Chapter 1

Nikunj C. Oza and Santanu Das, Anomaly Detection in a Fleet of Systems

"*Most users of complex statistical procedures have problems which can and should be handled with simple techniques." said Andrews, 1981 (Chatfield,* 1985*)*

1.1 What is a Fleet?

A fleet is a group of systems (e.g., cars, aircraft) that are designed and manufactured the same way and are intended to be used the same way. For example, a fleet of delivery trucks may consist of one hundred instances of a particular model of truck, each of which is intended for the same type of service—almost the same amount of time and distance driven every day, approximately the same total weight carried, etc. For this reason, one may imagine that data mining for fleet monitoring may merely involve collecting operating data from the multiple systems in the fleet and developing some sort of model, such as a model of normal operation that can be used for anomaly detection. However, one then may realize that each member of the fleet will be unique in some ways—there will be minor variations in manufacturing, quality of parts, and usage. For this reason, the typical machine learning and statistics algorithm's assumption that all the data are independent and identically distributed is not correct. One may realize that data from each system in the fleet must be treated as unique so that one can notice significant changes in the operation of that system.

The reality of data mining for fleet monitoring lies between these two extremes. Even though each system must be monitored on its own so that one can observe significant changes in its unique operations, data from all systems in the fleet are helpful in determining how unique each system really is relative to the remaining systems in the fleet and when fleet personnel should expect to perform various maintenance actions. One can begin with the assumption that each system in the fleet is comparable to a sample drawn from some distribution, so that all the systems in the fleet are independent and identically distributed. However, one can then adjust the methods being used to account for changes related to any information that is known. For example, if one knows which parts were used at different times, one can make adjustments to the model to account for differences in the qualities of those parts. That way, to evaluate subsystems related to that part, we can choose to use only the other systems in the fleet that have the same part to extract information on maintenance needs and remaining useful life.

In the next section, we will describe some past work in fleet monitoring. We will describe the drawbacks of this past work that motivated the work done recently in this area. We will also discuss what we see as the role of data mining in fleet health monitoring. In the following section, we will describe the key issues of heterogeneity and efficiency-stopping issues (large data volume and distributed data) that serve as challenges to effective fleet monitoring. The following two sections will give overviews of methods designed to handle heterogeneity and efficiency issues. In each section, we will give one example algorithm and its use in anomaly detection in more detail to give an idea of how these algorithms work. We will end with a summary of this chapter.

1.2 Background

In this section we describe some past work in the area of fleet health monitoring and then discuss the role of data mining within this area. This will serve as background material for the following sections which will provide more details.

1.2.1 Past work in fleet health monitoring

In the next two subsections, we briefly describe several systems that have been used to monitor fleets by analyzing numeric data and some other attempts at extracting useful information from text data, respectively.

1.2.1.1 Numeric data

In this section, we briefly describe six systems for fleet-wide health management. Two systems monitor each individual unit of the fleet separately and in an online manner. Information about how other units are currently behaving is not utilized in the monitoring of any unit. Three other systems we discuss collect data from all units in one place and analyze that data to monitor the fleet as a whole and units behaving abnormally are identified. Unlike the first two systems, baselines are not developed for each individual unit or for any subpopulations. The last system attempts to create baselines for the global population and for several subpopulations.

MineFleet [22] is a system that monitors individual vehicles in a fleet by collecting and processing data on-board. It collects data that are produced by many sensors present in most modern vehicles. The sensor data streams are continuously monitored by an on-board computing device for any new emerging patterns. If necessary, these patterns are sent over a low-bandwidth wireless network to a central location for possible further analysis. The system provides the ability to calculate numerous statistical aggregates such as correlations and distance matrices efficiently on the vehicle itself on embedded computing platforms such as PDAs, with the idea that these statistical aggregates form the basis of algorithms to detect anomalies in a vehicle's operation. The system also monitors changes in these statistical aggregates in real time and in an efficient manner. This system is intended to monitor changes in the way the vehicle is operating and is being operated in a setting where the available computing power and communication bandwidth are low. Because of this, each vehicle is only monitored relative to its own baseline—data and statistics from the various vehicles are not communicated to one another automatically. Any knowledge gained from analysis of the entire fleet's data would have to be manually added to each vehicle to enable it to assess itself relative to a global baseline during operation, and that global baseline is not updated in real time. The system allows data from vehicles to be uploaded manually

to a central repository for offline analysis and generation of a global baseline and global knowledge to facilitate manual updating of vehicles' baseline definitions.

The Quantitative Condition Alerting and Analysis Support (QCAAS) system [36] is an on-board software system that uses sensors currently available on most commercial aircraft to monitor for several events of interest such as severe load encounters, landing gear speed exceedances, and hard landings. This system additionally maintains cumulative load statistics for the aircraft on which it is installed and indicates when the allowable level of cumulative load between inspections has been exceeded. The system additionally uses both onboard sensors and physics-based models to estimate loads in the tail of the aircraft, where current commercial aircraft are not instrumented to take direct measurements. QCAAS checks each aircraft independently relative to a global baseline; however, this global baseline is fixed. That is, the global baseline is not learned from data but is rather set by regulations. QCAAS does not currently have the capability to detect anomalies relative to an individual vehicle's normal behavior and is, therefore, unable to adapt to typical vehicle-to-vehicle variations.

The Aviation Performance Measuring System (APMS) [15] was a NASA program to move analysis of data collected during commercial flights, known as Flight Operational Quality Assurance (FOQA) data, beyond identification of single-parameter exceedances. The program identified three major goals—analyzing data beyond simply looking for exceedances of typical ranges of single parameters, focused analysis of higher-risk phases of flight, and looking for potential precursors to aviation safety incidents and accidents. As part of APMS, numerous tools were developed such as tools to search for patterns in particular flight parameters within multiple flights' data, document the distribution of key parameters related to standard operating procedures, report key descriptive statistics grouped by phase of flight, link flight data to corresponding weather data or air traffic data, and multivariate clustering to group flights based on flight signatures derived from parameter values to identify atypical flights. APMS was clearly oriented toward analyzing the data from multiple flights together and did not look at individual flights to assess them based on their own unique baseline. Additionally, APMS's tools analyze the flight data as static data, i.e., the data at each point in time is examined independently rather than as a sequence. The sequential nature of the data is only considered indirectly—by adding parameters such as slopes and flight signatures.

Google [31] collects data from all of its systems and saves them in a repository for offline analysis. Such data is clearly too large to store in a single place; therefore, this data is stored on multiple machines using a distributed file system and Google's well-known Mapreduce framework is used to allow large-scale data analysis programs to analyze that data in a distributed manner and collect the results. However, the distributed storage of the data is not necessarily related to the distributed nature of the Google system being

monitored. Therefore, the data analysis on the collected data is necessarily an offline analysis. No attempt is made to analyze individual servers in the system to obtain a server-specific baseline.

The Morning Report (MR) [3] was designed for individual airlines to use to analyze their flight data in a manner much like APMS—that is, collect all the data together in one place and analyze them as if they came from multiple independent flights drawn from the same distribution. The subsequent System Level Morning Report (SLMR) attempted to address the problem we described earlier of balancing between analyzing all the data together the way MR does versus analyzing each flight as a unique entity and setting each flight's normal baseline separately. SLMR allowed users to analyze flight data in the context of individual airlines and the context of all the airlines. In particular, each airline participating in the system would analyze its data and send summary information on the typical patterns observed in their data to a central site—only summary information is sent in order to preserve each airline's privacy. The combined information is analyzed to find global patterns in various flight phases. Information on these global patterns is then sent back to each airline. Each airline then can analyze its data not only within the context of its own flights, but also within the context of the global patterns. This system was only designed for offline analysis; therefore, after each flight, data is uploaded to that airline's system, and the data is filtered, derived parameters are calculated, single-parameter exceedance checks are performed, flights are partitioned into phases, and each flight parameter's signature is calculated (capturing characteristics like rate of change and variability). These flight signatures are then clustered, enabling identification of flights with anomalous signatures. Linear Discriminant Analysis is then used to devise rules for each of these clusters, which are then used to "classify" new flights. The individual airlines' clusters are then sent to a central site and combined to form a global set of clusters that represent global flight patterns. SLMR has the notion of examining flights relative to the entire global set and an individual airline's flights, but does not examine each flight independently. Additionally, just like with APMS, the sequential nature of the data within a flight is only considered indirectly through the flight signatures, which only give a local indication of how parameters are changing—SLMR is not able to examine a full flight as a sequence. Additionally, the metric by which normality is measured is such that a flight can be normal or abnormal with respect to that airline's data and normal or abnormal with respect to the the global pattern—i.e., all four combinations are possible. This means that in order to assess whether a flight is globally normal, assessing its normality within one or even multiple airlines' data is not sufficient—it has to be examined with respect to all the airlines' data at once. For the sake of efficiency, we would like to have that only locally anomalous flights are globally anomalous. This way, local anomaly detection algorithms can identify relatively few candidates that can then be evaluated for anomalousness at the global level. We will later review two algorithms for which this guarantee has been proven.

1.2.1.2 Text

In this section, we describe issues in working with text. A more comprehensive survey of text classification and text mining in general can be found in [1]. In the area of text classification, there are two key decisions that have to be made: representation of the text and the classification method used. The documents in their raw form are not amenable to classification, so another representation is needed. The simplest document representation is the bag of words (BOW). In this representation, each document is represented by a vector consisting of as many elements as there are unique words in the document repository. Each vector element represents the number of times that a given word occurs in that document. The repository can be represented by a BOW matrix in which each document's vector is a row in the matrix. In machine learning terminology, each document is an example or instance and each word frequency is a feature. In principle, one could then apply one of many possible classification algorithms to document repositories after converting them to BOW matrices. One difficulty with this scheme is the weighting of words. Both rare words and very common words should be given low weight because both types of words are not useful for classification, where one generally wants features that come as close as possible to splitting the set of examples into equal-sized bins. However, very common words are given very high weight in BOW matrices due to their high word frequency. For this reason, weighting schemes such as term frequency inverse document frequency (TFIDF) are often used. TFIDF multiplies the term frequency by the inverse document frequency, which is related to the reciprocal of the fraction of documents in which the word appears. This reduces the weights of very common words. Term frequency and even TFIDF give excessive weight to long documents, so these are often normalized so that all documents' vectors have equal length. Stop words, which are very common words such as articles (e.g., 'the,' 'a,' 'an'), are often eliminated from documents using a stopword list.

The bag of words and its weighted variants have a major weakness—they lose all semantic information because the order of words is lost. A given document is treated exactly the same way no matter how its words are reordered. Semantic information is clearly very important in human understanding of text. Natural Language Processing (NLP) methods attempt to maintain semantic information in documents while making the representation more amenable to machine learning. Examples of what NLP methods do include expanding acronyms and collapsing phrases so that different expressions with exactly the same meaning (e.g., FL vs. flight level) are reduced to one word or phrase, and setting nouns to their singular form and verbs to their infinitive conjugation. Others have attempted to use pairs or triples of words as features rather than individual words. Full parse trees can be constructed in which the subject and object for each sentence are identified and sometimes broken down further into nouns, adjectives, prepositional phrases, etc. The chief drawback of such methods is that they are typically computationally

very intensive. Also, the BOW representation does maintain information on the collocation of terms within a document and the frequency with which pairs, triples, and larger groups of words appear. This appears to often be sufficient for classification, because NLP methods have so far yielded too little improvement in classification performance (at least in aviation safety [12])) to justify their computation time.

Many of the same methods used for non-text classification, such as decision trees, random forests, and Support Vector Machines, are used for text [19, 1]. Past experiments have shown that Support Vector Machines seem to be especially suitable because they are naturally suited to dealing with very large numbers of features, which typically exist in text classification problems [1].

Topic extraction is the act of examining the text repository and identifying the topics represented. Topics are typically in the form of a vector of words that tend to co-occur frequently. Examples of methods that do this include Latent Dirichlet Allocation [10] and Non-Negative Matrix Factorization (NMF) [26]. The representation issues for topic extraction algorithms are the same as for text classification algorithms. However, instead of necessarily relying on an existing document classification system, topic extraction algorithms operate in an unsupervised manner and identify topics based just on the documents themselves.

