of the graph regularized autoencoder.

Table 4.1: Strategy adopted regarding parameters and components of the graph regularized framework.

Parameters/Components	Setting adopted
Number of hidden layers	2
Activation function	Sigmoid
Dropout probability	0.5
Initialization strategy	Xavier
Optimization strategy	Gradient Descent
Number of epochs	500

4.3.3.3 Denoising graph regularized autoencoder

Basic autoencoder technology is sometimes criticized by arguing that the reconstructed output can be just a copy of the input provided. To prevent such risk, a variant of autoencoder is introduced, known as ***Denoising Autoencoder*** [106]. It takes partially corrupted input and reconstruct the original input with the autoencoder starting from this corrupted input. In this way, the autoencoder cannot simply memorize the training data and copy the input to its output.

There are two underlying assumptions present in this approach. First, the higher level representations are relatively stable and robust to the corruption of the input. Second, to perform denoising well, the model can extract features that capture useful structure in the input data. The steps of the denoising version of autoencoder is outlined below:

1. The initial input $\mathbf{X}$ is corrupted into $\widetilde{\mathbf{X}}$ through stochastic mapping, $\widetilde{\mathbf{X}} = q_p(\widetilde{\mathbf{X}} \mid \mathbf{X})$. Some examples of the corruption process are to add Gaussian noises, salt and pepper noises, or the like.

2. The corrupted input $\widetilde{\mathbf{X}}$ is then mapped to a hidden representation with the same process of the standard autoencoder, $\mathbf{Z} = g_\phi(\widetilde{\mathbf{X}}) = h(\mathbf{W}_{enc}\widetilde{\mathbf{X}} + \mathbf{b}_{enc})$.

3. From the hidden representation the decoder reconstructs $\mathbf{Y} = f_\theta(\mathbf{Z})$

4. The loss function then becomes $L = \|\mathbf{X} - \mathbf{Y}\|_F^2$

During our experimentation, we find that the traditional practice of incorporating noise, as outlined above, is not effective in anomaly detection. So, we devise a new way of incorporating noise in the input data which actually helps us in achieving better latent space representation and consequently improves the detection of anomalies. In this approach, the noisy version of each data point is constructed as the average of its first 5 nearest neighbors as in (4.11). This idea of reconstruction from neighbors is actually on par with the concept of LLE [88], a popular nonlinear embedding approach. The choice of 5 neighbors is arbitrary here, which can be replaced with any meaningful value.

$$\widetilde{\mathbf{x}_\mathbf{i}} = \frac{1}{5} \sum_{k=1, k \neq i}^{k=5} \mathbf{x}_k, \tag{4.11}$$

where $\mathbf{x}_k$ is the k^{th} nearest neighbors of point $\mathbf{x}_i$. If an observation is different from its neighbors, the reconstruction loss would be high; the thought process aligns with the very definition of anomaly.

4.3.4 Conceptual difference with other autoencoder frameworks

To highlight how our graph regularized autoencoder differs as compared to the works discussed in Section 4.1, we summarize them in Table 4.2. All of these autoencoder models are proposed for learning tasks such as clustering, classification, or anomaly detection. Under the heading of

"autoencoder variants," we mainly consider denoising autoencoder and autoencoder with GAN.

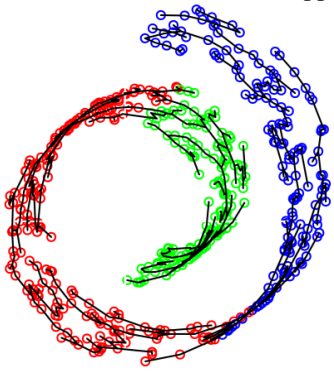
Table 4.2: Summary of autoencoder models. Here (✓) indicates the model includes the criteria.

Papers \ Criteria	Reconstruction loss	Multilayer network	Embedding loss (Euclidean)	Embedding loss (Geodesic)	Autoencoder variants
Hinton et al. [36]	✓	✓			
Yu et al.[92]	✓		✓		
Huang et al.[93]	✓	✓	✓		
Lu et al.[94]	✓	✓	✓		
Jia et al.[95]	✓	✓	✓		
Wei et al.[96]	✓		✓		
Ji et al.[97]	✓	✓	✓		
Schlegl et al.[98]	✓	✓			✓
Zenati et al.[99]	✓	✓			✓
MST regularized autoencoder	✓	✓		✓	✓

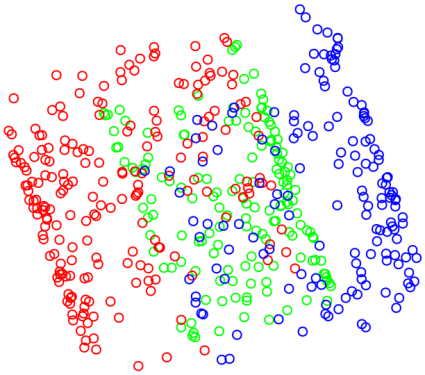
As highlighted in the table, the main difference is the different distance metrics introduced in the embedding function. There are two groups of alternatives that we will compare with in the next section of performance analysis. The first group is the methods listed in rows 1–7, for which either no embedding loss is used (using (4.4)) or the embedding loss is based on Euclidean distance (using (4.5)–(4.7)). The second group is the methods listed in rows 8–9 which are the GAN-based autoencoders. Our purpose is to demonstrate the impact on anomaly detection by using the MST-based metric in the embedding function when the proposed MST regularized autoencoder is compared with the above two groups of methods.

