DETECTION OF OUTLIERS IN SPATIAL-TEMPORAL DATA

by

James P. Rogers

A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

Dr. Daniel Barbara, Dissertation Co-Director

Dr. Carlotta Domeniconi, Dissertation Co-Director

Dr. Alex Brodsky, Committee Member

Dr. Edward Wegman, Committee Member

Dr. Daniel Menascé, Senior Associate Dean

Dr. Lloyd J. Griffiths, Dean, The Volgenau School of Information Technology and Engineering

Date: December 6, 2010

Fall Semester 2010
George Mason University
Fairfax, VA

Detection of Outliers in Spatial-temporal Data

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

James P. Rogers
Master of Science
George Washington University, 1991
Bachelor of Science
North Carolina State University, 1983

Director: Daniel Barbara, Professor
Department of (Computer Science)

Fall Semester 2010
George Mason University
Fairfax, VA

Dedication

I dedicate this dissertation to my family for their support and patience.

I dedicate this dissertation to my parents for providing me the opportunity to receive an education and believing in the importance of education and striving to improve. I hope this achievement makes you feel proud.

I dedicate this dissertation to my children for their encouragement and understanding. I hope you will be able to achieve your dreams.

I dedicate this dissertation to my wife for her love and being by my side during this journey. You showed me the inner strength to battle difficult problems.

Acknowledgments

I would like to thank the following people who made this possible:

Dr. Guido Cervone for providing the buoy dataset for my research.

Dr. James Shine for editing my manuscript and words of support.

Mr. Doug Caldwell for creating the weather animations to illustrate my results.

Mr. John Neander for weather subject matter expertise and creating the track of Hurricane Katrina and the cold front that were used to illustrate my results.

Table of Contents

List of Tables

List of Figures

Abstract

DETECTION OF OUTLIERS IN SPATIAL-TEMPORAL DATA

James P. Rogers, PhD

George Mason University, 2010

Dissertation Director: Daniel Barbara

Outlier detection is an important data mining task that is focused on the discovery of objects that deviate significantly when compared with a set of observations that are considered typical. Outlier detection can reveal objects that behave anomalously with respect to other observations, and these objects may highlight current or future problems.

Previous outlier detection methods have focused primarily on only one non-spatial numerical attribute and have not successfully dealt with multiple dimensions. Many previous methods assume a Gaussian distribution of the data which is probably a major fallacy in determining outliers for spatial-temporal data. Most previous efforts did not provide a statistical confidence measure, but including a confidence measure should improve the detection of outliers. Outlier detection is often complicated by noise in the data, so a good outlier detection methodology should be successful in identifying outliers in noisy data. Global outlier methods calculate a single outlier statistic that summarizes the outliers for the entire geographic area and temporal duration, while local outlier methods calculate a outlier statistic for each feature based on its similarity to its neighbors. Previous methods have not been able to determine outliers as the vector of attributes, location, and time change. The objective of my research is to devise a methodology to address these problems and challenges.

The objective of my research is to develop a robust method of diagnosing outliers and to extend it to detecting outliers in spatial-temporal data. A spatial-temporal outlier is an observation whose values are significantly different from those of other spatially and temporally referenced objects in its spatial-temporal neighborhood.

Geographic phenomena are difficult to analyze using traditional data mining methods. Determining relationships among phenomena as they move and change over time is not possible by means of human analysis of spatial-temporal data streams. Also, the volume of spatial-temporal data being collected is increasing steadily due to the usage of cameras, sensors, and mobile devices (e.g., cell phones) and is too much data for the human to analyze.

My method, unlike many detection methods found in the literature, does not require the user to enter the number of outliers to be found or the percentage of outliers to be found and does not assume any distribution of the data (e.g., Gaussian). My method only requires the input of two parameters: the statistical confidence level and the number of nearest neighbors, and only the statistical confidence level is significant. My method allows for different ways to measure the degree of non-conformity and works for high-dimensional data, noisy data, and data with or without clustering information.