One example of a text repository is the Aviation Safety Reporting System (ASRS), which was started 30 years ago. It is a joint effort by the FAA and NASA [29]. The program was established to collect data and information about aviation events which could lead to unsafe situations, or non-standard procedures, and use the data and information to identify deficiencies in the National Airspace (NAS) so appropriate solutions could be implemented. ASRS reports are publicly available and are written by pilots, flight controllers, technicians, flight attendants, and others including passengers. In general, the reports are filed in response to a specific event, but general concerns and complaints are also filed. ASRS reports include factual information about the aircraft, location, parties involved and a narrative in which the author describes the event(s) and/or situation. Since these narratives are specific to aviation they are filled with acronyms and abbreviations that only those with a working knowledge of the industry are likely to understand. Each report is read by at least two aviation experts who identify hazardous conditions and underlying causes of the reported events and classify the report into the appropriate categories. There are now over 700,000 documents in the ASRS data base with more than 3000 reports added each month. Many air carriers also have their own internal safety reporting system under the Aviation Safety Action Plan (ASAP) which may or may not be linked to the ASRS. Different airlines devised different categorizations for their documents, which necessitated the development of a master version of the categorizations which all organizations could reference. The archive consisting of these documents recategorized using this master version is known as the Distributed National ASAP Archive (DNAA).

The ASRS documents and their classifications can be used by text classification algorithms to learn models that can be used to classify new documents. However, aviation safety experts do not completely trust the classification system because many problems are unique and do not fit well within the classification scheme. Therefore, topic extraction algorithms that either do not use the text classifications at all or only use them as a guide are of interest with the goal of extracting new topics that represent operationally significant problems that have been previously unknown and/or unstudied.

1.2.2 Role of data mining in fleet health monitoring

Gathering data, transforming data into information, and discovering useful knowledge from the information have been identified as the three major steps in extracting valuable resources from the operations of all kinds of processes or systems. Organizations have recognized the importance of historical data that have been collected over decades and seek ways to utilize this knowledge in improving organizational effectiveness. For this, we turn to the field of data mining. By data mining we mean the art and science of analyzing a large collection of observations. More specifically, data mining is the process of sifting through data and extracting relevant knowledge about the system being described by the data. Such knowledge can then be used to make decisions relevant to the system, such as decisions involving appropriate utilization and maintenance of the system. Data mining is highly multidisciplinary in nature and uses sophisticated techniques from various areas like statistics, pattern recognition, information retrieval, machine learning, knowledge organizations, dynamic programming, high-performance computing, data visualization, etc. The whole purpose is to develop an automated process to manage information, discover previously unknown knowledge, predict trends and behaviors, comprehend the underlying relationships in large data and information sets and finally provide a means to aid decision-making activities.

1.2.2.1 Anomaly Detection Problem

Anomaly detection is an example of data mining. Since the theme of this chapter is anomaly detection, we will mostly discuss data mining problems and applications that involve anomaly detection, which is also known as outlier detection or surprise pattern detection. Outlier or anomaly detection refers to the task of identifying new or unknown patterns which, in many cases, are abnormal or inconsistent relative to most of the data. In the last decade or so, researchers have found tremendous potentials in applications of anomaly detection algorithms in various disciplines. Some of the thrust areas include medical applications, airspace safety, fraud and intrusion, and business analytics. The problem of outlier detection has been extensively studied using several approaches. The two broadest categories of anomaly detection problems are supervised and un-supervised. In the supervised approach a model

is built from training data containing inputs for which the class labels are known. The resulting model can classify new data into one or more of the classes, which may correspond to normal or anomalous categories. Supervised anomaly detection is typically implemented using standard machine learning methods for classification, such as Support Vector Machines (SVMs). However class labels are often not easily available and, if they are available, they are typically expensive to obtain—especially for historical data representing systems which are no longer available in the precise configuration used to generate the data. Additionally, it is often impossible to know all possible classes of normal and anomalous operation and corresponding data. Unsupervised techniques do not require knowledge of normal and anomalous operation modes/classes. Unsupervised anomaly detection methods typically assume that all or most of the training data represent normal behavior and attempt to model this data. When new data arrives, the model is used to identify them as normal if they fit the model and anomalous otherwise. We encourage interested readers to go through [27, 28, 33, 13] which have extensively reviewed and studied various methods for anomaly detection.

1.2.2.2 Some Preliminaries

Figure 1.1 gives an overview of various popular anomaly detection techniques and some of these techniques have been extensively discussed in a survey on outlier detection [13]. Typical classification based techniques such as Bayesian inference, decision trees, Support Vector Machines (SVMs) and neural network models are built on previously labeled instances of both normal and abnormal data instances. In classification based outlier detection, the objective is to assign a new test object to one of the existing classes i.e. label previously unseen data points either as a normal class or any of the abnormal classes. On the other hand, there are some kernel-based classification methods like one-class SVMs, one-class kernel Fisher Discriminants, etc., which fall under the anomaly detection category and are unsupervised in nature. In these detection techniques a solution is offered by modeling normal data and deriving a threshold for determining how far data can be from the model and still be considered normal. In the nearest neighbor based approach, the aim is to infer the outliers based on the data itself—for example, by finding those points which are at a greater distance from most of the other data points or by finding those points which are in low density regions. In most cases k-Nearest Neighbor (k-NN) based solutions have quadratic time complexity since a comparison between every two points is needed to find the nearest neighbors. Researchers like Angiulli and Pizzuti [5], Angiulli and Fassetti [4], Ramaswamy et al. [34], and Bay and Schwabacher [7] have addressed this problem by introducing some promising techniques with improved run time complexities. We will elaborate on some of the above methods in the sections to follow.

In some of the popular clustering techniques such as the ones based on

Gaussian Mixture model using EM, C-means fuzzy clustering, k-means (or median), and centroid linkage hierarchical approach, the basic approach is to partition the entire data into a number of clusters, where each data point will be assigned to each cluster with certain degree of association, also called "*degree of membership*". Here the anomaly detection problem is framed as a method of finding the odd member (data point) whose association to any of the existing clusters is very low—i.e. the sample does not belong to any of the existing clusters.

In statistical approaches, the models are based on the statistical properties of the training data themselves. In the testing phase the job boils down to estimating whether the sample under investigation could have been generated by the same distribution that generated the training data. There exist two main approaches, of which one makes some assumptions about the data distribution and the other derives the distribution along with the distribution parameters from the data. The former are known as parametric methods, which assume that the density function belongs to one of the standard and well-known distribution families, such as the normal distribution, binomial distribution, and Poisson distribution. On the other hand, the latter approach, which does not assume a particular form for the underlying distribution of the data in advance, is widely known as the non-parametric approach.

Most of the algorithms described above have been mostly used for data containing only continuous real-valued data attributes. However, many applications, such as aviation safety, have heterogeneous data sources (e.g., discrete, continuous, text, and others) and; therefore, there is a need to develop intelligent knowledge refinement and integration techniques that work with heterogeneous data sources. It is important to note that in order to enable knowledge discovery, algorithms in general require an integrated and merged view of the data available from various sources. Knowledge extraction from multiple heterogeneous data sources remains a challenge. Another challenging problem is to keep the methodologies flexible to the heterogeneous nature of the data sources so that problem reformulation is not required whenever there is a change in information content or data structure. In the rest of the chapter we will describe different frameworks for data integration and knowledge discovery that can utilize heterogeneous data sources. Our notion here is to demonstrate how to extract and merge information contained in heterogeneous data sources and how the combined information can be effectively used to achieve the needs of the analysis. More recently, much research has been conducted to extend the scope of some of the state-of-the art techniques to include information from continuous data, as well as knowledge created from other structured data sources, such as discrete data which may be binary, ordinal, categorical or sequential. Here we will discuss a few of them. Some of these techniques can be even further extended for various unstructured data sources as well.

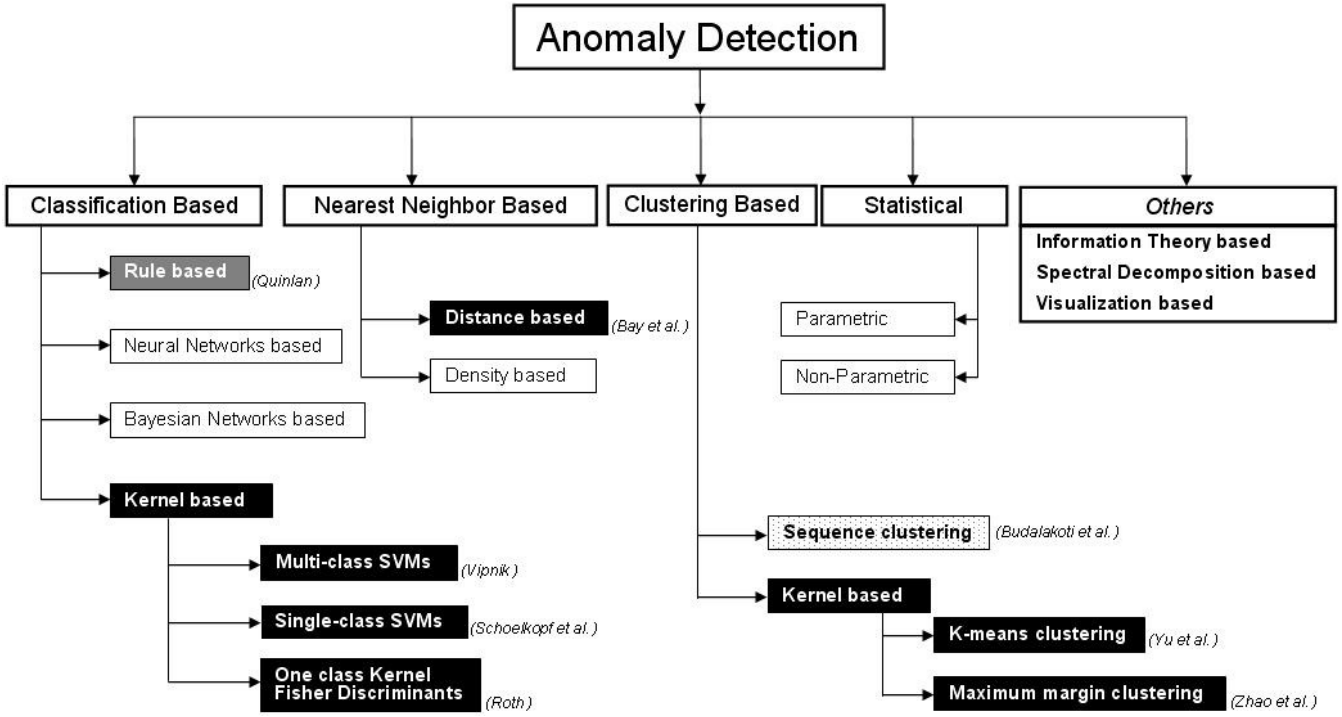

FIGURE 1.1: This figure shows a systematic chart representing some popular approaches to anomaly detection and their sub categories.

1.3 Key issues in fleet health monitoring

In this section, we examine two challenges in many modern fleet data sets: heterogeneity and large data volume. We follow these in the next section with methods designed to address these challenges.