Let us visualize, by using a toy example in Fig. 4.2, the impact of MST and the effectiveness of low-dimensional projection obtained by the MST regularized autoencoder. Fig. 4.2(a) represents the well-known Swiss swirl data structure (a type of nonlinear manifold structure) and the MST approximation of this structure (the black edges). Here, the data points are color coded to help visualize their relative positions. We obtain a 2D representation of the Swiss swirl by three different methods: the principal component analysis (PCA), as in Fig. 4.2(b); an autoencoder using Euclidean distance embedding function, as in Fig. 4.2(c), and our MST-regularized autoencoder,

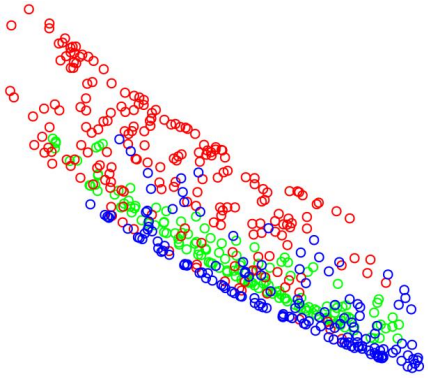
(a) A 3D Swiss roll data and its MST approximation.



(b) Representation in 2D using PCA.



(c) Representation in 2D using Euclidean distance regularized autoencoder.



(d) Representation in 2D using our MST regularized autoencoder under MDS formulation.

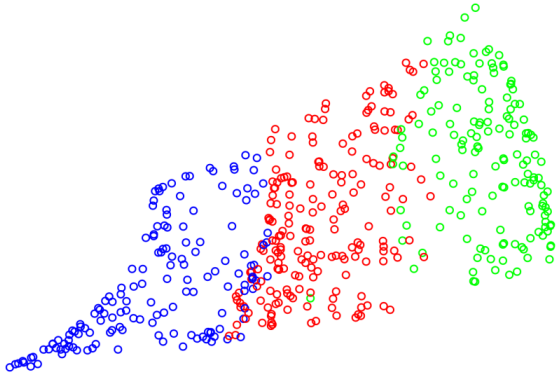


Figure 4.2: MST regularized autoencoder can maintain the structural similarity in low dimensional representation.

as Fig. 4.2(d). The proposed approach is able to maintain the structural similarity presented in the high-dimensional space as expected, when the data is projected to the lower dimension, while the other two approaches did not.

4.4 Performance analysis of graph regularized autoencoder

This section is devoted for evaluating the performance of the proposed graph regularized autoencoder framework for anomaly detection on benchmark datasets. At first, in Section 4.4.1, we compare the detection performance with the plain autoencoder with no regularizer at all and also with the autoencoders with the Euclidean distance-based regularizer. We discuss the effect of using denoising version of the autoencoder and present results using our special denoising option.

Table 4.3: Performance comparison of the autoencoder approaches.

Result (number of datasets) \ Autoencoders	MST regularizer (MDS)	MST regularizer (LE)	Euclidean regularizer (MDS)	Euclidean regularizer (LE)	No regularizer
Better (uniquely best result)	8	8	0	0	0
Equal (equal to the existing best result)	3	4	3	2	0
Close (within 20% of the best result)	3	3	5	6	2
Worse (not within 20% of the best result)	6	5	12	12	18

In Section 4.4.2, we incorporate the MST based graph regularizer into two GAN-based anomaly detection approaches and demonstrate its positive impact.

4.4.1 Performance comparison

We present the detection performance of five autoencoder variants in Table 4.3. The first two are the two formulations of the proposed MST-based regularizer under the MDS formulation and LE formulation, respectively. The third and fourth are the Euclidean distance based regularizer as in (4.6) and (4.7) respectively. The fifth one is the plain autoencoder with no regularizer. To better reflect the methods' comparative edge, we break down the comparison into four major categories in Table 4.3, namely *Better*, *Equal*, *Close* and *Worse*, as explained in the table. In this comparison, we use the reconstruction loss generated from the autoencoder to mark the anomalies (the first option mentioned in Section 4.3.2.

From Table 4.3, we see that MST-regularized methods outperform the other variants of autoencoder. Individually, each method produces 11–12 best detection results and is within the best results for another three cases. Together, the two MST-regularized methods produce the 16 uniquely best detection cases out of the 20 total cases. The two formulations, MDS versus LE, do not seem to make a big difference. The Euclidean distance-based regularizer is shown, on the other hand, to be inferior than the MST regularizer. However, the Euclidean distance-based regularizer performs better than the no regularizer scenario; the latter never scores any best detection but rather often lies far away from the best performer.

Table 4.4 presents the number of true detections by the competing methods. Table 4.3 is summarized based on the information in Table 4.4. In the first column of Table 4.4, the parenthesis besides each dataset's name contains the number of nodes in the input layer followed by the number of the nodes used in the two hidden layers. The last hidden layer node represents the intrinsic dimension of the dataset estimated by using the method in Section 4.3.3.1.

