FILTERING AND REFINEMENT: A TWO-STAGE APPROACH FOR EFFICIENT AND EFFECTIVE ANOMALY DETECTION

BY

XIAO YU

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2011

Urbana, Illinois

Adviser:

Professor Jiawei Han

Abstract

Anomaly detection is an important data mining task. Most existing methods treat anomalies as inconsistencies and spend the majority amount of time on modeling normal instances. A recently proposed, sampling-based approach may substantially boost the efficiency in anomaly detection but may lead to weaker accuracy and robustness. In this study, we propose a two-stage approach to find anomalies in complex datasets with high accuracy as well as low time complexity and space cost. Instead of analyzing normal instances, our algorithm first employs an efficient deterministic space partition algorithm to eliminate obvious normal instances and generates a small set of anomaly candidates with a single scan of the dataset. It then checks each candidate with density-based multiple criteria to determine the final results. This two-stage framework also detects anomalies of different notions. Our experiments show that this new approach finds anomalies successfully in different conditions and ensures a good balance of efficiency, accuracy, and robustness.

To my family for all their love.

Acknowledgments

First, I would like to thank Professor Jiawei Han for his guidance and insightful comments for my research work. Second, I would like to thank all the faculties and colleagues in DAIS (Data and Information System) research group at the University of Illinois, Urbana-Champaign. Especially, I thank Lu An for their help throughout the writing of the thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Anomaly detection, which could be informally defined as finding instances in a dataset with unusual properties, or patterns with unexpected behaviors, has been an important topic in data mining research, with broad applications. Usually, anomaly detection methods are studied or employed to minimize or eliminate negative influences on normal instances in datasets, but data mining researchers realize that besides data cleaning, anomaly detection could also be used to indicate hidden and critical information in various datasets and domains, such as computer network intrusions [5], fraudulent credit card usages [2], temporal anomalies in traffic data [9], as well as disease diagnosis in medical image analysis [12].

Anomaly detection has been studied from multiple technical aspects, and a number of detection algorithms have been proposed in both supervised and unsupervised fashion [1, 13, 10, 14]. These approaches have different advantages and drawbacks, and moreover, the notions of anomaly are also defined under specific scenarios, which makes these algorithms hard to be employed in general applications.

In general, for anomaly detection, a straightforward idea is to define a normal region based on various measures, like density of clusters or isolation degree. However, in complex datasets, the boundary between normal and abnormal instances is often not very clear. Moreover, in different domains, normal behavior could be evolving—a notion of normal behavior in current situation might be insufficient in the future. Therefore, if the major time and space are consumed on modeling normal instances, these approaches may not be suitable for large-scale real life applications, especially when there is a need to detect anomalies in *real time* in *gigantic data sets*, *dynamic data streams*, and/or other *fast changing environments*. Moreover, in different domains, degree of anomaly may indicate different semantic information, which needs to be accommodated in the algorithm as well. Due to the above challenges, the anomaly detection problem, in its most general

form, is not easy to solve.

Facing the above challenges, we believe a hybrid approach that utilizes different attributes of anomalies might be a qualified general solution in different scenarios. In this study, we propose a novel *filtering-and-refinement* approach for anomaly detection. To achieve highly accurate results with both efficiency and stability, we divide the anomaly detection process into two stages: *filtering* and *refinement*. First, a small set of anomaly candidates would be generated in sub-linear time in a deterministic way by *Deterministic Space Partition* (DSP). In this phase, the algorithm roughly separates normal instances and possible anomalies efficiently, by eliminating obvious normal instances, which usually present as highly coherent clusters, with a high confidence. Then, in the second phase, density-based measures are applied as *refinement*, which leads to the generation of stable and accurate final results, with a relatively high time complexity but only on the anomaly candidates. This method also generates attributes which describe the degree of outlying. Therefore, by splitting the anomaly detection process into two stages, we could take advantages of different methods, and generate better detection results with lower time complexity.

Compared with the existing popular anomaly detection methods, our approach eliminates the majority of normal instances in the first stage efficiently, and then focuses on describing the boundary of normal instances and anomalies in a more accurate and specific fashion. The time complexity of the second stage is $O(s^2)$, where s is the number of anomaly candidates. Based on our experiments, more than 70% of normal instances could be filtered out in the first stage. So the overall time complexity is low. The two-stage approach makes the best use of both time and space, and also ensures the accuracy of the final results.

This paper is organized as follows. In Chapter 2, we introduce related work on anomaly detection and compare different methodologies. In Chapter 3, we describe our anomaly candidate generation method and its characteristics. In Chapter 4, we provide a density-based method to extract final anomalies from all candidates. The comparison experiments on extensive datasets are analyzed in Chapter 5; and finally, our conclusions are resented in Chapter 6.

Chapter 2

Related Work

Traditional anomaly detection methods [1, 3, 8, 13, 10, 14], including supervised approaches, statistic approaches and also distance-based methods, etc, are proposed with different anomaly measures and notions. Each of these methods has advantages and drawbacks, so most of them can only solve a specific formulation of the problem [4, 7].

Statistic approaches [1] are mostly distribution based, *i.e.*, using a standard distribution to fit the dataset. These methods are impractical in real-world applications because it's difficult and sometimes impossible to generate accurate distribution for each dataset.

Distance-base methods [8] measure the distance or relative distance from an object to the rest instances in datasets. Extended methods, often referred as density-based methods [3], can detect local anomalies with high confidence and accuracy. However, the time complexity of these methods is high and can hardly be realistic at handling large-scaled or high dimensional datasets.

Another category of anomaly detection methods, which is relatively new, is to detect anomalies by exploring a spatial partition structure, such as Random Forest [13] or Isolation Forest [10]. One interesting observation is that anomalies are more likely to be data objects with smaller depths in these partitioning structures. Moreover, they employ sub-sampling or random partition methods to speed up the search process. Generally, these methods can achieve near linear time complexity and have the potential to perform nicely in high dimensional data, however, the random feature also reduces the accuracy and stability of the detection results. Isolation Tree (iTree) is a random space partition index data structure newly proposed by Liu *et al.* [10]. To construct iTree, random space partition process needs to be repeated recursively until all instances are isolated. During this process, instances with distinguishable attribute values are more likely to be separated in early partitions, leading to shorter paths in iTree structure.