1.3.1 Heterogeneity

"Heterogeneity is to be expected in a meta-analysis: it would be surprising if multiple studies, performed by different teams in different places with different methods, all ended up estimating the same underlying parameter. From the standpoint that heterogeneity is inevitable in a meta-analysis, we are left with the question of whether there is an acceptable degree of heterogeneity. My own view is that any amount of heterogeneity is acceptable, providing both that the predefined eligibility criteria for the meta-analysis are sound and that the data are correct. The challenge is then to decide on the most appropriate way to analyze heterogeneous studies, and this will depend on the aims of the synthesis....." (Julian P T Higgins [20])

In the light of the above statement, it seems clear that we must expect the presence of varying degrees of heterogeneity across different problems. It is fairly well accepted among practioners that mixtures of observations described using multiple attributes (parameters, features etc.) of diverse nature, measured with varying degrees of uncertainty results in heterogeneity in the data. Sometimes these heterogeneous data sets can reveal enriched information that can help in making critical decisions. These decisions can be in the form of detection, classification, prediction, forecasting, etc. The fact that this heterogeneous nature of the data influences the entire analysis and decision-making process is a central point of this chapter. Imagine that we are analyzing a large volume of heterogeneous data with unknown class labels and we are asked to identify those objects exhibiting strange behavior. One of the popular ways of doing this is using a data-driven anomaly detection model which is a process of identifying abnormal or inconsistent patterns in a dataset. Identifying strange events or entities can be extremely important in several applications including fraud and intrusion detection, financial market analysis, medical research, safety-critical vehicle health management etc. These adverse events or entities are also known as anomalies or outliers which can be defined as data points that lie in low density regions or in areas far away from the majority of points. The choice of the appropriateness of the algorithm is dependent on the nature of the input features and the goal that needs to be achieved. In this chapter, we study the requirements of anomaly detection methods that take into account the heterogeneous nature of the data set to accurately find anomalies. This chapter summarizes studies on such anomaly detection techniques that have appeared in the technical literature during the last two decades.

The primary purpose of this study is to bring together and compare a set of methods for anomaly detection on heterogeneous datasets. We start off by discussing various heterogeneous sources within data sets. We examine some of the existing state-of-the-art algorithms for anomaly detection in data containing both discrete and continuous variables. Furthermore we demonstrate that some of the benchmark algorithms may be motivated by the problem of working with more complex types of data where sequential dependencies exist between points in the data set. We focus on techniques that can extract information from multivariate heterogeneous data sources.

Over the last few decades we have seen a tremendous increase in information flow, in terms of the volume and complexity, in several disciplines. Today, we are left with the challenge of dealing with a vast number of heterogeneous information sources in a variety of semantic structures. Knowledge discovery from these heterogeneous resources is still a challenging task. By knowledge discovery we mean the process of extraction of implicit, either known or unknown, potentially useful information that can be used to infer particular events or general characteristics. Knowledge discovery is composed of many computer science subfields like machine learning, pattern recognition, and expert systems, as well as other fields like statistics, information theory, and stochastic machines. With this increased complexities of data sources there is a potential need for building intelligent refinement and integration frameworks, focusing on information content and semantics.

The key aspect of any data analysis method is how the input data were measured and transformed into information. The two major categories of data are structured data and unstructured data. Unstructured data are data with limited restrictions on their format. A typical example of unstructured data is free text data; for example, reports or scripts describing some events, and experts' feedback on a process. Structured data have much more restricted formats. One typical example is numeric data, in which the attributes may be either continuous or discrete. Continuous data can be defined as numerical responses measured on an infinitely divisible scale. Continuous variables can be used to describe the properties of any process or the assets related to that process. For example, the measured variables can be used to express parameters like height, weight, length, temperature, pressure etc., of a wooden box or it can express the time needed to build the box or the cost (in terms of money) of the box. Discrete data are numeric attributes measured on a finitely divisible scale. Discrete attributes can be either binary or categorical, and they can either be independent measurements or part of a sequence. An example of binary data is measuring the switching (on/off, presence/absence etc.) characteristics in a process. An example of categorical data is a satisfaction rating on a scale of 1 to 5. In contrast to the previous two categories, discrete sequences preserves the order (time aspect) of the discrete events. For example, the sequence ABC is different from CBA as the order information has been reversed in the later case, where A, B and C are categorical values or events.

In any application, heterogeneity is the measure of how different the attributes of the data are from one another. There are a wide range of sources from which heterogeneity may be introduced in data attributes. Heterogeneity may occur due to the presence of multiple attributes as compared to a single attribute in each data point. The latter is termed as univariate data while the former is multi-variate data. In multivariate data, the attributes may or may not belong to the same data type. For example each attribute can be either continuous or discrete, independent of every other attribute. The other source of heterogeneity can be the behavioral or functional properties of these attributes. Some typical relationships that are present among data instances are spatial, temporal, spatio-temporal, graphical etc. A detailed description of the nature of input data can be obtained in [13]. Another type of example is the logical ordering in sequential data. Either it defines a relationship among observations or it may define the switching behaviors of the associated features. As a simple illustration we can take the example of the Fibonacci numbers which are an integer sequence where the first two Fibonacci numbers are 0 and 1, and each subsequent number is the sum of the previous two [42]. An interesting way to characterize the switching behavior of a group of variables is to generate a vector where each item in the vector represents a combination of a switch attribute and the direction in which it was flipped. Later in this chapter we will further discuss situations where discrete switching patterns may have some causal relationship with some continuous attributes.

1.3.2 Efficiency: Large data volume, distributed data

Often, data representing the operations of fleets are very large. For example, major commercial airlines have databases containing tens to hundreds of terabytes of flight-recorded data. Simply reading the data from a disk may take months. Therefore, anomaly detection methods need to be judicious in deciding how large a subset of the data they need to read in order to derive useful results. Often, these data are distributed over many storage devices. This happens not only because of the substantial amount of data which is more than typically can be stored at one location, but also because of the geographically distributed nature of the data generation, which makes storing the data at locations close to where they were generated convenient. For example, one can imagine storing the data from a commercial aviation fleet at the destination airports of the corresponding flights. That is, every time an aircraft lands, its data may be uploaded to a storage device located at that airport. Consolidating the data may be impractical given how much data there is. In addition, the fact that the data are distributed should be beneficial—methods should be able to process each storage location in parallel, which would clearly be faster than processing all the data sequentially on a single computer. Therefore, anomaly detection algorithms that can operate on the data even though they are distributed, but give the same results as what a

traditional algorithm would give after centralizing all the data, are clearly useful. We will discuss such methods later in this chapter.

1.4 Dealing with heterogeneity

As discussed earlier, we needs ways to integrate and merge knowledge of the data available across various resources to discover abnormal patterns or outliers in the data. One approach, which we call the *hybrid approach*, involves multiple model development, where we choose, for each modality of data that is present, one or more models appropriate for extracting useful information from that data. The results of these models can then be combined to yield useful information about the entire dataset. This is depicted in figure 1.4. The basic idea is to complement the deficiencies of one algorithm with the strength of the other algorithm or algorithms. A decision is made taking into account the combined output of all the supporting models. For example if the objective is to discover abnormal patterns in a heterogeneous numerical data set with both continuous and discrete values, we can choose method A that works well on the continuous data and another method B that works in the discrete domain. We can then use a suitable way to amalgamate the findings of both the methods to generate a set of anomalies in that data set. Here A and B may be any machine learning techniques (see figure 1.4).

An alternative way to approach this problem is to examine the possibilities of knowledge integration, followed by the development of a single model on the combined knowledge. This is depicted in figure 1.4. In general, authors approach this problem by using a group of complementary metric functions to extract useful information from various data sources and thus, in a way, maximizing the knowledge extraction. Each metric function is chosen based on their performance on some information content or data types. Once the knowledge integration is complete, the remaining task is to build a suitable model to attain the goal of the analysis.

Choosing the right metric function and integrating knowledge from multiple sources must be done judiciously. We will later discuss a method that calculates a weighted combination of such functions to perform anomaly detection on heterogeneous data. Identifying the best combination of weights to integrate knowledge sources is itself an optimization problem and is another interesting research area which is not within the scope of this discussion.

In the following sub-sections we will describe some anomaly detection techniques that can easily integrate information from multiple heterogeneous data sources. Some of these are shown in dark shaded blocks in figure 1.1. The current state-of-the-art algorithms have both strengths and shortcomings in detecting a variety of anomalous conditions. The following subsections will present methods that aim to combine both strengths into a single approach

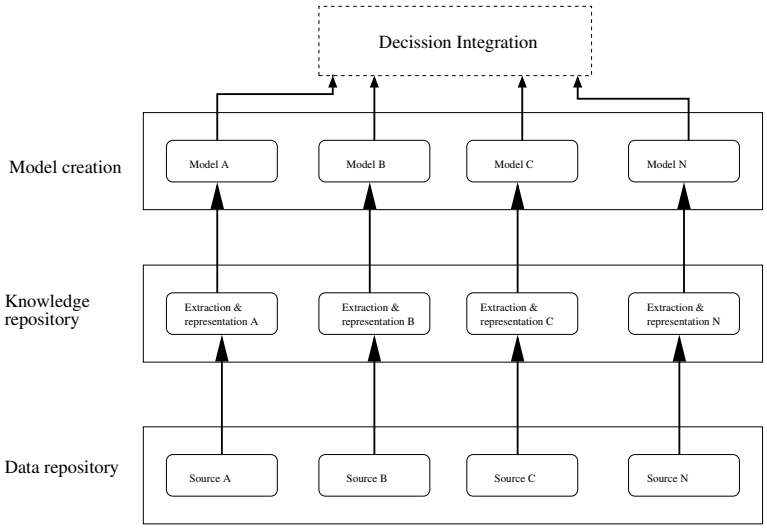

FIGURE 1.2: In this figure we illustrate the two stage conceptual model for knowledge discovery. The first layer represents heterogeneous data sources. The second layer is the knowledge management process layer where the data is appropriately processed (preprocessing, refinement etc.) to knowledge. The third layer represents how the knowledge was communicated, corroborated or shared to build models. In this framework individual models are created on different knowledge bases and a decision is made at the final stage corroborating all the outputs of different models.

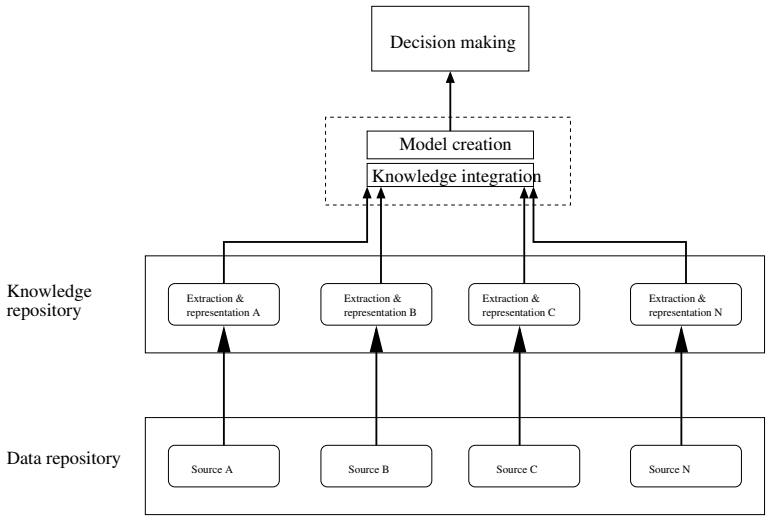

FIGURE 1.3: This is an alternative knowledge discovery scheme compared to the two stage conceptual model shown in figure 1.4. The first two layers represent heterogeneous data sources and knowledge management process. The third layer represents how the knowledge was integrated, communicated, collaborated or shared to a single model. In this framework single model is created on the fused knowledge and a decision is made at the final stage.

to allow for detection of a variety of anomalies. The benchmark anomaly detection algorithms discussed in this chapter are Orca, GritBot and one-class Support Vector Machines.

1.4.1 Orca

Orca [7] is a method used for detecting anomalies in both continuous and discrete (binary format) data in vector space, using a nearest neighbor based approach to detect anomalous points. For continuous data, Orca takes a reference data set and calculates some function over each data point's nearest neighbors (using the Euclidean distance) in the original vector space. For binary attributes the Hamming distance [43] is used. Orca has a nested loop structure to calculate pairwise distances between data points but uses randomization and a simple pruning rule to keep the algorithm's actual time complexity significantly less than the square of the number of data points. In fact, the pruning used in this algorithm helps to achieve near linear time performance with high dimensional data. This makes the algorithm scalable for analyzing large data sets. The advantages of distance-based anomaly detection algorithms are that no explicit distribution needs to be defined to determine a priori whether data points are normal or anomalous, and that they can be applied to any feature space for which we can define a distance measure. In this algorithm, each data point is scored independently and therefore it is unsuitable for sequential data where the order of data points is operationally significant. The pseudo code of Orca is shown in figure 1.4.1.