In the performance analysis, we find that the batch normalization and dropout options do not improve the detection results but help achieve faster convergence. We believe having these options could be more helpful, should we use more layers in the encoder/decoder networks. For generating the results in Table 4.4, we in fact did not use these options; but the results would be more or less the same, had we included either or both options.

We keep the number of epochs at 500. When we increase the number of epochs from 500, the performance tends to get better but may happen very marginally. We also test on having different numbers of layers, but this is a trickier problem. It turns out to be that having more layers sometimes helps but in other cases adversely affects the performance.

In Section 4.3.2, we mention the second option of flagging anomalies, which is to take the dimension-reduced output, $\mathbf{Z}$, and feed it to an existing anomaly detection method. We implement the second option, which is using the MST regularizer with the MDS formulation for dimensionality reduction and then use either LoMST or COF to conduct the subsequent anomaly detection. For LoMST, we select the neighborhood size, k following the best k scenario guideline discussed in [81]. We do that for each of the 20 datasets and then use the same k for the COF setting too.

Table 4.5 presents the number of true detections made by the LoMST and COF approaches. We compare them with the detection resulting from using the reconstruction loss to flag the anomalies. We see that the second-option anomaly detection approach in general performs better, especially the one using LoMST. This confirms the merit of autoencoder as a dimension-reducing mechanism enabling or enhancing the subsequent anomaly detection task.

Table 4.4: Number of true positive detections of the competing autoencoder methods. Bold entries represent the best detection performance in a respective dataset.

Dataset	MST regularizer (MDS)	MST regularizer (LE)	Euclidean regularizer (MDS)	Euclidean regularizer (LE)	No regularizer
WBC (9, 8, 6)	7	**8**	6	6	4
Heart(13, 9, 4)	**3**	2	2	2	2
Cardiotocography (21, 12, 3)	30	**32**	29	30	26
SPAMBASE (52, 25, 3)	**38**	27	31	29	18
PIMA (8, 7, 6)	1	**4**	**4**	**4**	2
WDBC (30, 16, 9)	**4**	**4**	**4**	**4**	2
Glass (7, 6, 4)	**1**	**1**	0	0	0
Shuttle (9, 5, 1)	0	**1**	0	0	0
Stamps (9, 7, 4)	3	**9**	3	5	0
Ionosphere (32, 16, 8)	**74**	67	63	59	58
WPBC (33, 16, 9)	10	**14**	10	11	7
KDDCup99 (41, 20, 2)	89	**93**	83	85	39
Lymphography (19, 12, 4)	3	**5**	3	4	1
Arrhythmia (259, 100, 17)	**4**	3	2	2	2
Pendigits (16, 10, 6)	**1**	0	0	0	0
Parkinsons (22, 15, 4)	0	**2**	0	0	0
ALOI (27, 12, 4)	**397**	371	371	337	152
Annthyroid (21, 12, 4)	**13**	11	10	8	4
Waveform (21, 19, 17)	**13**	10	9	6	2
PBLOCK (10, 6, 3)	**19**	**19**	**19**	17	12

Lastly, we explore the benefit of the denoising action. Table 4.6 presents the comparison of our MST regularized autoencoder under MDS formulation with regular and proposed new denoising option. We can see that the regular denoising option does not help much compared to the detection obtained without applying any sort of denoising in column 2 of Table 4.4. However, our proposed denoising option can improve the detection performance compared to the regular denoising process in the majority of the cases.

Table 4.5: Comparison between reconstruction loss based detection and those using an existing anomaly detection methods after the MST-regularizer.

Dataset	LoMST	COF	Reconstr-uction loss
WBC (9,8,6)	7	5	7
Heart(13,9,4)	4	1	3
Cardiotocography (21,12, 3)	23	12	30
SPAMBASE(52,25,3)	51	34	38
PIMA (8,7,6)	6	1	1
WDBC (30,16,9)	7	7	4
Glass (7, 6, 4)	3	1	1
Shuttle (9,5,1)	3	2	0
Stamps (9, 7, 4)	9	6	3
Ionosphere (32,16,8)	101	86	74
WPBC (33,16,9)	14	13	10
KDDCup99 (41,20,2)	91	77	89
Lymphography(19,12,4)	4	3	3
Arrhythmia (259, 100, 17)	5	4	4
Pendigits (16, 10, 6)	0	1	1
Parkinsons (22,15,4)	4	2	0
ALOI (27, 12, 4)	375	322	397
Anthyroid (21, 12, 4)	23	22	13
Waveform (21, 19, 17)	24	20	13
PBLOCK (10,6,3)	38	22	19

4.4.2 Influence of the MST based graph regularizer

GAN has been called one of the most interesting ideas proposed in the last 10 years [107]. We would like to see how our graph regularizer impacts GAN-based anomaly detection once it is incorporated into its loss function.

We consider two GAN-based anomaly detection approaches. The first one is known as the AnoGAN [98]. AnoGAN involves training a deep convolutional GAN, and, at inference, using the trained GAN to recover a latent representation for each test data point. The anomaly score is measured by taking the sum of reconstruction loss and discrimination loss. The reconstruction loss (also called the residual loss) measures how well the resulting GAN is able to reconstruct the

Table 4.6: Change in detection outcomes using the new denoising approach under the MDS formulation.