The basic outlier detection method was extended to spatial-temporal data by using kernels for the vector of attributes, spatial, and time that provides a capability to focus outlier detection on local neighborhoods, and the user is able to input weights for each of the kernels. Local spatial-temporal outliers are outliers determined within a specific spatial area and time frame which is a subset of the entire spatial area and total temporal duration.

Empirical evaluation was conducted on several datasets with very good results achieved. The datasets increased in complexity and dimensionality. The experiments on these datasets using my method produced results with a high True Positive percentage and a low False Positive percentage.

Chapter 1: Introduction

1.1 Outlier Detection

Outlier detection is an important data mining task that is focused on the discovery of objects that are exceptional when compared with a set of observations that are considered typical. These objects are important since they often lead to the discovery of exceptional events. Outlier detection can reveal objects that behave anomalously with respect to other observations, and these objects may highlight current or future problems. In other cases, the sudden appearance of a large number of outliers can point to a change in the underlying process that is generating the data. Hawkins (26) defines an outlier as "an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism." These observations are inconsistent with the majority of the data. Outliers deviate too much from the normal observations. This suggests the possibility of detecting outliers by knowing the mechanism by which the typical observations were generated and testing points for membership to this mechanism. That is the path that early work in outlier detection followed in the statistical community as can be seen in (34). First, postulate a model for the probability distribution of typical objects (e.g., a Gaussian model), and then compute the likelihood of an object being generated by the postulated model. Unfortunately, finding the right model is as difficult as the original problem of finding outliers. This approach does not always work well in practice, because it can be seen as inducing a model over the typical data and using it to test points, but the goal of outlier detection methodology is to not make assumptions about the data and the data distribution. A robust outlier detection methodology should not assume a Gaussian distribution of the data or any other distribution.

Outliers are defined by (38) as the identification of new or unknown data or signal that

a machine learning system is not aware of during training. The principles of outlier detection methods according to (38) are the following: (1) the outlier detection method must be capable of robust performance on test data that maximizes the exclusion of novel samples while minimizing the exclusion of known samples; (2) all training and test data must be normalized and be within the same range; (3) the goal of the outlier detection method is to minimize the number of parameters that need to be set; (4) the outlier detection method should be independent of the number of features and should have reasonable performance when the number of samples is small or noise is present; and (5) the computational complexity of the outlier detection algorithm should be as low as possible.

The objective of this research is to develop a robust method of diagnosing outliers and to extend it to spatial-temporal data. In spatial-temporal data, observations are tagged with their geographical location and their timestamp. These observations are a vector of attribute values for the features being measured. A spatial-temporal outlier is an observation whose values are significantly different from those of other spatially and temporally referenced objects in its spatio-temporal neighborhood. It represents an object that is significantly different from its neighborhood even though it may not be significantly different from the entire population. Results of experiments show that the technique utilized for the research described in this dissertation is capable of effectively detecting local outliers, which overcomes one of the main problems of previously developed techniques for outliers detection based on global parameters. Local spatial-temporal outliers are outliers determined within a specific spatial area and time frame which is a subset of the entire spatial area and total temporal duration. The discovery of outliers in spatial-temporal data is complicated by needing to focus the search on an appropriate local spatial-temporal neighborhood of objects.

1.2 Problems/Challenges

Geographic phenomena, which provide spatial-temporal data, are dynamic, active, and change over time in direction, speed, growth, location, and other variables. Many geographic-related phenomena could be changing simultaneously in space and time, and often it is difficult to predict these changes. These phenomena are difficult to analyze using traditional data mining methods. Determining relationships among phenomena as they move/change over time is not possible by means of human analysis of spatial-temporal data streams. Also, the volume of spatial-temporal data being collected is increasing steadily due to the usage of cameras, sensors, mobile devices (e.g., cell phones) and satellites and is too much data for the human to analyze. A crucial step in detecting spatial-temporal changes in the large volumes of spatial-temporal data is the formulation of an automated technique to detect spatial-temporal outliers.

Previous outlier detection methods have focused primarily on only one non-spatial numerical attribute and have not successfully dealt with multiple dimensions, so there is a need to better identify outliers in high-dimensional data. Many previous methods assume a Gaussian distribution of the data which is probably a major fallacy in determining outliers for spatial-temporal data. It is strongly desirable to not presume any distribution of the data. Most previous efforts did not provide a statistical confidence measure, but including a confidence measure should improve the detection of outliers. Outlier detection is often complicated by noise in the data, so a good outlier detection methodology should be successful in identifying outliers in noisy data.