1.4.2 GritBot

GritBot [35] is a commercially available machine learning tool that performs anomaly detection on discrete and/or continuous attributes. Gritbot uses the C4.5 decision tree algorithm [32] to detect outliers. GritBot was developed by RuleQuest Research. GritBot is more like a classifier which defines a simple boundary that separates the normal patterns from the rest and tests whether the new data falls inside or outside the boundary. The outlierness is based on values of discrete variables or ranges of continuous variables, for which the target variable holds a specific value. The point is considered to be an anomaly when the target variable corresponding to that point is significantly different from the rest of the target values of the other points in that subset. GritBot does not provide an overall score, or even monitor the score for each data point, but instead ranks the top anomalous scores according to their statistical significance. Since GritBot is based on decision trees, it requires no prior knowledge of the data unlike many machine learning techniques that may require parameters or distribution models derived from the data set.

Algorithm 1 Orca Algorithm

Input:
A_{pq} : Matrix with q dimensional dataset having p instances arranged in a random order
k : Number of nearest neighbors (default 5)
n : Number of outliers used (default: $n = p$)
Output:
$O(n)$: Set of outliers
S_{global} : Global scores
Argument:
a : Entries in A
B : Block of examples from A
b : Entries in B
C : Cut-off threshold
w_d : Weight of discrete parameters
Definitions:
$x \in (x_c, x_d)$
$D \in (D_c, D_d)$
$H_d(x_d, D_d) = w_{d_1}(x_{d_1} \neq D_{d_1}) + w_{d_2}(x_{d_2} \neq D_{d_2}) + \ldots + w_{d_n}(x_{d_n} \neq D_{d_n})$
$d(x, D) =$
$\sqrt{(x_{c_1} - D_{c_1})^2 + (x_{c_2} - D_{c_2})^2 + \ldots + (x_{c_n} - D_{c_n})^2} + H_d(x_d, D_d)$
$d(x, D)$: maximum distance between x and an example in D
$M^k_{x,D}$:k closest example in D to x
$S(D, x) = \frac{1}{m} \sum_{i=1}^{m} d(x, D)$ distance based score
Initialize:
Let p instances of A_{pq} be divided in N_B blocks and $K_{nn}(x)$ be the matrix that keeps track of the nearest neighbors/examples of x. And $C = 0$ and $O = \Phi$, where Φ is a null vector
For $block = 1 : N_B$ {
$B = A(:, block); K_{nn}(b) = \Phi$
For each a in A, {
For each b in B and $b \neq a$ {
If $Length(K_{nn}(b)) < k$ or $d(b, a) < dm(b, K_{nn}(b))$ {
$K_{nn}(b) \leftarrow M^k_{b, K_{nn}(b) \cup a}$
If $S(K_{nn}(b), b) < c$ {
Remove example b from set B
}}}}
$O = O \cup B$
$S_{global} = score(O)$
$C \leftarrow min(S_{global}(0))$
}

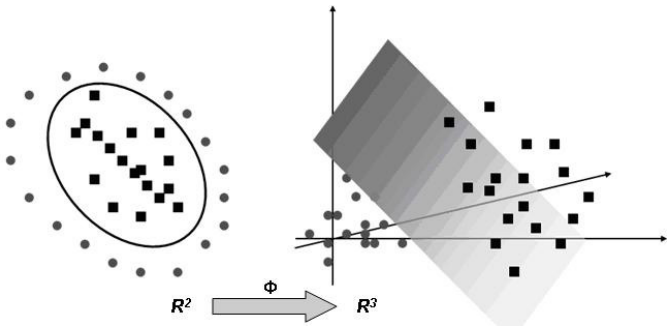

FIGURE 1.4: This figure illustrates the higher dimensional mapping of the data. The data is non-linearly separable in 2−d data space whereas it is linearly separable in the 3−d feature space.

1.4.3 Kernel Based Anomaly Detection

In this section, we describe kernel-based methods for anomaly detection in more detail than we devoted to other algorithms because of recent results demonstrating the strong promise of kernel methods for anomaly detection in heterogeneous datasets. In general, a kernel function, which is the heart of any kernel method, can be informally thought of as a measure to calculate the similarity between two data points. The use of certain kernel functions turns out to be equivalent to mapping $n-$dimensional input data into a high dimensional (possibly infinite dimensional) feature space and then using linear methods within that feature space. This is possible because vectors in the high-dimensional feature space are only present as part of dot products, which are always scalars. Tasks like anomaly detection, classification, clustering, or regression are then performed in this high-dimensional feature space. This method of mapping the data into a high-dimensional space and using linear methods rather than using non-linear methods in the original input space yields the benefits of linear methods (well established theory, guarantee of achieving the optimal solution given a fixed training set) with the representational benefits of a non-linear method. Support Vector based classification, regression, anomaly detection are classical examples of kernel based methods. Some popular kernel based techniques are shown in shaded (solid color) blocks in figure 1.1.

1.4.3.1 Kernel Theory and Operations

Often, we find that linear methods for anomaly detection, classification, and regression are not sufficient. In such cases, we may like to use methods that are nonlinear in the input space. However, such methods' theory are not as well developed as linear methods. Additionally, methods for finding nonlinear models, such as gradient descent methods, are typically not able to guarantee

that they find the optimal solution given the training data supplied. Kernel methods avoid this problem by instead mapping the data to a much higher dimensional space (called the *feature space*) and then using linear methods within the new space. The conceptual diagram shown in figure 1.4.3.1 depicts the data mapping. The left side shows the representation of the data in the space of the original features (depicted here as a two-dimensional space) whereas on the right hand side the same data has been represented in the feature space, which is depicted here as three-dimensional. The feature space is almost always of much higher dimension than the input space. In the input space $\mathcal{R}$, suppose we are given the data $\mathcal{D} = \{(\vec{x}_i)\}_{i=1}^n$, where $\vec{x}_i \in \mathcal{R}^d$ ($d = 2$ in figure 1.4.3.1). We define a feature space $\mathcal{F}$, and assume that there exists a function ϕ that can be used to map any variable x from the input space to the feature space i.e. $\phi : \mathcal{R}^d \rightarrow \mathcal{F}$. The feature space is often of infinite dimension. However, the feature space is never explicitly defined, but rather is induced by a *kernel function*. That is, a kernel function k is defined such that it induces a mapping ϕ. That is, $k(\vec{x}_i, \vec{x}_j) = \phi(\vec{x}_i) \cdot \phi(\vec{x}_j)$. For example, the Gaussian kernel $k(\vec{x}_i, \vec{x}_j) = exp(-\|\vec{x}_i, \vec{x}_j\|)/2\sigma^2$ induces a function ϕ that maps the input space into an infinite-dimensional feature space, but one need not know what that feature space is because the feature-space vectors are only represented as dot products which can be replaced by $k(\vec{x}_i, \vec{x}_j)$. This implicit use of very high or infinite-dimensional vectors is commonly referred as the *kernel trick* in the machine learning literature.

1.4.3.2 Is It a Mercer's Kernel

Now the question arises how to chose a kernel function which is "appropriate?" The answer to this question is illustrated using a combination of two very different views. The first and definitely the most immediate requirement to be an "appropriate" kernel function $k(x_i, x_j)$ is to prove that the function satisfies Mercer's conditions. This means that the function must be continuous, symmetric, and positive definite. Any function that satisfies these three conditions is known as a Mercer kernel. In general, a Mercer kernel can be used because the kernel trick can be applied to it.

$$\textit{Continuous} \quad \forall \epsilon > 0 \; \exists \delta > 0 \; s.t. |k(x_i, x_j) - k(x_i \pm \delta, x_j)| < \epsilon \tag{1.1}$$

$$\textit{Symmetric} \quad k(x_i, x_j) = k(x_j, x_i) \tag{1.2}$$

$$\textit{Positive definite} \quad \sum_{i,j} a_i a_j (k(x_i, x_j)) \geq 0 \tag{1.3}$$

The continuous (Eqn. (1.1)) and symmetric (Eqn. (1.2)) properties are fairly easy to understand. The positive definiteness property of function k means that it must satisfy the following mathematical condition. For all n, $i \in \{1, 2, \ldots, n\}$ and $\vec{x}_i \in \mathcal{R}^d$, the function k results in a kernel matrix (Gram matrix) K such that $K_{i,j} := k(x_i, x_j)$ which is positive definite as shown in Eqn. (1.3).

Apart from mathematical convenience, another criterion for selecting a kernel function typically depends on the purpose of the study. Using an appropriate function will help in extracting useful knowledge from the underlying data. The kernel function will be more effective if it is sensitive to the knowledge the user is looking for. This implies that the choice of the kernel is based on how the kernel function matches the application. The kernel function should be a measure of similarity appropriate to the application.

1.4.3.3 Problem Specific Kernel Functions

There are several class of kernels that satisfy Mercer's conditions. In this section we will elaborate on a few interesting candidates out of many popular kernel functions which are greatly used in kernel based learning algorithms. A very popular example of such kernel is the Gaussian kernel, also known as radial basis function (RBF), which takes the form of,

$$k(\vec{x}_i, \vec{x}_j) = e^{-||\vec{x}_i - \vec{x}_j||^2/2\sigma^2} \tag{1.4}$$

where $||.||$ denotes the Euclidean norm and σ defines the width of the Gaussian distribution, also known as kernel width. RBF is one of the most widely used kernel functions due to its translation invariance property—the value of the kernel at any two points does not depend on their absolute positions but only on the distance between them in the input space. The RBF kernel width (σ) must be greater than zero. The choice of the kernel width plays an important role in effectively capturing the spread among the data and thus affects the performance of algorithms. There is active research on techniques to automatically choose the optimal value of the kernel width (σ) from the data.

Another commonly used kernel is the polynomial kernel. There are two distinct features of the polynomial function, namely the degree or power (d) of the variables and the offset (c). The two different types of polynomial functions are shown in Eqn. (1.5) and Eqn. (1.6).

$$k(\vec{x}_p, \vec{x}_q) = \langle \vec{x}_p, \vec{x}_q \rangle^d \quad Homogeneous \tag{1.5}$$

$$or = (\langle \vec{x}_p, \vec{x}_q \rangle + c)^d \quad Inhomogeneous \tag{1.6}$$

Under specific scenarios when the user wants to model switching sequences for a given process and where the order of the switching is important, normalized Longest Common Subsequence (nLCS) based kernel is a good candidate. To define this, we first give some preliminary definitions. z is a subsequence of x if there are symbols that can be added before and/or after z to obtain x. z is a common subsequence of $\vec{x}_i$ and $\vec{x}_j$ if z is a subsequence of both $\vec{x}_i$ and $\vec{x}_j$. The longest such subsequence of is called the longest common subsequence (LCS) and is denoted by $LCS(\vec{x}_i, \vec{x}_j)$ and $|LCS(\vec{x}_i, \vec{x}_j)|$ is its length. Such a kernel over discrete sequences, when normalized, takes the form of,

$$k(\vec{x}_i, \vec{x}_j) = nLCS(\vec{x}_i, \vec{x}_j) = \frac{|LCS(\vec{x}_i, \vec{x}_j)|}{\sqrt{l_{\vec{x}_i} l_{\vec{x}_j}}}, \tag{1.7}$$

where $l_{\vec{x}}$ is the number of symbols in sequence $\vec{x}$. Each sequence of switches is compared against other sequences by using the longest common subsequence (LCS) as the metric for comparison. Sequences that are similar are bound to hold high nLCS values, while dissimilar sequences will hold very low nLCS values.

If the data being used is text, then one may use the bag-of-words representation, which is a very simple representation of text. A bag of words is a vector whose length is the number of words M used in all the documents in a repository (minus "stop words"—very commonly used words such as "the," "a," "an," and others that are found in nearly all documents and therefore have little usefulness is distinguishing unique characteristics of documents). Each entry in this vector is the number of times that this word appears in the corresponding document. Clearly, this representation does not retain information on the order of words in the document.