Dataset	With Regular denoising	With new denoising option
WBC (9,8,6)	7	7
Heart(13,9,4)	3	4
Cardiotocography (21,12, 3)	30	30
SPAMBASE(52,25,3)	38	40
PIMA (8,7,6)	1	1
WDBC (30,16,9)	4	6
Glass (7, 6, 4)	1	1
Shuttle (9,5,1)	0	1
Stamps (9, 7, 4)	3	3
Ionosphere (32,16,8)	74	76
WPBC (33,16,9)	10	11
KDDCup99 (41,20,2)	91	89
Lymphography(19,12,4)	3	4
Arrhythmia (259, 100, 17)	4	5
Pendigits (16, 10, 6)	1	1
Parkinsons (22,15,4)	1	2
ALOI (27, 12, 4)	394	387
Anthyroid (21, 12, 4)	13	15
Waveform (21, 19, 17)	6	19
PBLOCK (10,6,3)	19	25

data from the representative latent points using the trained generator, whereas the discrimination loss measures the performance of the discriminator of the GAN, which is to separate the real data from the fake sample generated by the generator of the GAN. The discrimination loss ensures that the generated data point from the latent space lies on the data manifold. To test the impact of our MST-based graph regularizer, we incorporate the MST-based regularizer as an additional loss component in the discriminator. If a test point is an anomaly, both of reconstruction loss and discrimination loss would be high.

The second approach that we consider, known as the adversarially learned anomaly detection [99, ALAD]. It is claimed to be an improvement over AnoGAN. In contrast to AnoGAN, ALAD uses bi-directional GANs, where an encoder network is used to map data samples to latent

Table 4.7: Performance of GAN-based anomaly detection with and without the MST regularizer.

Method	Precision	Recall	F1 score
AnoGAN (1000 iteration, with regularizer)	0.7461	0.758	0.752
AnoGAN (1000 iteration, without regularizer)	0.4399	0.4469	0.4434
AnoGAN (100 iteration, with regularizer)	0.3514	0.357	0.3541
AnoGAN (100 iteration, without regularizer)	0.1378	0.14	0.1389
ALAD (100 iteration, with regularizer)	0.1901	0.473	0.2712
ALAD (100 iteration, without regularizer)	0.0304	0.0585	0.04
ALAD (500 iteration, with regularizer)	0.3584	0.3641	0.3612
ALAD (500 iteration, without regularizer)	0.1877	0.4476	0.2645

variables. This design enables ALAD to avoid the computationally expensive inference procedure required by AnoGAN as the latent space coordinates can now be generated by using a single feed-forward pass through the encoder network. ALAD also incorporates other ideas to stabilize the GAN training procedure. We add our MST regularizer as an additional loss measurement during the encoder training process to test the impact of such modification.

Table 4.7 presents the performance of these two GAN-based anomaly detection approaches with and without the MST regularizer. The dataset used here is KDDCup99. We use the parameters and other choices as suggested by ALAD's authors [99]. We use their code shared at `GitHub`, modify it to incorporate the MST regularizer, and generate the evaluation scores, which include *precision*, *recall*, and *F1 score*. It is evident that the MST regularizer improves the detection performance for both approaches.

4.5 Summary

To obtain a useful representation of high dimensional data, we need to preserve the intrinsic structure of the data during the process of dimensionality reduction. In this chapter, we propose a new graph-based approach to approximate the manifold structure embedded in the data. We design two separate frameworks for incorporating this graph-based mechanism as an additional regularizer to an autoencoder. The proposed graph regularized autoencoder framework helps process complex data in high dimensions and obtain an effective low-dimensional latent space representation. We argue that adding this graph regularizer enhances an autoencoder's performance in the application of anomaly detection. To support our claim, we present a detailed performance comparison study using 20 benchmark anomaly detection datasets, where the graph regularized autoencoder clearly outperforms two competing autoencoder variants: the autoencoder with Euclidean regularizer and the autoencoder with no regularizer at all. To further demonstrate the positive impact that can be made by this graph regularizer, we incorporate it into two GAN-based anomaly detection approaches and compare the detection results before and after adding the regularizer. As we anticipated, GAN with the graph regularizer performs substantially better than their no-regularizer counterparts. In summary, we believe that autoencoders work more efficiently when a graph regularizer is added.

We investigate a number of issues related to the design of an autoencoder with the graph regularizer. One issue that still eludes us is how to choose the optimal number of layers in the encoder/decoder network design. This issue may be resolved by adopting some of the most recent methods in the AutoML research [108]. Another issue is that our proposed method is limited to numeric data. We believe that the proposed framework can be extended to tackle big data scenarios considering categorical, image and even sequential data but doing so requires changing the depth and type of autoencoder networks, most likely, making full use of modern deep learning techniques.

5. SUMMARY AND CONCLUSIONS

In this section, the overall contributions of this dissertation are summarized. Potential future extensions beyond the scope of this dissertation are also outlined.

5.1 Summary of contributions

The underlying theme of this dissertation is to address the issue of anomaly detection of higher dimensional streaming data with an embedded low dimensional manifold. Towards this theme, a new MST-based similarity measured is proposed and three sets of anomaly detection methods are developed using the newly designed similarity measure. These methods produce promising performances for anomaly detection in the context of high dimensional streaming data. Through this dissertation study, the following insights are gained.