One good characteristic of random space partition process is, instead of examining the whole

Type	Time	Space	Accuracy	Stability
statistical	various	Low	Low	High
density	High	High	High	High
iForest	Low	Low	Medium	Low
DSP	Low	Low	High	High

Table 2.1: Summary of anomaly detection methods.

dataset, it works well on a sample of the original dataset, because this method has an assumption that even small samples of dataset could preserve the boundaries of normal instances and anomalies in the original dataset. Random factors improve the efficiency of the algorithm. The algorithm does not spend any time on distance or density calculation; instead, it generates anomaly criteria and separates instances in an arbitrary fashion. This simplifies the problem and makes the original problem solvable in near linear time. However, the side effect of the random factors is obvious: by replacing traditional distance- or density- based criteria with random selection and generation, the precision and robustness of the algorithm are both decreased.

A number of clustering algorithms like DBSCAN [6, 11] are to some extent capable of handling anomalies, but these algorithms are designed to optimize clustering, rather than finding anomalies. So the efficiency and accuracy of anomaly detection in these algorithms are difficult to meet the scalability and fast response requirement in many data mining applications.

In our method, the notions of local anomaly share a few fundamental concepts with distance-based and space partition methods, and make the best use of these concepts to achieve a good performance with near linear time complexity.

Table 2.1 presents a summary of the methods in different categories in multiple criteria, where *accuracy* indicates the accuracy in complicated datasets with multiple distributions, different densities and local outliers.

Chapter 3

The Measure of Anomaly

Most traditional anomaly detection methods treat anomaly as a binary definition and work as unsupervised binary categorization algorithms. However, in real-world scenarios, it is insufficient to partition the whole datasets only into normal instances and anomalies. Further analysis and fine categorization of anomalies are required in many cases.

One such example is the animal species group analysis in biology. Dense clusters in geographic datasets of certain species often indicate fitting environment or resources that usually form nice habitats. Sparse clusters and their boundaries in these datasets could be used in analyzing annual migration activities, etc. But what are the anomalies in this case? Since such data sets are often collected by electronic devices, errors or mistakes could happen with a probability. This will lead to single or unique instances in the datasets, often referred as outliers by researchers. This kind of instances can be used to examine or tune detection devices, but do not have any significance in species group analysis. Another kind of anomalies that could be referred as *abnormal clusters* often have decent semantic meanings in the analysis. These clusters are usually far from normal instances yet somehow dense inside and can be used to measure the trend of species distribution and also to indicate slow environmental or climate changes.

From the above discussion, one can generalize the following three types of anomalies (Figure 3.1):

- *Unique Instances* (sparse and distant): A unique instance is an isolated point, far from the normal instances, and refers to rare and extreme cases;
- *Abnormal Clusters* (dense and distant): An abnormal cluster is far from the normal instances, but the density inside the cluster is close to the normal ones. Those clusters are quite useful in several applications since they have repeated appearances and are more creditable than

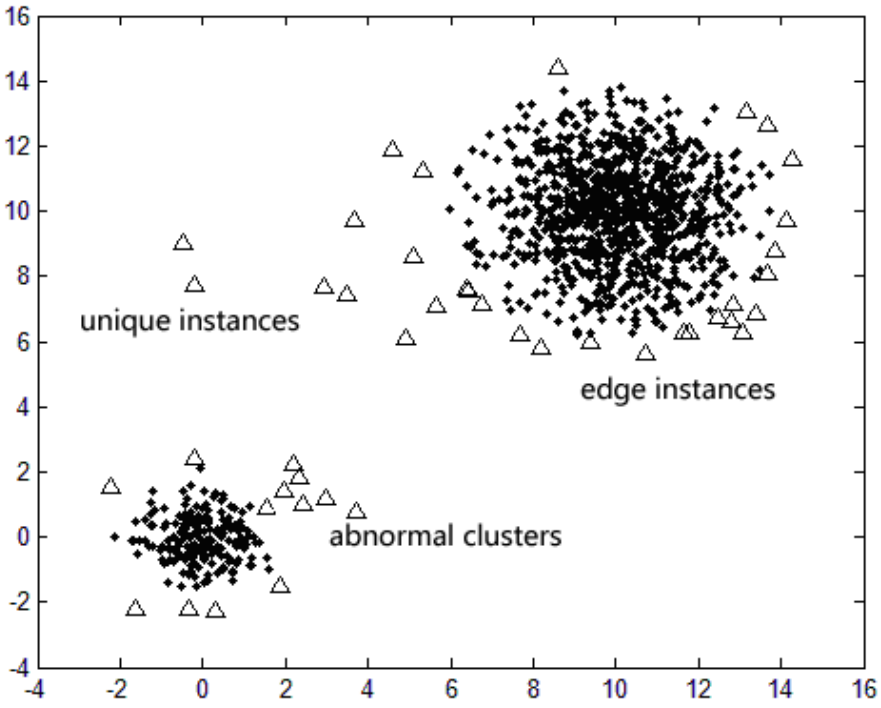

Figure 3.1: Three different anomalies.

the single points.

- *Edge Points* (sparse and close): They are around the clusters of normal instances, however, they are infrequent cases, and may refer to the boundary of the normal distributions. They have special meaning in some applications.

Among the various concepts and definitions of anomalies, there are two features that are commonly admitted: (i) *few*, *i.e.*, anomalies should be minority, only containing a very small number of instances, and (ii) *different*, *i.e.*, they are different from others. Instead of describing degree of anomaly as a binary attributes or single numerical measure, we propose a density-based hybrid measurement and employ it in the second stage. This measurement has a relatively high time complexity, yet generates accurate and meaningful results. Benefiting from the two-stage approach, only a small number of instances need to be processed by this time consuming measure which makes the overall computation time acceptable. The hybrid measure will be discussed in detail in Section 4.2.

Chapter 4

Anomaly Detection: A Filtering-and-Refinement Framework

One problem of the traditional anomaly detection methods is that most of their computational overhead is spent on building models or classifiers for the normal instances. In our approach, a filtering-and-refinement framework is utilized to improve the computational efficiency and effectiveness: In the first step, we scan the original dataset, filter out the normal data and generate a small portion of anomaly candidates, with linear time complexity. In the second step, we use a refinement algorithm to compute and check the detailed distance and other features of the candidate, and generate the final results. Since the candidate size is usually an order of magnitude smaller than the original data set, and can be loaded in the memory, the algorithm will cost much less time than directly computing and checking the details on the entire dataset. The overall framework of the algorithm is shown in Figure 4.1.