Given a corpus of N documents and a dictionary of M words, one possible goal is to establish "relationships" between documents. Latent semantic analysis aims to do this. A co-occurrence matrix (C) of size $M \times N$ can easily be formed. Each column in the co-occurrence matrix represents a document ($d_\diamond$) with the number of occurrences of every term while each row ($t_\diamond$) represents the number of times a term from the dictionary is contained in all the documents. Entry C_{ij} represents the frequency of the i^{th} term in the j^{th} document. A kernel based on the co-occurrence matrix can be defined as,

$$k(\vec{x}_p, \vec{x}_q) = d_p^T d_q \quad \forall p, q = 1, ..N \tag{1.8}$$

$$or = t_p^T t_q \quad \forall p, q = 1, ..M \tag{1.9}$$

Here Eqn. (1.8) gives the similarity between two documents over all the terms while Eqn. (1.9) gives the similarity between two terms over all documents.

There are many papers describing the use of kernel methods for many different types of data simultaneously with various types of features such as graphs and multiple feature types in computer vision such as color, shape, texture, and graphs based on image segmentations. Interested readers can explore literature [40, 23, 14, 44, 18] that looks into various other classes of kernels like sigmoid, spline, graph based, tree based, mismatch-based functions, etc.

1.4.3.4 Information Fusion

In previous sections we have demonstrated that evaluating a kernel functions on a pairs of objects is equivalent to measuring the similarity between those objects. Some of these similarity measures are normalized between 1 and

0. Once an "appropriate" kernel function is chosen and the kernel matrix K is formed, it can be incorporated into any kernel based classification, regression, or anomaly detection methods where the kernel matrix is the sufficient representation of the input data. In another word, by "kernelizing" any method we encode knowledge about the data, expressed in terms of pairwise similarities. This provides us with the opportunity to incorporate vast amount of knowledge from heterogeneous sources using particular kernel functions. This field of research is known as Multiple Kernel Learning (MKL) [6, 24]. MKL takes advantage of the mathematics of kernels allowing us to derive new kernels from existing ones, thereby using multiple kernels simultaneously. For any λ, if K_λ is a Mercer kernel then it can be easily shown that for positive coefficients, a weighted combination of kernels (Eqn. 1.10) also preserves the Mercer kernel properties, i.e. $\hat{K}$ is also a Mercer kernel.

$$\hat{K} = \sum_{\lambda} \beta_\lambda K_\lambda \qquad \lambda \in \mathcal{R}_+ \tag{1.10}$$

A common practice is to use a convex combination (i.e. $\sum_\lambda \beta_\lambda = 1$) of various kernels which may be constructed on very different feature sets such as color, shape, texture etc. Therefore a major advantage of the multiple kernel learning approach is its ability to incorporate more knowledge in the decision process while analyzing complex heterogeneous systems that involve various data sources and data structures.

1.4.3.5 One-Class Support Vector Machines: An Overview

Here we provide here a brief overview of the one-class SVMs, followed by some descriptions of a case study for handling heterogeneous data. Let us consider situations where the users have ideas about what could be the normal behavior of the system (or data) as opposed to the nature of abnormal patterns. Unsupervised techniques like one-class SVM are a perfect fit to such an anomaly detection problem. One-class SVM is designed o estimate the density of the data. It builds a model on single (known) class of data and then finds a set of outliers using a decision boundary—a hyperplane that separates these outliers from the rest of the training examples. Schölkopf [39] showed that in the high dimensional feature space it is possible to construct an optimal hyperplane by maximizing the margin between the origin and the hyperplane in the feature space by solving the following primal optimization problem,

$$\begin{aligned} \text{minimize} \quad & P\left(\mathbf{w}, \rho, \xi_i\right) = \frac{1}{2} w w^T + \frac{1}{\nu \ell} \sum_{i=1}^{\ell} \xi_i - \rho \\ \text{subject to} \quad & \left(\mathbf{w}.\phi(x_i)\right) \geq \rho - \xi_i, \quad \xi_i \geq 0, \quad \nu \in [0, 1] \end{aligned} \tag{1.11}$$

where ν is an user-specified parameter that defines the upper bound on

the training error, and also the lower bound on the fraction of training points that are support vectors, ξ is the non-zero slack variable, ρ is the offset, $\phi(x_i)$ represents the transformed image of x_i in the Euclidean space and $i \in [\ell]$. The origin is the only representative of the second class with negative label. For multiple kernels, the dual form of the optimization can be written as,

$$\begin{aligned} \text{minimize} \quad & Q = \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j(\sum_{\lambda}\beta_\lambda K^\lambda_{i,j}) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq \frac{1}{\ell\nu}, \sum_i \alpha_i = 1, \sum\lambda = 1, \rho \geq 0, \quad \nu \in [0,1] \end{aligned} \tag{1.12}$$

where β_λ are the weights of the kernels and the α_i are Lagrange multipliers. Once this problem is solved at least $\nu\ell$ training points with non-zero Lagrangian multipliers ($\vec{\alpha}$) are obtained and these points $\{x_i : i \in [\ell], \alpha_i > 0\}$ are called *support vectors*. The selected points can be marginal $\mathcal{I}_m = \{i \ : \ 0 < \alpha_i < 1\}$ and non-marginal $\mathcal{I}_{nm} = \{i \ : \ \alpha_i = 1\}$ support vectors. Once $\vec{\alpha}$ is obtained, SVMs compute the following decision function.

$$f(\vec{x}_z, \alpha, \beta, \rho) = sign(\sum_{i\in\mathcal{I}}\alpha_i(\sum_{\lambda}\beta_\lambda K^\lambda_{i,z}) - \rho) \tag{1.13}$$

where $\mathcal{I} = \mathcal{I}_m + \mathcal{I}_{nm}$. The key aspect of this formulation is that many training exampels are no longer needed—only the support vectors are used to define the function. If the decision function predicts a negative label for a given test point x_j, then it is classified as an outlier. Test examples with positive labels are classified as normal. The pseudo code of one class SVMs is shown below.

Algorithm 2 Single Class SVMs Algorithm

1: Input Vector: $X = \{x_1, x_2....x_m, z\}$, $X \in \mathcal{R}^d$.
2: Map Features: $\sum_\lambda \beta_\lambda K^\lambda_{i,j}$).
3: Solve Eqn. 1.12 to obtain α corresponding to Support Vectors (SVs).
4: Calculate bias, $\rho = \sum_{k=1}^{Ns}\alpha_k(\sum_\lambda \beta_\lambda K^\lambda_{\vec{x},\vec{x}_k})$.
5: Calculate score, $f(\vec{z}) = \sum_{k=1}^{Ns}\alpha_k(\sum_\lambda \beta_\lambda K^\lambda_{\vec{z},\vec{x}_k})$.
6: **if** $f(\vec{z}) > \rho$ **then**
7: return 1
8: **else**
9: return 0
10: **end if**

1.4.3.6 An Example of Multiple Kernel Anomaly Detection

The whole detection process is nothing but a systematic way of looking at events in the collected data, analyzing the extracted knowledge, and finally making some decisions using certain criteria. As researchers we may gain a better understanding of why a certain event happened as it did, and what might be causing it. We give here an example that focuses on the use of data analysis techniques to identify anomalies on systems where the data is recorded in both the continuous and discrete form across its subsystems. We chose to elaborate on this example because it is the only example we are aware of that utilizes multiple kernel learning for anomaly detection over heterogeneous data. We feel this method has great potential to improve anomaly detection for engineered systems such as commercial aircraft.

Here we consider a scenario where the order of switching is absolutely important as compared to the magnitude. Here, the sequential nature of the discrete switches influences the dynamics of the system and hence constitute the main driving factors in the measured continuous output parameters of that system. For an algorithm to find anomalous behavior in such systems it should be able to detect anomalies in both the continuous and discrete sequences simultaneously. Anomaly types that may appear in such continuous and discrete sequences are shown in figure 1.4.3.6. By continuous sequence we mean the quantized version of the continuous variables arranged in a sequence form. Discrete sequences are representations of the order in which the transitions of the switches happens. The details on the preprocessing and feature extraction steps are scripted in [16].

A synthetic data set was generated to simulate the scenario where the discrete switches are the driving factors in the measured continuous parameters. The data generation method allows the continuous parameters to vary directly with the state of the binary inputs. In addition to this, Types I, II, III, & IV anomalies were randomly injected in the synthetic data. Ten binary parameters were generated with three fundamental behaviors: random flipping, constant throughout, and deliberate switching. One parameter was set to randomly switch between 0 and 1, while two parameters never changed states. For the deliberate switching six channels would hold a value at their initial state and then change to the alternate state when a separate channel toggled from 0 to 1. Apart from all the above three, abnormal patterns which are independent of discrete variables were injected arbitrarily in certain parts of the continuous data. This serves as an excellent platform where different algorithms could be tested to demonstrate their ability to detect anomaly of each type and for comparing the performances of multiple algorithms.

For comparison to MKL we choose Orca and SequenceMiner which, with their own strengths, are strong contenders for detecting a variety of anomalous conditions. SequenceMiner [11] was originally designed to model binary switch inputs from pilots during important periods in flight. The switch transitions for a given flight are formatted into a single sequence of switches. Each

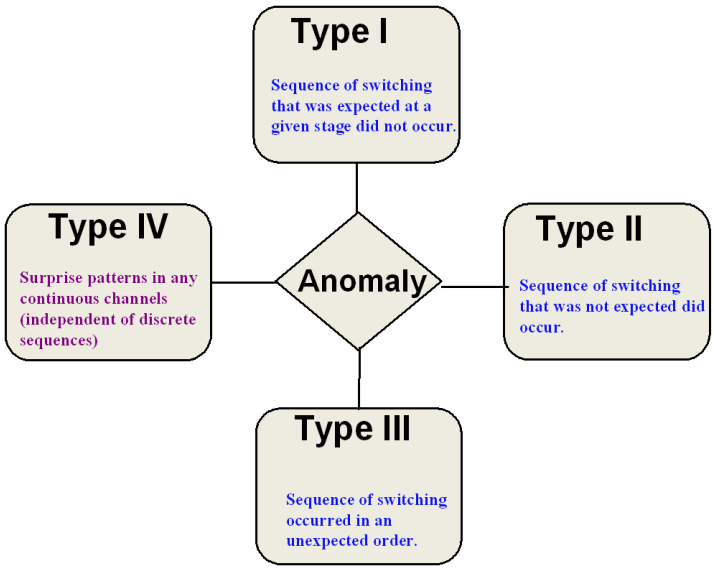

FIGURE 1.5: The figure represents the summary of the different abnormality categories injected in the synthetic data. A total of 12 faults have been randomly injected, out of which 3 are continuous (Type IV) and 9 are discrete (Type I, II and III).

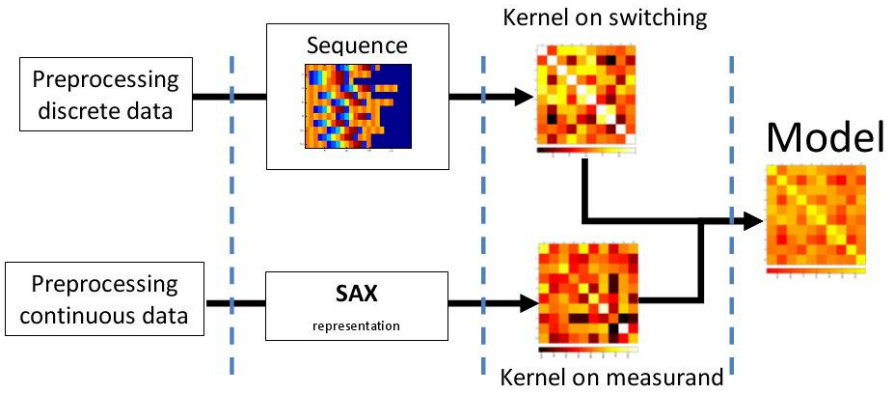

FIGURE 1.6: This figure represents three stages of kernel based anomaly detection technique. The first stage being the data preprocessing & refinement, the second stage is information fusion followed by the final stage which is model development.

sequence of switches is compared against other sequences by using the longest common subsequence (LCS) as the metric for comparison. Sequences that are similar are clustered together. Outliers are sequences that have very low LCS values. Since sequenceMiner takes into account the order in which the switches were triggered it has the ability to identify anomalies in the temporal domain, however it is unable to handle continuous data and therefore does not have the ability to detect anomalies in continuous parameters. Both Orca and sequenceMiner will be compared with multi-kernel one-class SVMs.