Global outlier methods calculate a single outlier statistic that summarizes the outliers for the entire geographic area and temporal duration, while local outlier methods calculate a outlier statistic for each feature based on its similarity to its neighbors. Most previous outlier detection methods have focused on global outliers instead of local outliers, and previous methods have not been able to determine outliers as the vector of attributes, location, and time change.

The objective of my research is to devise a methodology to address these problems and

challenges.

1.3 Applications

Outlier detection can uncover important anomalies in fields such as intrusion detection, fraud analysis, telecommunications, signal processing, pattern recognition, image analysis, medical analysis, population movement, and human behavior. Applications of outlier detection abound in fields such as credit fraud, criminal investigation, spatio-temporal analysis, and computer intrusions. Outlier analysis may identify fraudulent credit card usage by finding large purchase amounts for a given credit card holder in comparison to normal charges made by the same person. Other outliers possibly detected from credit card usage can be the location of the purchase, the type of purchase, or the purchase frequency. Outliers can be useful in marketing by identifying the buying behavior of poor or wealthy consumers. Outliers can be useful in the field of medicine by signaling an alert when patient data deviates significantly from the normal, healthy range or unusual responses to treatments. This research will examine spatial-temporal data to find meaningful patterns and statistically determine the data that does not belong to any patterns. Potential spatial-temporal outlier applications are biological, chemical, or radiation detection and propagation; pollution detection (air, water, ground); movement of groups and individuals; identification and alerting of "strange" or different human behaviors; and weather disaster forecasting (floods, tornados, earthquakes, hurricanes).

For instance, consider the evacuation of a city population under an emergency, such as chemical contamination of the air. It would be important to know when the chemical contamination level reaches a critical level through outlier detection, and then monitor the movement of the masses, as well as the movement of the contaminants, dynamically altering the escape routes if a prediction is made showing that people will come in contact with the contaminants while evacuating the city. Thus, the results of the proposed research could reduce the possibility of catastrophe from biological, chemical or radiation contamination; disease propagation; and natural disasters.

The results of this research could provide a capability to adapt and adjust to emerging situations more quickly and provide an overall improvement in situational understanding and information superiority. The results of this research could possibly be utilized to mine data from multiple data sets to create information and knowledge; draw relationships among the data; provide meaning to the data; and convert data into actionable information. This research could utilize information from multiple data sets to identify spatial-temporal outliers on people, objects, and the environment; provide knowledge discovery; enhance the cognitive senses to perceive and identify outliers, and would rapidly process information to identify data that does not fit a pattern. This spatial-temporal outlier research could improve the understanding of the environment including terrain, weather, hazards, populations, and their interaction.

1.4 Contributions

- The method, unlike many detection methods found in the literature, does not require the user to enter the number of outliers to be found and does not assume any distribution of the data (e.g., Gaussian).
- The method only requires two parameters: the statistical confidence level and the number of nearest neighbors.
- The method allows for different ways to measure the degree of strangeness.
- The method works for high-dimensional data, noisy data, and data with or without clustering information.
- The basic outlier detection method was extended to spatial-temporal data that uses kernels for vector of attributes, spatial, and time to provide capability to focus outlier detection on local neighborhoods, and the user is able to input weights for each of the kernels.

- A non-parametric likelihood-based method was created, using Kernel Density Estimation, to improve computational performance, because nearest-neighbor computations are costly.

Chapter 2: Related Work

2.1 Outlier Detection

A large body of work has been published in the area of discovering outliers with respect to clustering models (17; 42; 48; 50; 23). However, most of these algorithms do not focus on the discovery of outliers, but rather offer ways to deal with them, and in all the cases, the discovery of outliers is done by careful setting of the threshold parameters that are difficult to determine.