4.1 Deterministic Space Partition

The task of the first stage is to roughly separate original dataset into normal instances and anomaly candidates. Since one has to scan the entire dataset at this stage, the algorithm should be efficient in

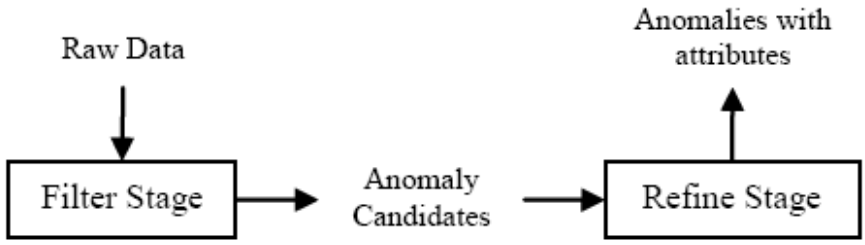

Figure 4.1: The Two-Stage Algorithm Framework: Filtering-and-Refinement

both time and space. We propose a novel *deterministic space partition* process to filter out obvious normal instances and generate anomaly candidates efficiently. By analyzing the random partition process, we believe that a balance of efficiency and effectiveness could be achieved by replacing the random factors with more deterministic measures, and this method should be a qualified choice for the filtering stage of our anomaly detection algorithm, considering the efficiency and robustness to find normal instances.

In order to process deterministic space partition (DSP), one need to first generate an optimal dimension or sub-dimension based on a deterministic test on dimension level, denoted as T_{dim}, then, based on an attribute-level deterministic test, denoted as T_{sp}, one can calculate a local optimal split point on this optimal sub-dimension. With T_{dim} and T_{sp}, the data can be split into two subsets. Repeat this process recursively until the partition reaches a predefined depth or the instances are isolated.

The DSP algorithm is outlined in Algorithm 1.

Algorithm 1 Deterministic Space Partition

INPUT: dataset D, current depth c, depth limit l
OUTPUT: partitions of the dataset D with related depths

if $c \geq l$ or $|D| \leq 1$ **then**
  return current_partition D;
else
  generate all sub_dimensions, denoted as d_list;
  $max\_td = 0$; $best\_dim = null$;
  for each sub_dimension dim in d_list **do**
    calculate $T_{dim}(d)$;
    if $T_{dim}(d) > max\_td$ **then**
      $best\_dim = dim$;
  $max\_sp = 0$; $best\_sp = null$;
  for each split point s in $best\_dim$ **do**
    calculate $T_{sp}(S)$
    if $T_{sp}(S) > max\_sp$ **then**
      $best\_sp = s$;
  $D_{left} = \text{getSubset}(D, d.best\_dim < best\_sp)$;
  $D_{right} = \text{getSubset}(D, d.best\_dim \geq best\_sp)$;
  $\text{DSP}(D_{left}, c+1, l)$;
  $\text{DSP}(D_{right}, c+1, l)$;

If we apply the DSP algorithm on a given dataset, we can get at most 2^l partitions with data

instances distributed in all partitions, and then we go through all the partitions and count the number of instances in each partition. Partitions with a relatively large number of instances tend to be clusters of normal instances, whereas partitions with a small number of instances tend to be anomalies.

After analyzing the time and space complexities of the DSP algorithm, it is clear that the algorithm only splits data on one sub-dimension at each iteration whereas most of the other sub-dimensions stay the same. Hence it is unnecessary to calculate T_{dim} measures for each dimension in all iterations. We can store T_{dim} and the intermediate partitions to speed up subsequent calculations.

Considering the recursive nature of DPS, we can employ a full binary tree structure to store and represent the partitions and intermediate results. By traversing the information stored in this binary tree, we could review the partition process and also reuse dimension measures. To differentiate the binary tree structure with partition information with regular tree structure, we define a *Filter Tree* structure to utilize the result of the filtering stage and keep track of the deterministic space partition process.

Definition 1 (Filter Tree): Let T be a node of a Filter Tree, and d be a tuple in the current subset. T is either an external-node with no child, or an internal-node with one test and exactly two children nodes (T_l, T_r). A test consists of a dimension q and also a split value p such that the test $d.q < p$ can split the current subset into T_l and T_r. Also, each node keeps the corresponding intermediate result of the deterministic space partition process.

Constructing Filter Tree in DSP process is intuitive and straightforward. In each partition, after initializing a filter tree node T, we need to first generate all sub-dimension candidates based on current status, and then select an optimal sub-dimension and keep it in current node as q. Then by analyzing data distribution on q, we can get an optimal split value, which is denoted as p. With this partition test, we can split the current subset into two parts by examining $d.q < p$ on each instance d in the subset. These two parts can be used to construct two child nodes of current filter tree node. Repeat this process recursively, a filter tree could be constructed based on the given dataset.

Example 1. An example of DSP and Filter Tree construction is shown in Figure 4.2, where a

2-dimensional dataset with 255 instances is presented. To split the original dataset as the first partition in DSP, we need to measure dimension d_1 and d_2 and choose an optimal dimension with more anomaly information. Assume d_1 is chosen as our split dimension based on T_{dim} measures of both dimensions, then we need to select an optimal split value on d_1 based on T_{sp}, which is (-1.6) in this example. After this, we could split the whole dataset into two parts and go on with similar partition recursively. Note that, in DSP, anomalies are more susceptible to isolation and hence have a short path-length. For example, a_0 could be isolated with two steps, so the path-length of a_0 in the filter tree is 2. However, more steps are needed to isolate those instances deep in the cluster.

4.2 Dimension Measure and Split Point Measure

Given a dataset with n dimensions, in order to partition the dataset in an optimal way which can isolate anomalies easily and also keep dense normal instances together, we need to measure each possible dimensions with some criterion based on the current partition status, and choose the dimension which contains more clustering and anomaly information. After that, a split value should be generated on this dimension in a fashion that tends to preserve the boundary of normal instances and anomalies. However, in large and high-dimensional datasets, measuring each dimension and all the possible split points in the original data could be time consuming. In addition, to the best of our knowledge, no measure or statistic test describing the clustering and anomaly distribution has been proposed before. So it is necessary to design an efficient and effective method to illustrate distribution in favor of detecting anomalies.

A split test on a specific dimension serves like a classifier with the target to separate normal instances and anomalies. In the filtering stage, partitions with different split tests would be processed multiple times recursively until a rough boundary of normal instances and anomalies could be worked out. To generate a relatively accurate boundary description with limited partitions in DSP, we need to measure the quality of each split test to make sure they separate datasets in a good way.