In using multi-kernel one-class SVMs, as an initial step we preprocessed the data and then generate continuous and discrete features. We used a convex combination of two kernels with equal weights. The resultant kernel takes the form of,

$$K(\vec{x}_i, \vec{x}_j) = \eta K_d(\vec{x}_i, \vec{x}_j) + (1 - \eta) K_c(\vec{x}_i, \vec{x}_j) \tag{1.14}$$

where K_d is a kernel over discrete sequences constructed using the normalized Longest Common Subsequence (nLCS) metric, K_c is a kernel constructed over the Symbolic Aggregate approXimation (SAX) [2] representation of continuous data, i.e continuous sequences, using the same nLCS metric, and η is used to weight the two kernels. From our earlier discussions, one can understand that the constructed kernel K, as well as K_d and K_c are all symmetric positive semi-definite matrices. MKL appears to be a promising way to meet our requirement of incorporating knowledge of both discrete and continuous sequences in anomaly detection as this method aims to combine both strengths of Orca and SequenceMiner into a single approach to allow for detection of a variety of anomalies. The different steps of multiple kernel one class SVMs is shown in figure 1.6.

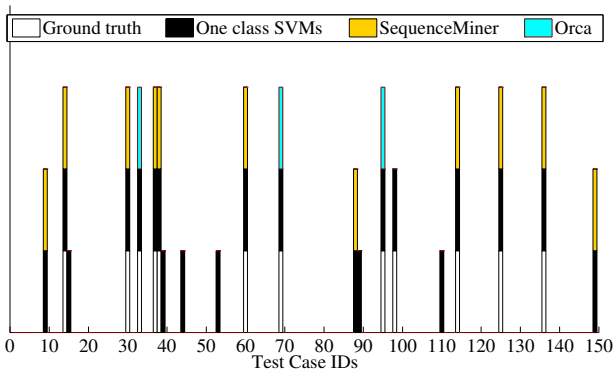

FIGURE 1.7: The figure represents the summary of the performance of all three algorithms in detecting the abnormalities in the synthetic data for each abnormality category. A total of 12 abnormal candidates have been randomly injected, and these abnormal cases are represented by the ground truth. Multiple kernel one class SVMs was the only algorithm to detect all fault types.

Figure 1.7 summarizes the outcomes. Since the actual fault injection incidents are known, we are able to evaluate the performance of all algorithms in detecting those faults. Out of twelve injected faults, Orca was able to find the three continuous anomalies. Even though Orca can handle both discrete (binary) variables and continuous variables, the algorithm is unable to detect sequential anomalies where the ordering of transitions is embedded in some form. SequenceMiner, using one standard deviation threshold calculated from the reference set, was able to detect most of the discrete anomalies and clearly missed all the continuous anomalies. Whereas the multiple kernel one class SVMs stand out among all the algorithms since it was able to identify all twelve fault types (both discrete and continuous).

1.4.4 Text Mining

1.4.4.1 Data and Algorithms

As mentioned in section 1.2.1.2., our example text repository, ASRS, has two relevant problems: classification and topic extraction. We now discuss two algorithms developed and implemented to analyze and extract useful information from ASRS reports. The first is Mariana [30], a Support Vector Machine with Simulated Annealing, which is a search method used to find the best hyperparameter settings for the model. Since SVMs were explained earlier, we will not do so here. The second method is classification built on top of Non-negative Matrix Factorization (NMF), which attempts to find a document model that represents document features that add up in various

combinations to form documents. NMF has the potential to provide basis vectors that are interpretable and indicative of different topics present within the repository. This is critical to discovering new, previously undiscovered problem areas within ASRS and other repositories.

Non-negative Matrix Factorization (NMF) [25] is a variation on the host of mathematically motivated techniques for factoring large vector-valued data sets into basis and distribution matrices. Suppose we have d documents and t terms.[1] The general approach is to seek a relatively small set of k basis vectors represented by the t x k matrix W, and a corresponding set of distribution weight vectors represented by the k x d matrix H, such that the transposed bag-of-words matrix X (for NMF, we assume X is t x d rather than the usual convention of being d x t) is factored according to $X \approx W * H$ by minimizing some measure of the difference, $X - W * H$. The hope is that the basis vectors in W will correspond to some fundamental properties of the data set, with the distributions in H combining those properties to form the data.

The convention in NMF-based text analysis is that the jth column of X represents the term weights of document j, each column of W is a basis vector over the term set, and each column of H is a set of weights over the basis vectors. Thus, if we use $H_{\bullet j}$ to denote the jth column of H and $H_{i\bullet}$ to denote the ith row of H, then for the jth document, $W * H_{\bullet j} \approx X_{\bullet j}$. If each basis vector $W_{\bullet b}$ is individually L1 normalized to a single term weight, and the rows $H_{b\bullet}$ are inversely scaled to maintain the product $W_{\bullet b}H_{b\bullet}$, then H_{bj} is the approximate number of terms, from basis vector $W_{\bullet b}$, found in the jth document. If the columns $H_{\bullet j}$ are then $L1$-normalized, they give the relative basis vector weights, independent of document size.

In NMF applications, the data values are non-negative, typically counts or scalar measurements, and the factorization is constrained to keep both W and H non-negative. With non-negativity, the basis vectors may be thought of as components, and the distributions as recipes for adding components to match the data. Both aspects seem more natural than the alternatives, particularly so for intrinsically non-negative data.

While non-negativity is an appealing property for factorization methods, constraining conventional difference minimization algorithms to maintain non-negativity has mostly been difficult. This changed in 2002, with spreading recognition of the potential of Lee and Seung's multiplicative update approach [25, 37]. For the squared Frobenius norm, the standard sum of squared matrix values $\|X - WH\|_F^2$, Lee and Seung's original paper gives the minimizing reestimation equations for iteration as:

$$W_{ab} \leftarrow W_{ab}\frac{(X * H^T)_{ab}}{(W * H * H^T)_{ab}} \tag{1.15}$$

$$H_{bi} \leftarrow H_{bi}\frac{(W^T * X)_{bi}}{(W^T * W * H)_{bi}} \tag{1.16}$$

[1] in general, NMF can be used in the case where there are d examples that have t features each, but we present NMF as used for text classification.

where a and i index over the attributes and instances of X, respectively, and b indexes over the basis vectors. This is actually a reformulation of the standard gradient driven norm minimizing search, augmented with a conceptually simple step size computation that maintains the non-negativity constraint.

Starting with non-negative W and H, and applied alternately, these reestimation equations are proven to monotonically lower the norm toward a local stationary point, while maintaining the non-negative properties of W and H. Lee and Seung also provided an alternate multiplicative minimization for the Kullback-Liebler divergence of probability matrices, and [17, 41] describe schemes for other matrix norms.

In factoring the prepared data, there is the fundamental choice of what function of $X - WH$ shall be minimized with consequences that are not yet well understood. A basis size must be chosen, or a range of sizes searched over and evaluated. Algorithmic details, particularly factor initializations, may have significant effects. Since the multiplicative NMF algorithms are gradient driven, they approach their stationary points at exponentially decreasing rates. This requires stopping criteria that balance the opposing requirements of computational efficiency and numerical accuracy.

The columns of W form vectors of words, each of which can be seen as a topic represented in the text repository. If one wants to use NMF for classification, then a linear model is fit using the criterion that the product of the H matrix and the document categorization matrix is the same for the training and the test set. Since the H matrices for the training and test data are known and the categorization matrix for the training set is known, one can solve for the categorization matrix for the test data.

1.4.4.2 Results

We first discuss text classification results. The authors of [12] tested Mariana with raw text in BOW format and the same text after being processed by an NLP system called PLADS to assess how useful PLADS and the simulated annealing within Mariana would be. Mariana's process of choosing better hyperparameters improves the area under the ROC curve by 10% or more in some of the categories on raw text. It also gives equal improvement on text that has been processed by the PLADS system. As shown in figure 1.8, there is little difference in performance when this process is used on raw text and PLADS processed text. Others have also found that NLP methods do not give enough performance improvement to justify their high running time.

The authors of [12] then evaluated the performance of Mariana by autoclassifying 100 randomly selected reports and having the results reviewed by a problem report expert. The reviewer agreed with the top classification by Mariana 73% of the time. The reviewer agreed with one or both of the top two classifications 86% of the time, and with the top three classifications 90% of the time. A separate review of the 100 reports was done by another subject expert. Then the first reviewer reviewed the second reviewer's classifications,

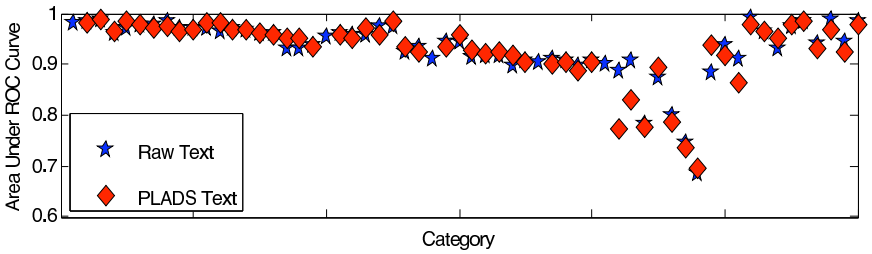

FIGURE 1.8: Comparison of Raw Text and PLADS Processed Text using Optimum Hyperparameters

and agreed with their top (and only) classification 89% of the time. These results indicate that Mariana is classifying documents reasonably well and in a manner reasonably consistent with human experts given that reviewers (including presumably those who provided the original classifications to the documents that they used in training) disagree among themselves.

Figure 1.9 shows the areas under the ROC curves for Mariana, NMF, and a baseline method (Linear Discriminant Analysis) applied to the ASRS categories, as well as the fraction of documents in each category (the boxes), normalized so that the category with the largest number of documents has a value of 1. The categories on the x-axis are sorted in ascending order of this normalized fraction. The vertical lines connect the highest area to the lowest area for each category and are meant to facilitate comparisons within each category. Overall, Mariana outperforms Linear Discriminant Analysis on 16 of the 22 categories and NMF outperforms Linear Discriminant Analysis on 21 of the 22 categories—however, not by a great amount.

We now discuss NMF's results for topic extraction. Table 1.1 gives examples of two basis vectors from three runs of NMF (each with a different training and test set mix drawn from the original dataset) on the ASRS repository. One basis vector was drawn from each of the three runs such that the resulting three basis vectors were the closest such triple in terms of $L1$-norm—these form the three left columns of table 1.1. The three right columns are the basis vectors corresponding to the second closest such triple. Within each column, the words are given from top to bottom in order of weight. One can see from these examples that the three runs came up with bases that are relatively close to one another. The two triples are also quite different from each other, indicating that the basis vectors represent different topics (combinations of categories). However, the 20 words within a basis vector are clearly quite related. The first basis set clearly seems to be describing some fuel tank related problem and the second basis set clearly describes an issue related to a repair. It appears that NMF is finding the significant collocations that are indicative of topics and keeping them together.