The first and foremost important insight is that in the presence of structured space, Euclidean distance cannot measure the true intrinsic distance between data points and geodesic distance should be capitalized to measure the true structure constrained distance. This phenomenon explains the substandard performance of existing anomaly detection approaches employing the Euclidean distance metric in high dimensional datasets. In this dissertation, the minimum spanning tree (MST), a graph theoretic approach is proposed to approximate the geodesic distance and measure the similarity among observations instead of Euclidean distances.

The second insight is that to detect local anomalies, connectedness needs to be measured on the neighborhood level. A local MST appears to be an effective way of quantifying the connectedness. Using LoMST for anomaly detection, which compares the local connectedness of each observation to that of its neighbors, outperforms nearly all neighborhood-based competitors. In using the LoMST, the same as while using other neighborhood-based methods, the choice of the neighbor size is a crucial decision to make. This dissertation devises a principled approach, which is to max-

imize the standard deviation of the LoMST scores. Empirical analysis supports the effectiveness of this decision process.

The third insight is that ensuring the local invariance property is central to generate a faithful low dimensional embedding. Preserving the local invariance implies that observations maintain the relative closeness in the latent space just like the original space. NMF and autoencoder are among the most popular approaches to create latent space. NMF does so through linear mapping of observations while autoencoder can accomplish nonlinear mapping. However, both NMF and autoencoder do not have any provision for preserving the local neighborhood similarity in the latent space. The newly devised MST based similarity measure provides an effective solution in both circumstances. A revised NMF formulation (NS-NMF) by taking into account the MST approximated similarity outperforms the original NMF model in the task of anomaly detection. A detailed theoretical investigation by comparing NS-NMF with state of art NMF formulations (GNMF and SNMF) confirms that the supremacy in detection is directly attributed to the specific treatment of capturing the neighborhood similarity using the MST. In autoencoder, when an MST-based graph regularizer is added along with the traditional loss function of squared errors, the anomaly detection performance becomes much better compared with the no regularizer or Euclidean distance-based regularizer situation. The ability of the MST based regularizer for a better approximation of the geodesic distances helps produce finer separation of the normal and anomalous observations.

Last but not least, the massive amount of data generated nowadays from the interconnected network of sensors and machines calls for real-time processing due to storage limitations as well as the need to take immediate actions when warranted. However, traditional anomaly detection algorithms are predominantly offline and require an online conversion to meet real-time decision making needs. Keeping these needs in mind, the online versions of the offline methods are developed. The online versions entertain a number of advantages, such as it uses only the current batch of a small number of observations and provision to continually update the variables linked to detection process to keep up with the underlying process, and it updates the decision threshold,

upon receiving the new batch of observations, for deciding in real-time whether an observation is an anomaly or not. The online versions produce comparable outcomes when compared with their offline counterparts. These online approaches can assist the industries to track down any anomaly as soon as it appears and also help prevent any major breakdown or equipment shutdown.

5.2 Further study

Unsupervised learning is undoubtedly one of the least understood territories of machine learning. Yann LeCun, a prominent data scientist and machine learning expert once said: 'If intelligence is a cake, the bulk of the cake is unsupervised learning'. So, it goes without saying that there exist many ways to explore the area of unsupervised anomaly detection and extend this dissertation as well. The manifold hypotheses suggests to consider the possibility of low dimensional embedded manifold, but the truth is the true structure cannot be visualized and consequently it makes the true intrinsic distances among points hard to measure. One can only try to approximate these distances as closely as possible. Still there is no mathematical theorem suggesting an ideal geodesic measure. Empirically MST shows promising performance, but the detailed theoretical study can be carried out in the future.

To implement the online approach, temporal and local neighborhood are needed to be considered. So, there comes always a question of suitable neighborhood size selection. In the dissertation, an adhoc policy is proposed but still there lies a scope to propose a global policy. There are number of issues related to the design parameter selection in autoencoder, waiting to be settled. Most importantly, the number of neurons to be considered in the hidden layer of an autoencoder is still an open problem and needs to be explored. In the dissertation, only the numeric data is considered. So, there remains scope to extend the proposed methods to include the image data. Image data are rich and traditional feed-forward neural network-based autoencoders cannot be utilized. One might adopt convolutional neural network-based autoencoding policy and add graph regularizer on top of that.

In the dissertation, it is proposed to convert the dataset into a graph object and then apply the MST to measure the similarity among the observations. These similarities are used later in the anomaly detection approaches. But instead of using the similarity, one can use the graph itself as an input the neural network model and develop a graph neural network model, which is one of the promising research areas in deep neural network. Sometimes it is also important to understand the anomalies along with the detection. So, a study can be carried out to investigate the root causes behind the detected anomalies and interpret them in terms of the attributes. It can eventually help to pinpoint emerging anomaly patterns.

Finally, it is worth noting that the anomaly detection approaches proposed in the dissertation are generic and can be applied to any domain. Domain specific physical constraints can be generated and added to the model to make a particular approach more suitable for an application area.

REFERENCES

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

[2] Y. Ding, *Data Science for Wind Energy*. CRC Press, 2019.

[3] C. Cortes and V. Vapnik, "Support-vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[4] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[5] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[6] O. Chapelle, B. Schölkopf, and A. Zien, *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006.