Another important factor of this problem is efficiency of measuring the quality of split tests. Intuitively, partitions near the boundaries of dense clusters can separate normal and abnormal effec-

tively, however, identifying dense clusters in large scaled and high-dimensional datasets is difficult and time consuming. To generate good partitions and calculate split tests efficiently, we propose a histogram-based partition method to roughly estimate the quality of a split test and return a relatively good partition in a very short time. By projecting instances into a number of bins on a specific dimension, we can build a histogram for each dimension in linear time, which can be reused in the whole process continuously. The histogram-based method helps DSP isolate anomalies and preserve normal instances, and also accelerate anomaly detection process with confident intermediate results.

Example 2. An example of qualified split value is shown in Figure 4.3 (a). At the initial step of DSP, assume we already choose dimension x as our optimal dimension for the first partition. From two candidates as presented in the figure, which should be chosen as optimal split value on dimension x? Intuitively, both split tests can be used in space partition process because S_1 (solid line) and S_2 (dash line) could both roughly separate the whole dataset against dimension x, and drive the partition process. However, if we measure each split test against isolation degree, and analyze them based on its contribution to the anomaly detection process, we will claim that S_1 is a more effective split. With S_1 as the split test, this partition tends to separate possible anomalies from normal instances and preserve obvious normal instances within a cluster. If we choose S_2 as split test, which is a partition on an obvious dense normal instance cluster, this will lead to a relatively low isolation degree if we want to separate normal and anomalies. In Figure 4.3 (b), a histogram with 50 bins which summarizes the dataset on dimension X is presented.

If we build histogram by discretizing the original distribution and projecting instances on the given dimension, we can find that isolation degree of S_1 and S_2 still can be measured precisely. And also, we could get a set of split value candidates as well: the boundary values of each bin in histogram. Each of the split value could partition the whole dataset into two parts. But which one is the best? The goal of space partition is to roughly describe the boundaries between normal and anomalies gradually. So we hope each partition could treat normal and anomalies as different classes, and keep instances in each class as tight and homogeneous as possible, and also try to maximize inter-class variance.

Motivated by Otsu's method [15], we employ inter-class variance as split point test, denoted

as T_{sp}. First, define the within-class variance as the weighted sum of variances of each cluster as follows,

$$\sigma_{within} = w_1\sigma_1^2 + w_2\sigma_2^2$$

where w_n the relative frequency of class n, which is defined as,

$$w_n = \sum_{i \in class_n} P(i), P(i) = \frac{n_i}{N}$$

and σ_n is the variance within class n. And then we can calculate inter-class variance as follows,

$$T_{sp} = \sigma_{between} = \sigma^2 - \sigma_{within} = w_1 w_2 (\mu_1 - \mu_2)^2$$

where σ is the variance of the histogram, and μ_n is the mean of class n.

After calculating this measure for each split candidate, we can generate an optimal value by selecting the one with maximal T_{sp} which is the inter-class variance. This optimal value could be used to split normal and anomalies, as well as different normal clusters. It could help partition the dataset in a more meaningful and anomaly sensitive fashion.

Another problem in the DSP algorithm is, at each iteration, based on partition status, a meaningful sub-dimension should be chosen based on some criteria and then a split value could be calculated based the measure presented above. But what are the criteria to describe the quality of a sub-dimension?

In real-life high dimensional datasets, different dimensions usually have different domain meanings. What is more, some dimensions may contain more information and are more related to specific applications, like classification or clustering, whereas others might be irrelevant. More specifically, in anomaly detection, dimensions or sub-dimensions with data distribution information that have potential to separate normal and anomalies should be chosen and analyzed in our process.

To illustrate the quality of sub-dimensions, we propose a histogram-based data distribution measure. We denote the dimension test used in this approach as T_{Dim}, which could be described as follows,

$$T_{Dim} = \frac{span(\text{dim})/spam(original\_\text{dim})}{\sum_{i=1}^{n} -P(i)\log_2 P(i)}.$$

This dimension quality measure is combined with two parts. $\sum_{i=1}^{n} -P(i)\log_2 P(i)$ is the entropy of this sub-dimension, which describes the randomness and informative degree of the data. And the enumerator part is an adjustment which leads to relatively larger dimension. A sub-dimension with a higher value of T_{Dim} tends to contain more information about data distribution. After calculating T_{Dim} for each of the candidate dimension, we can easily choose an optimal partition dimension by selecting the sub-dimension with the largest value of T_{Dim} measure.

An example of dimensions with different T_{Dim} values can be found in Figures 4.3 (b) and (c). Based on the histogram, which dimension we should choose for the next step of DSP? Intuitively, we believe dimension in Figure 4.3 (b), denoted as D_1 is a better choice, because on this sub-dimension, more information about normal clusters and anomalies could be obtained, and we can easily separate dense normal clusters and anomalies on this sub-dimension. Data in Figure 4.3 (c), denoted as D_2, is somehow evenly distributed, based on the histogram, it is difficult to tell which bins contain normal instances and which bins might be anomalies. If we calculate T_{Dim} on these two dimensions separately, D_1 would get a higher value and it could generate a better separation that leads to a more accurate result of anomaly detection.

Both T_{sp} and T_{Dim} are calculated based on a fixed length histogram, which reduces the time complexity to constant even if we evaluate the whole dataset on multiple sub-dimensions. The overall time complexity at the first stage is linear with only one scan of all data instances. In our experiments, a histogram with 50 bins can roughly represent the clustering and anomaly distribution, leading to a reasonable partition process. Also, fine-grade histograms could lead to better partition results.

In isolation forest [10], considering dense normal clusters may shade anomalies from being separated, the authors chose to use samples in the size of 256 in each iTree construction process. They also indicated that, small sized sample of original dataset could roughly preserve the boundary of normal and anomalies. Based on this observation, a number of iTrees are constructed on many small sized samples, and then calculate anomalies by traversing each iTree. This method works well in isolation forest with simple datasets, but intuitively, for complex datasets with local anomalies, and clusters with different distributions and densities, like Figure 3.1, sub-sampling could blur the boundary of normal and anomalies, which would reduce the anomaly detection accuracy, especially

for those local anomalies.

In DSP, we adopt different methods based on different sizes of datasets, and try to use the whole dataset by different means. If the size of the dataset doesn't exceed a predefined size limit Ψ, we apply DSP on the original dataset directly. However, if the size of dataset is tremendous, we would first employ random sampling without replacement to partition the dataset into large samples with the size of Ψ, then apply DSP on all samples to get final results. Based on our experiments, we define Ψ as 5000, which leads to reasonable results. Note that although we try to optimize each partition in DSP, it is still impossible to distinct all anomalies from normal instances correctly at the filter stage. We need to store the controversial instances as the anomaly candidates and go on with a more strict and time-consuming criterion to the refinement stage.