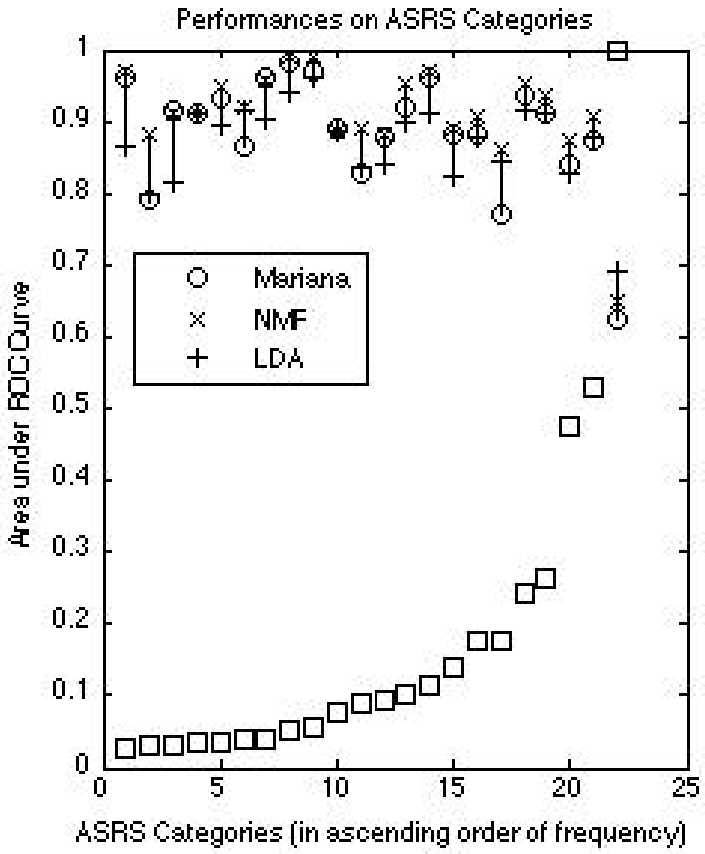

FIGURE 1.9: Performances on ASRS categories.

1.5 Dealing with efficiency issues: Distributed Anomaly Detection

Often, data representing the operations of fleets are distributed over many storage devices. This happens not only because of the substantial amount of data which is more than typically can be stored at one location, but also because of the geographically distributed nature of the data generation process, which makes storing the data at locations close to where they were generated convenient. For example, one can imagine storing the data from a commercial aviation fleet at the destination airports of the corresponding flights. That is, every time an aircraft lands, its data may be uploaded to a storage device located at that airport. Consolidating the data may be impractical given how

much data there is. Therefore, anomaly detection algorithms that can operate on the data even though they are distributed, but give the same results as what a traditional algorithm would give after centralizing all the data, are clearly useful.

We briefly give overviews of two algorithms for which there are mathematical proofs that they yield the same anomalies with the same ranking of anomalousness on any distributed dataset as what a corresponding centralized algorithm would give after centralizing all the data. Additionally, these algorithms are designed to be efficient in that they require much less communication than what would be required to actually copy all the data from the different nodes to one location. One algorithm is a distributed expectation maximization (EM) algorithm for learning Gaussian Mixture Model (GMM) parameters. The second algorithm is a distance-based anomaly detection algorithm.

1.5.1 GMM

Here, we discuss a density-based anomaly detection algorithm. This is an example of a class of algorithms that learn a density function over the training dataset. This density function can clearly be used for many purposes, but when used for anomaly detection, the user assumes that normal data tend to be in higher density regions whereas anomalous data tend to be in lower density regions. The density may be learned using non-parametric methods, such as histogram methods or methods to construct Voronoi tesselations of the data [38]. A parametric method may also be used, whereby a model structure is assumed and the learning task involves calculating the parameters of the model that best fit the training data. Either way, when new data arrive, the learned density function is calculated on these points and those for which the probability density is lower than a certain threshold may be judged to be anomalous data. The algorithm we discuss in this section is an example of a parametric method.

This algorithm, called PeDEM [9] learns the parameters of a GMM on the entire distributed dataset, monitors whether the GMM's parameters are still valid given new data that has arrived at each of the nodes of the distributed system, and, if necessary, updates the GMM's parameters to reflect the inclusion of the new data in the distributed repository.

Expectation Maximization (EM) is an iterative optimization technique to estimate unknown parameters Θ (typically sufficient statistics for the probability distribution being used to model the data) given data D. The EM algorithm alternates between two steps to maximize the posterior probability distribution of Θ given D. The algorithm utilizes hidden variables H, which often are assumed to represent hidden state information. The two steps are

- E-step: Calculate the expected value of H given Θ and D.

- M-step: Calculate Θ to maximize the likelihood of the data D given the expected value of H.

For GMMs, the generative model for a vector $\mathbf{x} \in R^d$, where $\mathbf{x}$ is a d-dimensional vector of real numbers, is

$$P(\mathbf{x}) = \sum_{j=1}^{M} \pi_j P(\mathbf{x}|j),$$

where M is the number of Gaussians in the mixture, π_j is the prior probability of the jth mixture being used to generate a data point, and the jth mixture's density is

$$P(\mathbf{x}|j) = \frac{1}{(2\pi)^{d/2}\|\mathbf{C}_j\|^{1/2}} exp-(\mathbf{x}-\mu_\mathbf{j})^T \mathbf{C}_j^{-1}(\mathbf{x}-\mu_\mathbf{j})/2.$$

Each density is parameterized by its mean vector $\mu_\mathbf{j} = [\mu_{j.1}\mu_{j,2}\ldots\mu_{j,d}]$ and its covariance matrix $\mathbf{C}_j = (\mathbf{x}-\mu_\mathbf{j})(\mathbf{x}-\mu_\mathbf{j})^T$. Assuming that we have a training dataset $X = \{\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}\}$, EM attempts to estimate the model parameters such that they maximize the likelihood of the parameters given the data. Normally, the log-likelihood is maximized instead, which gives the same result since logarithm is a monotonically increasing function.

A centralized version of EM for GMMs would have the following E-step:

$$q_{j,i} = \frac{\pi_j N(\mathbf{x_i}; \mu_\mathbf{j}, \mathbf{C_j})}{\sum_{k=1}^{M} \pi_j N(\mathbf{x_i}; \mu_\mathbf{j}, \mathbf{C_j})}.$$

and the following M-step:

$$\begin{aligned}
\pi_j &= \frac{\sum_{i=1}^{n} q_{j,i}}{n} \\
\mu_\mathbf{j} &= \frac{\sum_{i=1}^{n} q_{j,i}\mathbf{x_i}}{\sum_{i=1}^{n} q_{j,i}} \\
\mathbf{C_j} &= \frac{\sum_{i=1}^{n} q_{j,i}(\mathbf{x_i}-\mu_\mathbf{j})(\mathbf{x_i}-\mu_\mathbf{j})^T}{\sum_{i=1}^{n} q_{j,i}},
\end{aligned}$$

for each of the M Gaussians in the Gaussian mixture. $N(\mathbf{x_i}; \mu_\mathbf{j}, \mathbf{C_j})$ denotes the probability density function at data point $\mathbf{x_i}$ under a normal distribution with mean $\mu_\mathbf{j}$ and covariance matrix $\mathbf{C_j}$.

The corresponding version of EM in PeDEM has the following E-step:

$$q_{a,j,i} = \frac{\pi_j N(\mathbf{x_{a,i}}; \mu_\mathbf{j}, \mathbf{C_j})}{\sum_{k=1}^{M} \pi_k N(\mathbf{x_{a,i}}; \mu_\mathbf{k}, \mathbf{C_k})}.$$

and the following M-step, where B is the number of nodes in the distributed system:

$$\begin{aligned}
\pi_j &= \frac{\sum_{a=1}^{B}\sum_{i=1}^{n} q_{a,j,i}}{\sum a = 1^{B} N_a} \\
\mu_{\mathbf{j}} &= \frac{\sum_{a=1}^{B}\sum_{i=1}^{n} q_{a,j,i}\mathbf{x_{a,i}}}{\sum_{a=1}^{B}\sum_{i=1}^{n} q_{a,j,i}} \\
\mathbf{C_j} &= \frac{\sum_{a=1}^{B}\sum_{i=1}^{n} q_{a,j,i}(\mathbf{x_{a,i}} - \mu_{\mathbf{j}})(\mathbf{x_{a,i}} - \mu_{\mathbf{j}})^T}{\sum_{a=1}^{B}\sum_{i=1}^{n} q_{a,j,i}},
\end{aligned}$$

The E-step computation for each node involves only data and parameters available at that node. However, the M-step computation at each node requires information from all the other nodes. Obviously, the distributed anomaly detection problem would be trivial if all the computations at each node only involved information available at that node.

The goal of the exercise in [9] was to continually check that the current parameters of the GMM are valid with respect to the global data set even as new data are added and to update the parameters when they are incompatible with the data. In PeDEM, four monitoring problems are solved—one each for the log-likelihood ($\bar{L}$), prior probabilities of the mixture models, the means of the mixture models, and the covariances. In particular, they monitor for

$$\begin{aligned}
\bar{L}(\hat{\Theta}|G) &> \epsilon \\
\\
|\pi_j - \hat{\pi_j}\| &< \epsilon_1 \\
\|\|\mu_{\mathbf{j}} - \hat{\mu_{\mathbf{j}}}\|\| &< \epsilon_2 \\
\left|||\mathbf{C_j}|\,|_F - \hat{\mathbf{C}}_{\mathbf{j}}\right| &< \epsilon_3.
\end{aligned}$$

In case of log-likelihood, clearly one would want the log-likelihood to be high—if it drops below a threshold, then that means that the new data is too different from the data seen so far for the current model to be valid, so model updating is needed. For the remaining three conditions, if the difference between the last stable value calculated based on the global data and what is estimated at a node based on current data at a given node, then model updating is needed. More details on the algorithm and the proof that the distributed algorithm returns the same results as the centralized EM algorithm are shown in [9]. However, in summary, the proof relies on a notion of covering the global data with convex regions where if each node's knowledge (the data stored at that node plus any data communicated to it by other nodes) are contained within one such convex region, then the results that each node returns would be the same as what would be returned by a centralized algorithm. This condition means that the nodes do not need to communicate until

the condition is violated. The authors of [9] demonstrated experimentally that the level of communication required for the algorithm is quite low assuming that the data is not changing wildly at every node.

1.5.2 Distance-based

In distance-based anomaly detection methods, data points for which neighboring data points are relatively nearby are judged to be normal, whereas those with relatively distant neighbors are judged to be anomalous. A variety of distance functions is used, such as distance to the kth nearest neighbor for some chosen k, and average distance to the k nearest neighbors, Given a dataset, one can calculate this distance function for every data point and rank order the data points in decreasing order of the distance function. The data points with high values for the distance function can be returned as examples of anomalies.

A naïve algorithm for this task would require calculation of the distance between all pairs of data points. In particular, for a given data point x, its distances to all the other data points seemingly need to be calculated in order to determine its k nearest neighbors, which in turn can be used to calculate the distance function. However, there is one restricted scenario in which all pairs of points do not need their distances calculated. This idea has been implemented within Orca [8]. The algorithm takes as input a dataset and a number L and returns the top L anomalies in the dataset by one of several metrics such as distance to the kth nearest neighbor or average distance to the k nearest neighbors—that is, the points having the L highest values of the chosen metric are returned as anomalies. Orca starts running in the naive way by calculating the distance from the first data point to all the remaining data points, calculating the chosen metric for that data point, calculating the distance from the second data point to the following data points, etc. However, Orca keeps track of the top L anomalies and their values for the metric. Note that while a data point is being examined, its value for the metric can only decrease. [2] Therefore, if the value of the metric for the current data point being examined drops below the value of the metric for the Lth most anomalous data point found so far, then the current data point cannot be an anomaly, so we can stop calculating distances for the current point and move onto the next one. This happened because we have already determined that there are at least L data points with higher values for the metric than the current point being considered. Orca has been demonstrated experimentally to have a running

[2] For example, suppose we are calculating the metric for the first data point, have checked its distances to the next 20 data points, and are finding points based on distance to the 10th nearest neighbor. In this case, the 22nd data point can either be closer than the current 10th nearest neighbor or farther away. If the 22nd data point is farther away, then the distance to the 10th nearest neighbor will not change. If it is closer, then either the 22nd point or the current 9th nearest neighbor will become the 10th nearest neighbor and the distance to the 10th nearest neighbor will decrease.

time that is, on average, slightly greater than linear in the number of data points submitted, although its worst-case running time is quadratic in the number of data points just like the naïve algorithm.