[7] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[8] J. Costa and A. O. Hero, "Manifold learning with geodesic minimal spanning trees," *Computing Research Repository*, vol. cs.CV/0307038, 2003.

[9] W.-C. Tu, S. He, Q. Yang, and S.-Y. Chien, "Real-time salient object detection with a minimum spanning tree," in *The IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2334–2342, IEEE, 2016.

[10] R. C. Prim, "Shortest connection networks and some generalizations," *Bell Labs Technical Journal*, vol. 36, no. 6, pp. 1389–1401, 1957.

[11] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48–50, 1956.

[12] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar Boruvka on minimum spanning tree problem translation of both the 1926 papers, comments, history," *Discrete Mathematics*, vol. 233, no. 1-3, pp. 3–36, 2001.

[13] E. M. Knox and R. T. Ng, "Algorithms for mining distance based outliers in large datasets," in *Proceedings of the 24th International Conference on Very Large Data Bases*, pp. 392–403, Citeseer, 1998.

[14] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, vol. 29, pp. 427–438, ACM, 2000.

[15] F. Angiulli and C. Pizzuti, "Outlier mining in large high-dimensional data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 2, pp. 203–215, 2005.

[16] K. Zhang, M. Hutter, and H. Jin, "A new local distance-based outlier detection approach for scattered real-world data," in *Advances in Knowledge Discovery and Data Mining: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 813–822, Springer, 2009.

[17] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, vol. 29, pp. 93–104, ACM, 2000.

[18] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 535–548, Springer, 2002.

[19] W. Jin, A. K. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 577–593, Springer, 2006.

[20] L. J. Latecki, A. Lazarevic, and D. Pokrajac, "Outlier detection with kernel density functions," in *Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pp. 61–75, Springer, 2007.

[21] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "LoOP: Local outlier probabilities," in *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pp. 1649–1652, ACM, 2009.

[22] L. Ertoz, E. Eilertson, A. Lazarevic, P.-N. Tan, V. Kumar, J. Srivastava, and P. Dokas, "Minds – Minnesota intrusion detection system," in *Next Generation Data Mining* (H. Kargupta, J. Han, P. S. Yu, R. Motwani, and V. Kumar, eds.), ch. 3, pp. 199–218, MIT Press Cambridge, MA, 2004.

[23] D. Yu, G. Sheikholeslami, and A. Zhang, "Findout: Finding outliers in very large datasets," *Knowledge and Information Systems*, vol. 4, no. 4, pp. 387–412, 2002.

[24] A. Pires and C. Santos-Pereira, "Using clustering and robust estimators to detect outliers in multivariate data," in *Proceedings of the International Conference on Robust Statistics*, pp. 1–3, CROS, 2005.

[25] M. Amer and M. Goldstein, "Nearest-neighbor and clustering based anomaly detection algorithms for rapidminer," in *Proceedings of the 3rd RapidMiner Community Meeting and Conference*, pp. 1–12, RCOMM, 2012.

[26] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pp. 444–452, ACM, 2008.

[27] K. Pearson, "LIII. On lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[28] J. B. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

[29] C. Aggarwal and P. S. Yu, "An effective and efficient algorithm for high-dimensional outlier detection," *The International Journal on Very Large Data Bases*, vol. 14, no. 2, pp. 211–221, 2005.

[30] J. Zhang, M. Lou, T. W. Ling, and H. Wang, "HOS (high dimensional outlying subspace)-

miner: a system for detecting outlyting subspaces of high-dimensional data," in *Proceedings of the 30th International Conference on Very Large Data Bases*, pp. 1265–1268, VLDB, 2004.

[31] E. Müller, I. Assent, U. Steinhausen, and T. Seidl, "Outrank: Ranking outliers in high dimensional data," in *IEEE 24th International Conference on Data Engineering Workshop*, pp. 600–603, IEEE, 2008.

[32] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek, "Outlier detection in axis-parallel subspaces of high dimensional data," in *Advances in Knowledge Discovery and Data Mining: Proceedings of the 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 831–838, Springer, 2009.

[33] F. Keller, E. Muller, and K. Bohm, "HiCS: High contrast subspaces for density-based outlier ranking," in *IEEE 28th International Conference on Data Engineering (ICDE)*, pp. 1037–1048, IEEE, 2012.

[34] B. Van Stein, M. Van Leeuwen, and T. Bäck, "Local subspace-based outlier detection using global neighbourhoods," in *IEEE International Conference on Big Data*, pp. 1136–1142, IEEE, 2016.

[35] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233–243, 1991.

[36] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[37] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[38] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.

[39] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 267–273, ACM, 2003.

[40] H. Tong and C.-Y. Lin, "Non-negative residual matrix factorization with application to graph

anomaly detection," in *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 143–153, SIAM, 2011.

[41] N. Liu, X. Huang, and X. Hu, "Accelerated local anomaly detection via resolving attributed networks," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pp. 2337–2343, AAAI Press, 2017.

[42] D. Kuang, C. Ding, and H. Park, "Symmetric nonnegative matrix factorization for graph clustering," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, pp. 106–117, SIAM, 2012.

[43] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.

[44] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016.

[45] I. Ahmed, A. Dagnino, A. Bongiovi, and Y. Ding, "Outlier detection for hydropower generation plant," in *Proceedings of the 14th IEEE International Conference on Automation Science and Engineering*, IEEE, 2018.