As a summary of all the steps of the filter stage, Algorithm 2 describes the details of the filtering stage.

Algorithm 2 Filter Process

```
INPUT: dataset D;
OUTPUT: anomaly candidate C;

sample_list = random_partition(D, Ψ);
depth limit l = ceiling(log2(Ψ/8));
for each sample in sample_list do
   dsp_list = DSP(sample, 0, l);
   for each d instance in D do
      getIsolationDegree(dsp_list, d);
getAverageDegree(d);
C = getAnomalyCandidate(D);
```

To improve efficiency, we limit the height of filter tree to l, which makes most of the instances cannot be isolated in such a short height. The path length can be estimated based on the height of actual node and number of instances inside the node, as an adjustment, which is getIsolationDegree function in the algorithm. For example, after traversing the filter tree, if d instances are found in node n, the path length of x can be estimated as follows,

$$PathLength(d) = Height(n) + c(sizeof(n))$$

where

$$c(sizeof(n)) = \frac{2H(sizeof(n)-1) - 2(sizeof(n)-1)}{2(sizeof(n)-1)sizeof(n)}$$

$H(i)$ is an adjustment and can be estimated as $ln(i) + e$, where e is Euler's constant. If the path length of an instance is larger than $1.75l$, we believe that with a high probability, these instances are deep in a cluster.

As a test, we apply filter process on a dataset containing clusters with different distributions and densities, as well as local anomalies, with the size of 3040 and 123 anomalies (Figure 3.1). After anomaly candidate generation, we eliminate 2764 normal instances, and get only 276 anomaly candidates with 0.11 seconds (Computer configuration could be found in Section 5). In all 2764 normal instances detected in this method, no false negative mistake is found, which means the precision of this anomaly candidate generation method on this dataset is 100%.

4.3 Candidates Refinement

After generating anomaly candidates in the first stage, we need to refine them with more stable and accurate criteria to handle complicated clustering and local anomaly situations. Most normal instances (usually more than 70% based on our experiments) are eliminated in the first stage. With a small amount of anomaly candidates, robust and accurate methods can be used with relatively high time complexity in the refinement stage. Based on the requirements and data properties, we propose two density-based tests for each anomaly candidate to illustrate its anomaly degree so that one can refine the candidates and get final results. In traditional density-based approaches, one needs to go through all the instances calculating *k-nearest neighbors* (KNN), which is inefficient. After eliminating most of normal instances, the calculation time would be reduced significantly.

Unlike other methods, instead of treating anomaly detection a binary problem, we assign to each candidate two numerical attributes of being an anomaly to make the anomaly definition compatible with complex applications. These attributes are designed to measure both local and global anomaly degrees. To calculate these attributes for an instance d, we need to calculate the k-distance and k-distance neighbors first.

Definition 2 (k-distance): For any positive integer k, the k-distance of object p, denoted as

$k\text{-}dist(p)$, is defined as the distance $d(p, o)$ between p and an object $o \in D$ such that: for at least k objects $o' \in D(p)$, it holds that $d(p, o') \leq d(p, o)$, and for at most $k - 1$ objects, $o \in D(p)$, it holds that $d(p, o') < d(p, o)$.

Definition 3 (k-distance neighbors):

$$knn(o) = \{p \in D | d(p, o) \leq k\_dist(o)\};$$

Based on the above definitions, we can calculate local and global attributes for all anomaly candidates generated from stage one.

Definition 4 (local and global anomaly attribute):

$$T_l = \frac{1}{k} \sum_{p \in knn(o)} \frac{\mu_{kd(o)}}{\mu_{kd(p)}}; T_g = \frac{size(c)\mu_{kd(o)}}{\sum\limits_{p \in C} \mu_{kd(p)}}$$

where

$$\mu_{kd(o)} = \frac{1}{k} \sum_{p \in knn(o)} d(p, o),$$

which represents the relative distance from o to its neighbors.

T_l measures the isolation degree of one instance. If the data point is unique in the dataset, the value of T_l would be high. T_g measures the distance of anomaly to normal instance clusters. So high value of T_g could be used to detect small size yet relatively dense cluster in the dataset.

In the refinement stage, we go through the output of DSP, and calculate T_l and T_g for every anomaly candidate. After calculation of all anomaly candidates, anomalies can be found by setting application-specific thresholds for T_l and T_g separately, denoted as Δ_l and Δ_g. By examining whether the measure is larger than predefined threshold, all the candidates can be put into the following four classes:

- *Unique Instances*: Candidates with high T_l and T_g are sparse instances far from normal clusters.
- *Abnormal Clusters*: Candidates with low T_l and high T_g could form relatively dense clusters yet far from dense normal clusters.

- *Edge Points*: Candidates with high T_g and low T_g are sparse instances which are close to normal clusters.
- *Normal Instances*: Candidates with low T_l and low T_g are normal instances but mislabeled in DSP.

In this way, we can split anomaly candidates into different categories and find different anomalies based on their properties.