Although there is some overlap between various types of methods, outlier detection can be classified into the following categories: (1) parametric, which assumes a distribution or probability model for the data; (2) non-parametric, which does not make any assumptions on the data but is usually more computationally expensive; (3) distance-based, where objects that are a substantial distance from any other cluster are considered outliers; (4) deviation-based, which examines differences in the main characteristics of objects in a group; (5) density-based; and (6) visualization-based. A very good survey of anomaly detection methods can be found in (12).

2.1.1 Parametric Methods

Parametric methods assume knowledge of the underlying data distribution, and previous outlier work was usually limited to univariate data since fitting the Gaussian model becomes difficult for higher dimensional spaces due to the computations required resulting from the increase in the number of parameters for multivariate data. Two or more Gaussian distributions may not imply a multivariate Gaussian distribution if they are not independent. This approach can be seen as inducing a model over the Gaussian data and using it to test points.

The outlier detection methods that assume a distribution or probabilistic model for the given data, then identify the outliers with respect to the model using a test. The test verifies whether an object varies significantly in relation to the assumed distribution. A Gaussian distribution is the distribution that is usually assumed. Using this method requires knowledge of the assumed data distribution, the mean, variance, and the expected number of outliers. An object may be an outlier under one assumed distribution and not an outlier under a different distribution. The outlier methods that assume a distribution are most suited for one-dimensional data and not for multi-dimensional data. If the distribution cannot be modeled with an existing standard distribution, it is difficult to identify outliers.

Parametric statistical outlier detection methods include Gaussian Mixture Modelling (40; 47) and Hidden Markov Models (13). Gaussian Mixture Modelling models general distributions estimating the density using fewer kernels than the number of patterns in the training data. The parameters of the model are determined by maximizing the log likelihood of the training data with respect to the model. If the number of dimensions of the data is high, a large number of samples are needed to train the model. Hidden Markov Models contain a finite number of unobservable/hidden states. State transitions are governed by a stochastic process to form Markov chains. At each state, some state-dependent events can be observed. The emission probabilities of these observable events are determined by a probability distribution with one for each state. To estimate the parameters of a Hidden Markov Model, sequences of normal events collected from normal system operation are used as training data. An expectation-maximization (EM) algorithm is used to estimate the parameters. Once a Hidden Markov Model has been trained, probability measures can be thresholded with the test data for outlier detection.

The Local Moran's (Moran's I) (2) is a parametric method that employs a statistical test that is widely employed in geographical information systems to test outliers. However, the test assumes a Gaussian distribution of the data and is employed for data with only one non-spatial attribute. Local Moran's I, fully implemented in ArcGIS 9, compares the value of each point that needs to be tested with the values of all the points in the study

area, or neighborhood of points adjacent to the test point. The local Moran's I statistic is expressed as

$$I_i \;=\; z_i \sum_j w_{ij} z_j \tag{2.1}$$

where i is the point to be tested, z_i is the deviation from the mean of the attribute for the test point i, z_j is the deviations from the mean of the attribute for each of the points in the neighborhood j, and w_{ij} are weights that indicate whether j is spatially close to i. $w_{ij} = 0$ if i and j are not close, and $w_{ij} = 1$ if i and j are close. The value of I_i is compared to the distribution of values for the rest of the points I_j by calculating a Z-score. First, the expected I_i, assuming a Gaussian distribution of values, is calculated. This is subtracted from the computed I_i and the difference divided by the square root of the variance of I_i, to obtain a Z-score. A large, positive value for the Z-score indicates that the neighbors of i share similar values for the attribute. A large negative Z-score indicates the point is surrounded by dissimilar values, and therefore it is an outlier with respect to them. The local Moran's I statistic can be used, as defined, for data that contains only one attribute beside the spatial coordinates. While it is possible to extend this to two attributes, the approach has not become very popular or successful. Moreover, the test assumes Gaussian distributions for the neighborhood of values surrounding the point. This is in contrast to the method proposed here that is designed for data with more than one non-spatial attribute and does not assume any distribution for the values. The Moran's I statistic assumes no correlation between points, in spite of the fact that spatial data sets are notoriously correlated. Randomization principles could be applied to adjust the estimates as planned for this method, but this has not been done and no provisions to that effect are implemented in ArcGIS 9.