[46] A. Zimek, E. Schubert, and H.-P. Kriegel, "A survey on unsupervised outlier detection in high-dimensional numerical data," *Statistical Analysis and Data Mining*, vol. 5, no. 5, pp. 363–387, 2012.

[47] M. Yu, A. Hillebrand, P. Tewarie, J. Meier, B. V. Dijk, P. V. Mieghem, and C. J. Stam, "Hierarchical clustering in minimum spanning trees," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 25, no. 2, pp. 1–10, 2015.

[48] X. Tang, W. Huang, X. Li, S. Li, and Y. Liu, "A new local distance-based outlier detection approach for scattered real-world data," in *Proceedings of the 13th Pacific-Asia Conference on Information Systems*, 2016.

[49] S. P. Kola, R. Arun, P. Shanthoosh, P. J. I. Ezra, and A. Kannan, "Network level anomaly detection system using MST based genetic clustering," *Advances in Network Security and Applications. CNSA 2011. Communications in Computer and Information Science*, vol. 196, pp. 113–122, 2011.

[50] X. Wang, X. L. Wang, and D. M. Wilkes, "A minimum spanning tree-inspired clustering-based outlier detection technique," *Advances in Data Mining. Applications and Theoretical Aspects. ICDM 2012. Lecture Notes in Computer Science*, vol. 7377, pp. 209–223, 2012.

[51] A. O. Hero, "Geometric entropy minimization (GEM) for anomaly detection and localization," in *Advances in Neural Information Processing Systems*, pp. 585–592, 2007.

[52] M.-F. Jiang, S.-S. Tseng, and C.-M. Su, "Two-phase clustering process for outliers detection," *Pattern Recognition Letters*, vol. 22, no. 6-7, pp. 691–700, 2001.

[53] D. Montgomery, *Introduction to Statistical Quality Control*. Wiley, 2009.

[54] H. Hotelling, "The generalization of student's ratio," *The Annals of Mathematical Statistics*, vol. 2, no. 3, pp. 360–378, 1931.

[55] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *The Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[56] W. Conover, *Practical Nonparametric Statistics*. Wiley, 1999.

[57] Y. Benjamini and Y. Hochberg, "Controlling the false discovery rate: a practical and powerful approach to multiple testing," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol. 57, no. 1, pp. 289–300, 1995.

[58] I. Ahmed, A. Dagnino, and Y. Ding, "Unsupervised anomaly detection based on minimum spanning tree approximated distance measures and its application to hydropower turbines," *IEEE Transactions on Automation Science and Engineering*, 2019.

[59] W. Wang, T. Guyet, R. Quiniou, M.-O. Cordier, F. Masseglia, and X. Zhang, "Autonomic intrusion detection: Adaptively detecting anomalies over unlabeled audit data streams in computer networks," *Knowledge-Based Systems*, vol. 70, pp. 103–117, 2014.

[60] M. Chenaghlou, M. Moshtaghi, C. Leckie, and M. Salehi, "Online clustering for evolv-

ing data streams with online anomaly detection," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 508–521, Springer, 2018.

[61] Y.-J. Lee, Y.-R. Yeh, and Y.-C. F. Wang, "Anomaly detection via online oversampling principal component analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 7, pp. 1460–1470, 2013.

[62] L. Akoglu and C. Faloutsos, "Event detection in time series of mobile communication graphs," in *Army Science Conference*, pp. 77–79, 2010.

[63] M. Solaimani, M. Iftekhar, L. Khan, and B. Thuraisingham, "Statistical technique for online anomaly detection using spark over heterogeneous data from multi-source vmware performance data," in *IEEE International Conference on Big Data*, pp. 1086–1094, IEEE, 2014.

[64] A. Kejariwal, "Introducing practical and robust anomaly detection in a time series," *Twitter Engineering Blog*, 2015.

[65] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. CRC press, 1984.

[66] E. G. Allan, M. R. Horvath, C. V. Kopek, B. T. Lamb, T. S. Whaples, and M. W. Berry, "Anomaly detection using nonnegative matrix factorization," in *Survey of Text Mining II*, pp. 203–217, Springer, 2008.

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

[68] M. Otey, S. Parthasarathy, A. Ghoting, G. Li, S. Narravula, and D. Panda, "Towards NIC-based intrusion detection," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 723–728, ACM, 2003.

[69] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognition Letters*, vol. 24, no. 9, pp. 1641–1650, 2003.

[70] C. Ding, X. He, and H. D. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proceedings of the 2005 SIAM International Conference on Data*

Mining, pp. 606–610, SIAM, 2005.

[71] R. Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis, "Large-scale matrix factorization with distributed stochastic gradient descent," in *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, pp. 69–77, ACM, 2011.

[72] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1087–1099, 2012.

[73] R. Zhao and V. Y. Tan, "Online nonnegative matrix factorization with outliers," *IEEE Transactions on Signal Processing*, vol. 65, no. 3, pp. 555–570, 2016.

[74] F. Zhu and P. Honeine, "Online kernel nonnegative matrix factorization," *Signal Processing*, vol. 131, pp. 143–153, 2017.

[75] D. Tu, L. Chen, M. Lv, H. Shi, and G. Chen, "Hierarchical online nmf for detecting and tracking topic hierarchies in a text stream," *Pattern Recognition*, vol. 76, pp. 203–214, 2018.