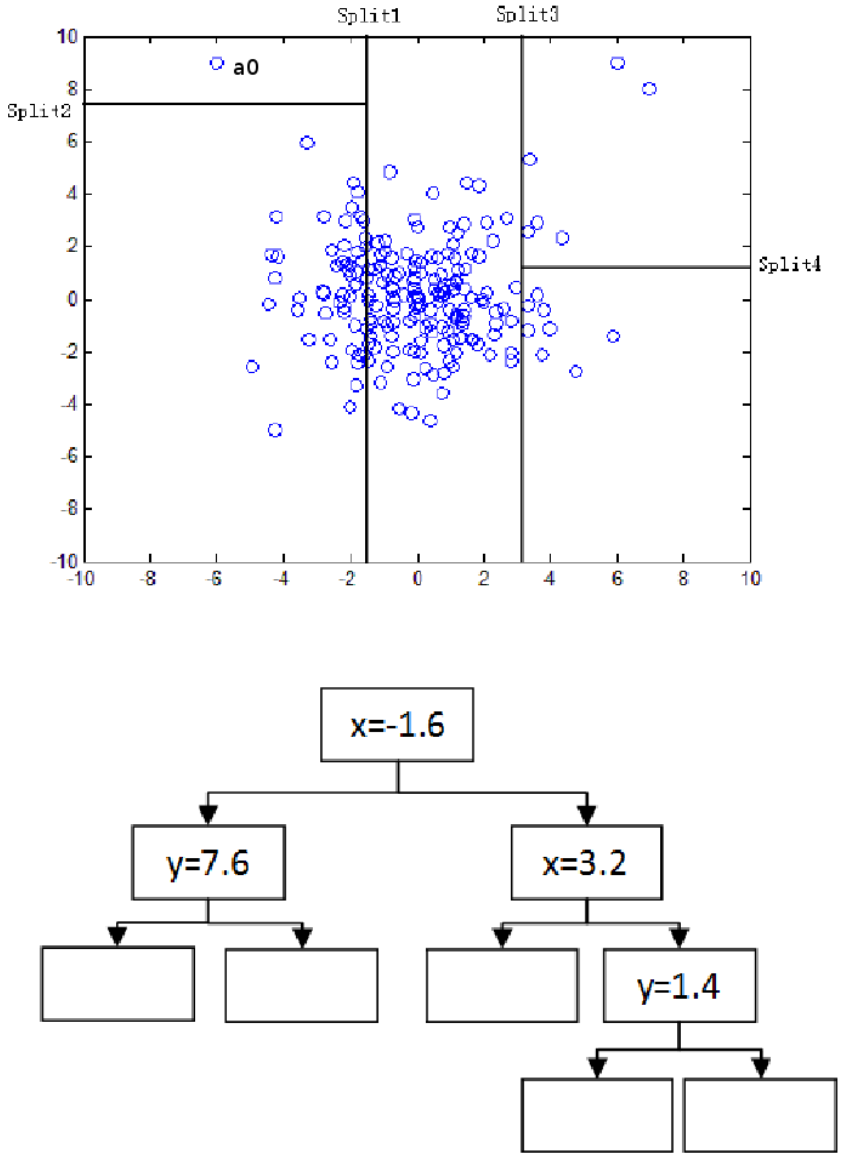

Figure 4.2: Deterministic Space Partition and Filter Tree.

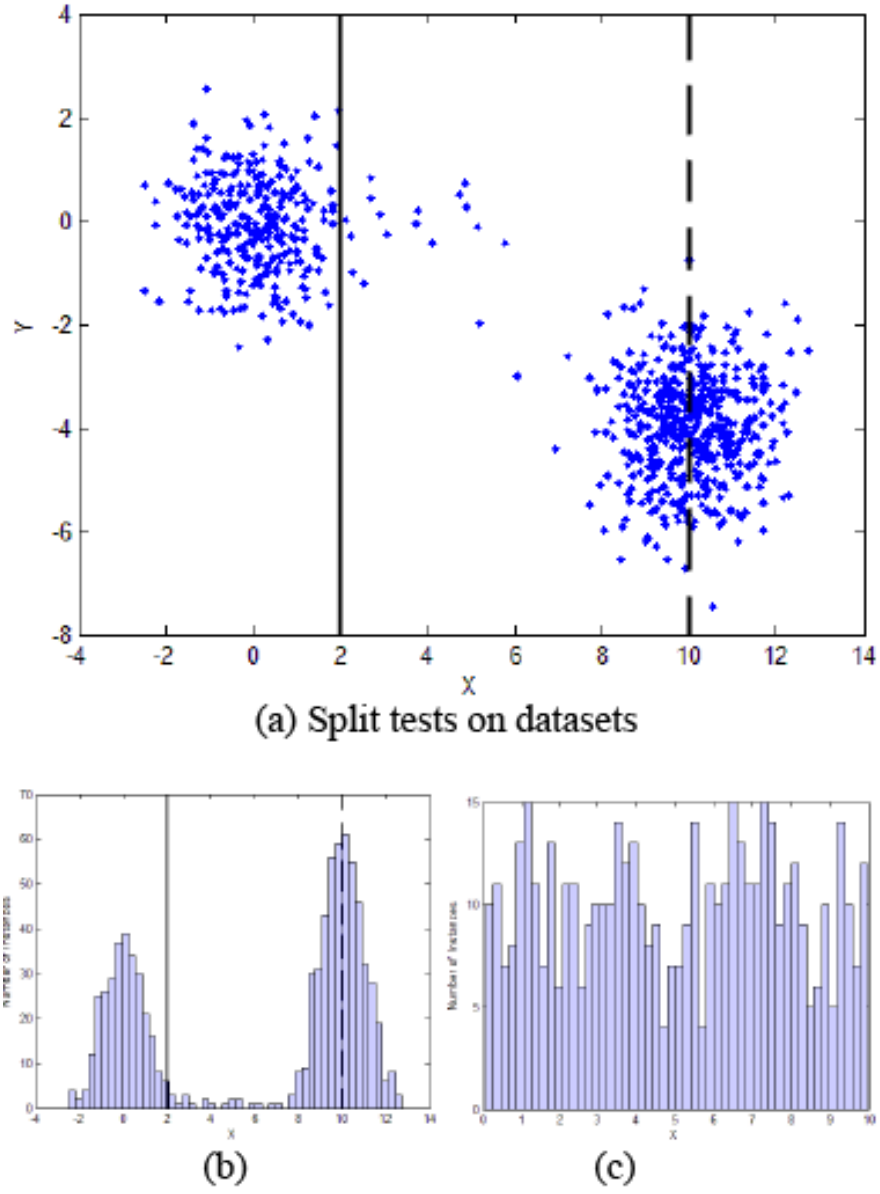

Figure 4.3: Different split tests on datasets and the corresponding histogram, the solid line is good split test, denoted as S_1, and the dash line is a bad split test, denoted as S_2

Chapter 5

Performance Study

This section presents the experiments results of a series of synthetic datasets to evaluate the proposed anomaly detection method. In these experiments, our method is compared with the isolation forest method and a density-based nested loop method. For all experiments, CPU time, the number of mistakes, including false positives and false negatives, as well as the number of true anomalies that can be detected are reported. These experiments are conducted as single threaded jobs processed at 2.26GHz CPU frequency with 2GB memory.

Because different methods have different criteria and parameters to separate anomalies with normal instances, we try to set all parameters as reasonable and equally effective as possible. For the isolation forest algorithm, we use the recommended parameter setting, which is the size of samples = 256, and also set the number of samples to 100. For density-based method, we use a commonly used setting of $k = 6$ in our experiment. For our method, as mentioned above, we set dataset size threshold Ψ to 5000 in filtering stage, and $k = 6$ in the refinement stage.