2.1.2 Non-Parametric Methods

Non-parametric outlier detection methods include nearest neighbor (45; 20) and Parzen density estimation (14). For outlier detection, the distribution of typical vectors are placed

in clusters by the k-nearest neighbor distances. Outliers are then assessed by measuring the normalized distance of a test points from the cluster centers. The problem with this technique can be that for large datasets, a large number of computations are necessary. This technique does not require a smoothing parameter like the Parzen window technique. The Parzen window technique is based on estimating the density of the training data and comparing the test point with the density estimates of the training data.

2.1.3 Distance-based Methods

A distance-based outlier methodology, proposed in (32), shows that outlier detection can be done efficiently for large datasets and for k-dimensional datasets with large values of k ($k >= 5$). These cell-based algorithms have a worst case complexity of $O(kN^2)$ where N is the number of objects. The user must input the p and D parameters where p is a fraction and D is an absolute distance. There is no universally correct function for determining p or D, so it is done by refining the parameters through multiple trials, by permitting the user to change the values of p and D for each new trial during the search for outliers. The fraction p is the minimum fraction of objects that must be outside the D-neighborhood of an outlier, where 0.995 is the value most often used as the starting value for p. Once there are at least (M+1) neighbors in the D-neighborhood, the search is stopped and O (the object) is declared a non-outlier. If there are less than (M+1) neighbors in the D-neighborhood, O is declared an outlier. M is the maximum number of objects within the D-neighborhood of an outlier where $M=N(1\text{-}p)$. The basic theory is that an outlier in multidimensional space must be significantly far away from the rest of the data points along at least one dimension.

A cell-based approach utilizing the method in (32) begins with a 2-D simplified version of the algorithm. Each of the objects is quantized into a 2-D space that has been partitioned into cells or squares. A typical non-boundary cell has eight Layer 1 neighbors, where Layer 1 neighbors are the immediate neighboring cells. Any pair of objects within the same cell are at most a distance D/2 apart. Two objects that are Layer 1 neighbors are at most a distance D apart. Layer 2 neighbors are those additional cells within 3 cells of the object.

If an object is neither a Layer 1 or Layer 2 neighbor, then those two objects must be at least a distance D apart. If there are greater than M objects in the cell, none of the objects in the cell are outliers. If there are greater than M objects in the cell and Layer 1, none of the objects in the cell are outliers. If there are less than or equal to M objects in the cell and Layer 1 and Layer 2, every object in the cell is an outlier. When moving from k =2 dimensions to k >2 dimensions, the algorithm requires only one change (cell length) to incorporate a general k-dimensional cell structure. This work requires the user to input a distance and a fraction of objects that must be outside of the distance.

In comparison to the two distance-based methods described above, the method my research technique focuses on does not require the user to input a distance or a fraction of objects that are outliers. My method does not assume that a certain percentage are outliers or that there are any outliers in the data, and my method does not require the user to change parameters prior to each trial during the search for outliers.

An exception to the univariate restriction is the method of Rousseeuw et al (29). This method attempts to find a robust fit model, i.e., one that is similar to the fit that would have been found without the outliers, and use it to diagnose those points that do not fit the model well. The choice of this method is to compute the Minimum Covariance Determinant (MCD) estimator, which finds a subset of the data whose covariance matrix has the lowest possible determinant. Their algorithm, FAST-MCD, uses randomization to speed up the high-complexity calculation. Outliers are detected by computing the Mahalanobis distance, using the covariance matrix computed by MCD, and finding points that are too far away from the robust centroid. Looking for points with Mahalanobis distance that are too large, calls for a threshold, i.e., it is equivalent to consider as outliers those points that are three or more standard deviations away from the centroid which is the common practice of univariate statistical techniques. In (25), which uses a technique based on MCD to find multiple clusters and diagnose outliers, the authors argue that not every data set will give rise to an obvious separation of outliers and non-outliers by using robust estimators.