[76] Z. Guo and Y. Zhang, "A label-embedding online nonnegative matrix factorization algorithm," *IEEE Access*, vol. 7, pp. 105882–105891, 2019.

[77] F. Liu, X. Yang, N. Guan, and X. Yi, "Online graph regularized non-negative matrix factorization for large-scale datasets," *Neurocomputing*, vol. 204, pp. 162–171, 2016.

[78] J. Sun, Z. Wang, H. Li, and F. Sun, "Incremental nonnegative matrix factorization with sparseness constraint for image representation," in *Pacific Rim Conference on Multimedia*, pp. 351–360, Springer, 2018.

[79] Z. Chen, L. Li, H. Peng, Y. Liu, and Y. Yang, "Incremental general non-negative matrix factorization without dimension matching constraints," *Neurocomputing*, vol. 311, pp. 344–352, 2018.

[80] X. Zhang, D. Chen, and K. Wu, "Incremental nonnegative matrix factorization based on correlation and graph regularization for matrix completion," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 6, pp. 1259–1268, 2019.

[81] I. Ahmed, A. Dagnino, and Y. Ding, "Unsupervised anomaly detection based on minimum

spanning tree approximated distance measures and its application to hydropower turbines," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 654–667, 2019.

[82] A. B. Goldberg, M. Li, and X. Zhu, "Online manifold regularization: A new learning setting and empirical study," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 393–407, Springer, 2008.

[83] J. M. Bland and D. G. Altman, "Multiple significance tests: the Bonferroni method," *BMJ*, vol. 310, p. 170, 1995.

[84] I. Ahmed, T. Galoppo, and Y. Ding, "O-LoMST: An online anomaly detection approach and its application in a hydropower generation plant," in *Proceedings of the 15th IEEE International Conference on Automation Science and Engineering*, IEEE, 2019.

[85] C. Zhou and R. C. Paffenroth, "Anomaly detection with robust deep autoencoders," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 665–674, ACM, 2017.

[86] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, "Detection of anomalies in large scale accounting data using deep autoencoder networks," *arXiv preprint arXiv:1709.05254*, 2017.

[87] B. Schölkopf, A. Smola, and K.-R. Müller, "Kernel principal component analysis," in *International Conference on Artificial Neural Networks*, pp. 583–588, Springer, 1997.

[88] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[89] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in Neural Information Processing Systems*, pp. 585–591, NIPS, 2002.

[90] B. Shaw and T. Jebara, "Minimum volume embedding," in *Artificial Intelligence and Statistics*, pp. 460–467, JMLR, 2007.

[91] C. Fefferman, S. Mitter, and H. Narayanan, "Testing the manifold hypothesis," *Journal of*

the American Mathematical Society, vol. 29, no. 4, pp. 983–1049, 2016.

[92] W. Yu, G. Zeng, P. Luo, F. Zhuang, Q. He, and Z. Shi, "Embedding with autoencoder regularization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 208–223, Springer, 2013.

[93] P. Huang, Y. Huang, W. Wang, and L. Wang, "Deep embedding network for clustering," in *2014 22nd International Conference on Pattern Recognition*, pp. 1532–1537, IEEE, 2014.

[94] S. Lu, H. Liu, and C. Li, "Manifold regularized stacked autoencoder for feature learning," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2950–2955, IEEE, 2015.

[95] K. Jia, L. Sun, S. Gao, Z. Song, and B. E. Shi, "Laplacian auto-encoders: An explicit learning of nonlinear data manifold," *Neurocomputing*, vol. 160, pp. 250–260, 2015.

[96] C. Wei, S. Luo, X. Ma, H. Ren, J. Zhang, and L. Pan, "Locally embedding autoencoders: A semi-supervised manifold learning approach of document representation," *PloS One*, vol. 11, no. 1, p. e0146672, 2016.

[97] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *Advances in Neural Information Processing Systems*, pp. 24–33, NIPS, 2017.

[98] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging*, pp. 146–157, Springer, 2017.

[99] H. Zenati, M. Romain, C.-S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially learned anomaly detection," in *2018 IEEE International Conference on Data Mining (ICDM)*, pp. 727–736, IEEE, 2018.

[100] J. B. Kruskal and M. Wish, *Multidimensional Scaling*. Sage Publications, 1978.

[101] J. Tang, Z. Chen, A. W.-c. Fu, and D. Cheung, "A robust outlier detection scheme for large data sets," in *6th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 6–8, Citeseer, 2001.

[102] E. Levina and P. J. Bickel, "Maximum likelihood estimation of intrinsic dimension," in *Advances in Neural Information Processing Systems*, pp. 777–784, NIPS, 2005.

[103] E. Facco, M. d'Errico, A. Rodriguez, and A. Laio, "Estimating the intrinsic dimension of datasets by a minimal neighborhood information," *Scientific Reports*, vol. 7, no. 1, p. 12140, 2017.

[104] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[105] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[106] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *The Journal of Machine Learning Research*, vol. 11, pp. 3371–3408, 2010.

[107] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, pp. 2672–2680, NIPS, 2014.

[108] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-weka: Combined selection and hyperparameter optimization of classification algorithms," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 847–855, 2013.