The first part of the experiments is based on a set of 2-D synthetic datasets (SDS) specially designed to test the anomaly detection ability of each method on complex datasets. In these datasets, we try to cover different scenarios with different data distributions as much as possible. Different densities, clustering forms, distributions, as well as local anomalies are simulated to test these methods. Table 5.1 provides the properties of all datasets and information on anomaly classes. Figure 5.1 gives an intuitive glance of part of the datasets.

The aim of this experiment is to compare our method with the isolation forest and density-based nested loop methods in terms of processing time and precision. Figures 5.2 and 5.3 report the accuracy of all methods on the 2-D datasets. Figure 5.4 reports the processing time for all methods.

From Figures 5.2 and 5.3, we can notice that, for detection accuracy, DSP and the nested-loop

Name	Total Size	Anomalies	Normal Clusters
SDS 1	4060	60	3
SDS 2	5075	75	3
SDS 3	6090	90	5
SDS 4	7105	105	4
SDS 5	8120	120	6
SDS 6	10150	150	4

Table 5.1: Synthetic Dataset Properties

algorithm give very similar results: they are stable and effective in complicated datasets, and could detect different anomalies with high accuracy and confidence. However, the accuracy of isolation forest is 10 times worse than our algorithm, and it makes a number of mistakes in datasets with local anomalies and multiple clusters. Considering processing time, one could notice that, isolation forest is the most efficient algorithm, whereas the density- based method is approximately 1000 times slower. Our method is at least 100 times faster than density-based algorithm, and very close to isolation forest in terms of efficiency. What's more, the results of our method and the density-based nested loop method are very robust, whereas that of isolation tree is instable and changes with variances in multiple executions.

Another important issue in anomaly detection is high dimensional data. In high dimension space, density and distance measures lose their meaning because data instances in high dimension space are sparsely distributed and also nearly equally spaced.

Considering the dimension selection ability of DSP, our method has potential for accurate anomaly detection tasks in high dimensional data. To test this ability, we designed a dataset with 50 dimensions, and apply three methods mentioned above to this high dimension scenario. In all 50 dimensions only 6 sub-dimensions contains meaningful clustering abd anomaly information whereas the rest of dimension are generated randomly. This dataset contains 3000 instances, and 124 out of them are anomalies.

Our method could detect 99 anomalies correctly within 6042 ms, while isolation tree could only detect 49 anomalies. In this scenario, density-based method does not give any significant result within a reasonable time.

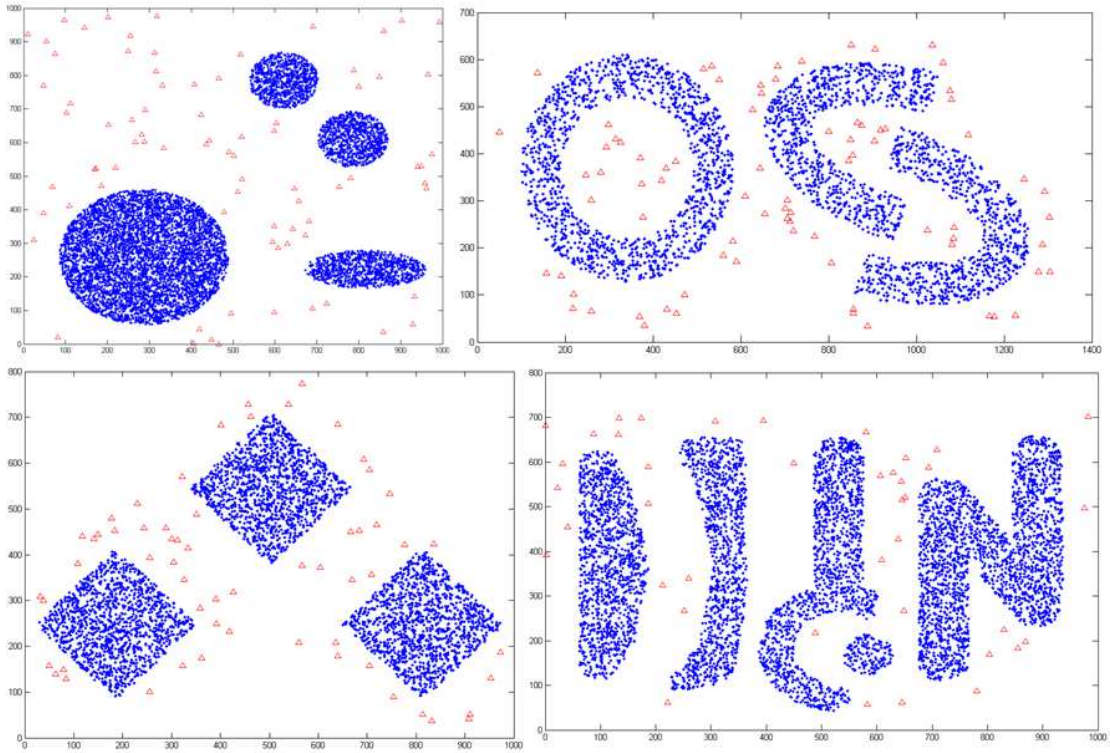

Figure 5.1: 2-D synthetic datasets, blue points are normal instances whereas red rectangles are anomalies.

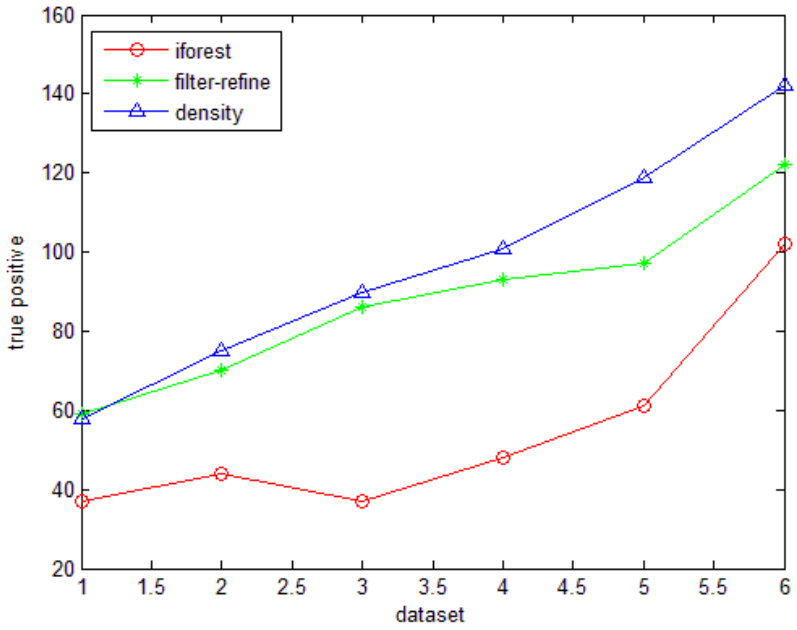

Figure 5.2: Number of anomalies detected by these three methods(true positive).