For distance-based distributed anomaly detection algorithms to return the same anomalies as the centralized algorithm discussed above, we depend on this property that adding new data points can only reduce a given data point's value for a metric. This is known as the anti-monotonicity property [21]. More formally, assume that the anomaly detection algorithm A takes as input a finite set of points P, a function $R : D \times 2^D \rightarrow R^+$, and a positive integer L, and returns the data points having the L highest values of R. Given a data point $x \in D$ and two sets of data points $P_1 \subseteq P_2 \subseteq D$, we require the following two properties

- Anti-monotonicity: $R(x, P_1) \geq R(x, P_2)$.
- Smoothness: if $R(x, P_1) > R(x, P_2)$, then there exists $y \in P_2$ P_1 such that $R(x, P_1) > R(x, P_1 \cup z)$.

According to the theorems and proofs in [21], only the anti-monotonicity property is needed to prove that the distributed algorithm will terminate in finite time with all the nodes agreeing on the anomalous points and their supports (the nearest neighbors needed to calculate the metric). Both anti-monotonicity and smoothness properties are needed to prove that the agreed upon anomalies are the globally correct ones—that is, the ones that would have been identified by the corresponding distance-based anomaly detection acting on all the data after being collected at one site.

1.6 Conclusions

It may be noticed that in this chapter, the detection problem was formulated with an emphasis on how to handle heterogeneous knowledge sources to obtain a meaningful solution in a more robust way. The data-driven concept for developing heterogeneous anomaly detection models outlined in this chapter is an attempt to provide a broad sample of recent techniques with detailed coverage on a few algorithms to illustrate how they work but obviously, we are unable to cover all approaches in a single chapter. Readers are advised to consider that there is no single universally accepted approach that can perform extremely well on all possible datasets and anomalous conditions. The goodness of any algorithm depends a lot on its applicability which in most cases is very application specific. So far authors have developed a wide variety of machine learning tools and techniques and have demonstrated their applicability on various complex systems. Often it is seen that a particular approach is well suited to address certain problem. Also one cannot ignore

the importance of data refinement, cleaning, and transformation techniques in effective data mining and knowledge discovery. The nature of the algorithm decides how well and in what form this knowledge can be integrated in the core algorithm. We have already seen in previous examples that the ability to represent the data in the form of discrete and continuous sequences and to integrate this knowledge, with the notion of similarity between sequences, in the optimization problem of multiple kernel based technique, provides us with an opportunity to detect all different types of anomalies simultaneously. Moreover since the discrete and continuous sequences can be transformed into a single sequence of switches and are representative of "system level" information over time, another important feature of the multiple kernel anomaly detection algorithm is its ability to do "system-wide" analysis to detect anomalies. Even though improvements to the multiple kernel learning-based technique that we discussed may be found, it does point in the direction of the types of methods that we want—those that can handle heterogeneous sequence data.

TABLE 1.1: Examples of ASRS Basis Vectors. After performing three runs of NMF (each with a different initialization of the W and H matrices), we selected the three basis vectors (one column of W from each run) that were closest to each other in terms of $L1$-norm—these constitute the first three columns of this table. The next three columns are the second closest such triple of basis vectors.

TABLE 1.1: Examples of ASRS Basis Vectors. After performing three runs of NMF (each with a different initialization of the W and H matrices), we selected the three basis vectors (one column of W from each run) that were closest to each other in terms of $L1$-norm—these constitute the first three columns of this table. The next three columns are the second closest such triple of basis vectors.

FUEL	FUEL	FUEL
TANK	TANK	TANK
POUND	POUND	POUND
GALLON	GALLON	GALLON
GAUGE	GAUGE	GAUGE
PUMP	PUMP	PUMP
FUELTANK	BURN	BURN
BURN	FUELTANK	FUELTANK
FUELER	FUELER	FUELER
FUELQUANTITY	FUELQUANTITY	FUELQUANTITY
CENTER	CENTER	CENTER
MAINTANK	DISPATCH	FUELGAUGE
FUELGAUGE	FUELGAUGE	MAINTANK
IMBAL	MAINTANK	IMBAL
REFUEL	IMBAL	REFUEL
CROSSFEED	REFUEL	PLAN
QUANTITY	QUANTITY	CALCULATE
BALANCE	PLAN	CROSSFEED
CALCULATE	CROSSFEED	BALANCE
EMPTY	CALCULATE	EMPTY
INSTALL	INSTALL	INSTALL
INSPECT	INSPECT	REMOVE
REMOVE	REMOVE	REPLACE
REPLACE	MECHANIC	ENGINEER
MECHANIC	REPLACE	MANUAL
FOUND	PART	INSPECT
WORK	MANUAL	WORK
MANUAL	WORK	SHIFT
REPAIR	REPAIR	FOUND
PART	FOUND	ASSEMBLE
ENGINEER	SIGN	TECHNICIAN
TEST	ENGINEER	REPORT
CHECK	NUMBER	PANEL
SHIFT	SHIFT	REPAIR
SIGN	MAINTAIN	JOB
ASSEMBLE	TEST	XYZ
MAINTAIN	ASSEMBLE	BOLT
SERVE	AIRCRAFT	CARD
CARD	XYZ	LEAK

Bibliography

[1]

[2]

[3] B.G. Amidan and T.A. Ferryman. Apms svd methodology and implementation. Technical Report PNWD-3026, Battelle, 2000.

[4] F. Angiulli and F. Fassetti. Very Efficient Mining of Distance-based Outliers. In *Proceedings of CIKM '07*, pages 791–800, 2007.

[5] F. Angiulli and C. Pizzuti. Outlier Mining in Large High-Dimensional Data Sets. *IEEE Transactions on Knowledge and Data Engineering*, 17(2):203–215, 2005.

[6] F.R. Bach, G.R.G. Lanckriet, and M.I. Jordan. Multiple kernel learning, conic duality, and the smo algorithm. In *International Conference on Machine Learning*, 2004.

[7] S. D. Bay and M. Schwabacher. Mining Distance-based Outliers in Near Linear Time with Randomization and a Simple Pruning Rule. In *Proceedings of KDD'03*, pages 29–38, 2003.

[8] S.D. Bay and M.Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2003.

[9] Kanishka Bhaduri and Ashok N. Srivastava. A local scalable distributed expectation maximization algorithm for large peer-to-peer networks. In *Proceedings of the IEEE International Conference on Data Mining*, pages 31–40. 2009.

[10] David. Blei, Andrew Ng, and Michael Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[11] Suratna Budalakoti, Ashok N. Srivastava, and Matthew E. Otey. Anomaly detection and diagnosis algorithms for discrete symbol sequences with applications to airline safety. *Trans. Sys. Man Cyber Part C*, 39:101–113, January 2009.

[12] J.P. Castle, J.C. Stutz, and D.M. McIntosh. Automatic discovery of anomalies reported in aerospace systems health and safety documents. In *AIAA Infotech@Aerospace*. American Institute of Aeronautics and Astronautics, 2007.

[13] Banerjee A. Chandola V. and Kumar V. Anomaly detection: A survey. *ACM Comput. Surv.*, 41:1–58, 2009.

[14] O. Chapelle and P. Haffner. Support vector machines for histogram-based classification. *IEEE Transactions on Neural Networks*, 10(5):1055–1064, 1999.

[15] Thomas Chidester. Understanding normal and atypical operations through analysis of flight data. In *Proceedings of the 12th International Symposium on Aviation Psychology*, pages 239–242, May 1997.

[16] Santanu Das, Bryan L. Matthews, Ashok N. Srivastava, and Nikunj C. Oza. Multiple kernel learning for heterogeneous anomaly detection: algorithm and aviation safety case study. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '10, pages 47–56, New York, NY, USA, 2010. ACM.

[17] I.S. Dhillon and S. Sra. Generalized nonnegative matrix approximations with bregman divergences. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, pages 283–290. MIT Press, 2006.

[18] Z. Harchaoui and F.R. Bach. Image classification with segmentation graph kernels. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[19] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2001.

[20] Julian P T Higgins. Commentary: Heterogeneity in meta-analysis should be expected and appropriately quantified. *Int. J. Epidemiol.*, 37(5):1158–1160, 2008.

[21] Chris Giannella Ran Wolff Joel Branch, Boleslaw Szymanski and Hillol Kargupta. In-network outlier detection in wireless sensor networks. In *International Conference on Distributed Computing Systems*, pages 51–58. 2006.

[22] Hillol Kargupta, Vasundhara Puttagunta, Martin Klein, and Kakali Sarkar. On-board vehicle data stream monitoring using minefleet and fast resource constrained monitoring of correlation matrices. *Next Generation Computing*, 25(1):5–32, 2006.

[23] Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. Kernels for graphs. *Kernel Methods in Computational Biology*, 39(1):101–113, 2004.

[24] G.R.G. Lanckriet, N. Cristianini, L. El Ghaoui, P. Bartlett, and M.I. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5:27–72, 2004.

[25] D. D. Lee and H.S. Seung. Algorithms for non-negative matrix factorization. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems-13*, pages 556–562. MIT Press, 2001.

[26] Daniel D. Lee and H. Sebastian Seung. Algorithms for non-negative matrix factorization. In *NIPS*, pages 556–562, 2000.

[27] Markos Markou and Sameer Singh. Novelty detection: A review - part 1: Statistical approaches. *Signal Processing*, 83:2003, 2003.

[28] Markos Markou and Sameer Singh. Novelty detection: A review - part 2: Neural network based approaches. *Signal Processing*, 83:2499–2521, 2003.

[29] NASA. Aviation safety reporting system (asrs) program overview.

[30] Nikunj C. Oza, J. Patrick Castle, and John Stutz. Clasification of aeronautics system health and safety documents. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(6):670–680, 2009.

[31] Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz Andre Barroso. Failure trends in a large disk drive population. In *Proceedings of the 5th USENIX Conference on File and Storage Technologies (FAST'07)*, Feb 2007.

[32] J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[33] John A. Quinn and Christopher K. I. Williams. Known unknowns: Novelty detection in condition monitoring. In *Proceedings of 3 rd Iberian Conference on Pattern Recognition and Analysis, Lecture Notes in Computer Science 4477. Springer-Verlag*, pages 1–6. Springer LNCS, 2007.

[34] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Rec.*, 29(2):427–438, 2000.

[35] RuleQuest Research. Gritbot, 2007. [Online; accessed 04-December-2010].

[36] Paul Robinson. A real-time quantitative condition alerting and analysis support system for aircraft maintenance. *NASA SBIR Phase II Abstract*. (from: http://www.atr-usa.com/documents/AeroTech$_Q$$CAAS_N$$arrative_B$$rief.pdf$).

[37] L.K. Saul and D. D. Lee. Multiplicative updates for classification by mixture models. In T.G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems-14*, pages 897–904. MIT Press, 2002.

[38] Jeffrey D. Scargle. A novel approach to novelty detection: Voronoi tesselation, machine learning seminar, nasa ames research center, 2004.

[39] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the Support of a High-Dimensional Distribution. *Neural Comput.*, 13(7):1443–1471, 2001.

[40] B. Schölkopf and A. Smola. *Learning with Kernels*. MIT Press, 2002.

[41] Y. Wang, Y. Jia, C. Hu, and M. Turk. Fisher non-negative matrix factorization for learning local features. In *Asian Conference on Computer Vision*. 2004.

[42] Wikipedia. Fibonacci number, the free encyclopedia, 2004. [Online; accessed 04-December-2010].

[43] Wikipedia. Hamming distance, the free encyclopedia, 2010. [Online; accessed 04-December-2010].

[44] H. Zhang, A.C. Berg, M. Maire, and J. Malik. Svm-knn: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition*, pages 2126–2136, 2006.