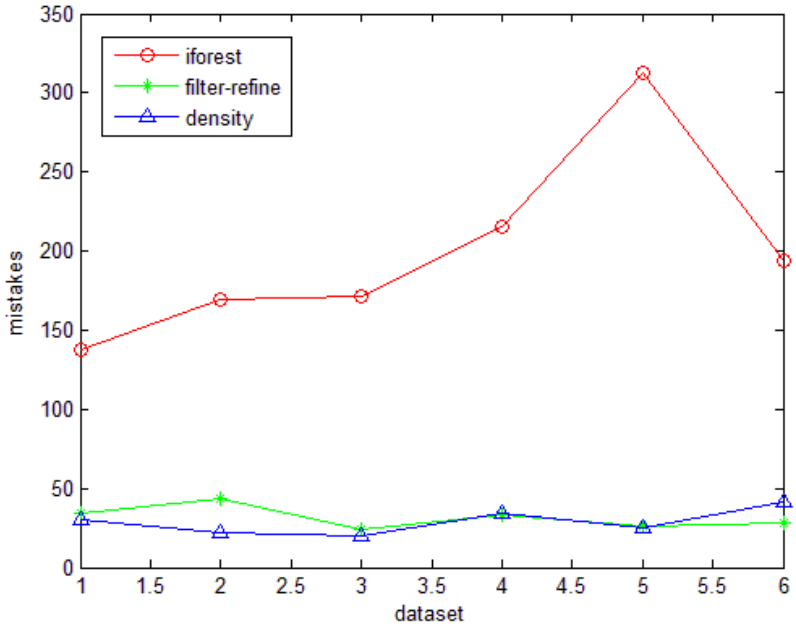

Figure 5.3: Errors of these three methods (include false positive and false negative).

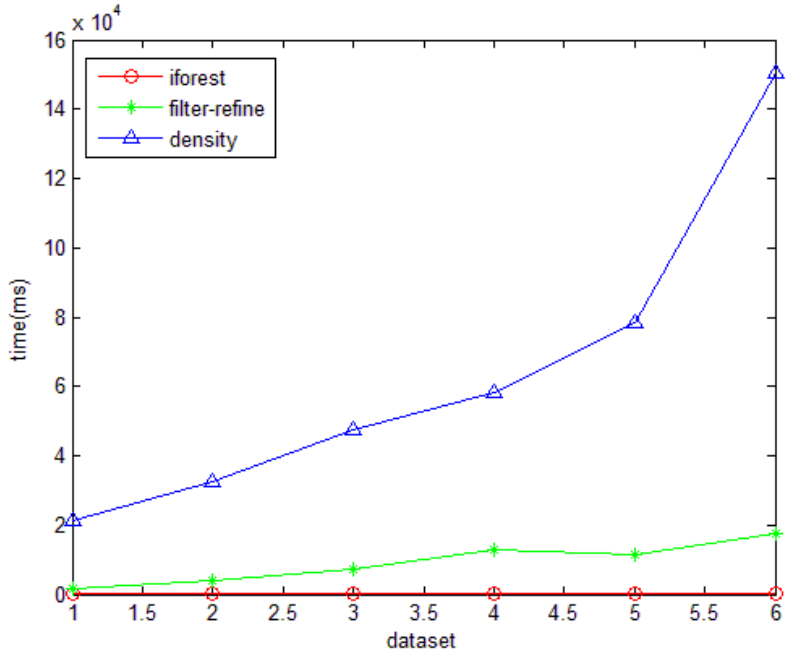

Figure 5.4: CPU time of all anomaly detection methods.

Chapter 6

Conclusions

In this paper we propose an anomaly detection method called filtering and refinement. Observing that anomaly notions in different domains have different characteristics,we divide the anomaly detection process into two stages to achieve high efficiency, accuracy and robustness. In the filtering stage, all possible anomaly candidates are generated efficiently; in the refinement stage, each candidate is examined with strict anomaly definition to finally determine whether it is an anomaly. This two-stage approach takes advantages of different methods and generates better detection results with lower time complexity.

Moreover, at the refinement stage, by employing two new anomaly measures, anomalies can be accurately detected and also be separated into different categories, conveying appropriate meanings in real applications.

Our experiment results show that the proposed approach performs much better than random spatial partition algorithm like isolation forest in datasets with complicated cases, and almost as good as density-based methods but with a much lower processing cost. This two-stage approach strikes a good balance of accuracy and computational complexity, and makes it a good and practical choice in anomaly detection applications.

References

[1] V. Barnett and T. Lewis. *Outliers in statistical data*. John Wiley, 1994.

[2] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical Science*, 17:235–249, 2002.

[3] M. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *SIGMOD'00*, pp. 93–104.

[4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys*, 41, 2009.

[5] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anomaly detection: Detecting intrusions in unlabeled data. In *Proc. 2002 Int. Conf. of Data Mining for Security Applications*, 2002.

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases. In *KDD'96*, pp. 226–231.

[7] V. J. Hodge and J. Austin. *A Survey of Outlier Detection Methdologies*. Kluwer Academic Publishers, 2004.

[8] E. Knorr and R. Ng. Finding intensional knowledge of distance-based outliers. In *VLDB'99*, pp. 211–222.

[9] X. Li, Z. Li, J. Han, and J.-G. Lee. Temporal outlier detection in vehicle traffic data. In *ICDE'09*.

[10] F.T. Liu, K.M. Ting, and Z. Zhou. Isolation forest. In *ICDM'08*.

[11] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *VLDB'94*.

[12] M. Prastawa, E. Bullittc, S. Hoa, and G. Geriga. A brain tumor segmentation framework based on outlier detection. *Medical Image Analysis*, 8, 2004.

[13] T. Shi and S. Horvath. Unsupervised learning with random forest predictors. In *J. Computational and Graphical Statistics*, 2006.

[14] O. Tylor and D. Addison. Novelty detection using neural network technology. In *Proc. 2000 Int. Conf. on COMADEN*, 2000.

[15] N. Otsu. A Threshold Selection Method from Gray-Level Histograms. In *IEEE Systems, Man and Cybernetics*, 1979.