Angiulli (1) proposed an outlier-discovery algorithm that uses the sum of the distances to

the k-nearest neighbors as a measure of isolation. This sum is called weight by the authors. Their algorithm aims to discover the top n outliers by sorting points by their weight and declaring the n points with largest weight as outliers. It is obviously difficult to set the value of n in advance for a given value, since it is highly likely that some of the points are not outliers. The method my research focuses on avoids this by providing a statistical test that will determine if a value of the weight is indeed strange enough to declare the point an outlier. Their method, however, describes a very useful technique that uses space-filling curves to avoid the costly computation of the k-nearest neighbors of each data point. The technique is based on transforming the d-dimensional points into one-dimensional points lying on a Hilbert or z-curve (46). The transformation guarantees that two points that are close in the curve are also close in the d space. However, two points in the d space are not necessarily close in the Hilbert curve, so it is necessary to produce multiple transformations by shifting the curves. The idea of shifted Hilbert curves, described in (36) guarantees that a search for a nearest neighbor of a query point conducted over several lists of linearized points (each list corresponding to a shifted Hilbert curve) finds the true nearest neighbor with a high accuracy. Such technique can easily be employed in my method to speed the computation.

Another technique for spatial outlier detection is described in (33) where an appropriate function is designed that can effectively represent the outlierness of an object. The outlierness can be viewed as the difference between a single non-spatial attribute and its neighbors. The two approaches described in (33) are a weighted z-value approach and an averaged difference approach. Both of these approaches use spatial weights. The weight is determined by spatial relationships such as distance and common border length. The user inputs the number of requested outliers which the authors recommend be less than five percent. The steps in the weighted z-value approach are: 1. identify k nearest neighbors using Euclidean distance; 2. compute the weighted average for each object of the non-spatial attribute values; 3. determine the weight based on the spatial relationships of distance and border length between an object and its neighbors; 4. calculate the weighted average by summarizing the

product of the weight and the non-spatial attribute value for each neighbor; 5. compute the difference between object's non-spatial attribute value and the weighted average; 6. compute the standardized difference, based on the mean and standard deviation, as the outlierness factor; and 7. the top objects with largest outlierness factors are identified as outliers. If the impact of spatial properties is ignored in this weighted z-value approach, it becomes the statistical z-value approach. The averaged difference approach is based on the weighted average of the absolute difference between an object and each of its neighbors. For this averaged difference approach, an object is compared with each of its neighbors one by one, instead of obtaining the average of all its neighbors before comparison. This work examines only one non-spatial attribute and requires the number of requested outliers to be input. In contrast, my research technique is capable of incorporating multiple non-spatial attributes. Also, this work requires an input on the percent of requested outliers, whereas my research technique does not require an input on the number or percent of outliers and does not assume that there are outliers.

Some distance-based methods for determing outliers are index-based, nested-loop and cell-based (24). The index-based method uses multi-dimensional indexing structures, such as R-trees or $k-d$ trees, to search for neighbors of each object within a radius around that object. When the number of objects within the neighborhood of an object exceeds the maximum, the object is not an outlier. This algorithm has a worst case complexity of $O(kN^2)$, where k is the dimensionality and N in the number of objects. The complexity takes only the search time into account even though building an index can be computationally intensive. The nested-loop algorithm has the same computational complexity as the index-based algorithm, but avoids index structure construction and tries to minimize the number of inputs and outputs. The memory buffer space is divided into two halves and the data into several logical blocks. A good choice of the order in which the blocks are loaded can achieve input and output efficiency. A cell-based algorithm was developed to avoid the $O(N^2)$ computational complexity for memory-resident data sets. Its complexity

is $O(c^k + N)$, where c is a constant depending on the number of cells and k is the dimensionality. In this method, the data space is partitioned into cells, and each cell has two layers surrounding it. The algorithm determined outliers by cell instead of by object.

2.1.4 Deviation-based Methods

Deviation-based methods do not use statistical tests or distance measures to identify outliers. Outliers are identified by examining the main characteristics of objects in a group. Objects that differ from this description are considered outliers. Two methods for deviation-based outlier detection are the sequential exception technique that sequentially compares objects in a set, and a second approach that uses an on-line analytical processing (OLAP) data cube technique. The sequential exception technique builds a sequence of subsets of objects and dissimilarities are assessed between subsets in the sequence. The OLAP approach uses data cubes to identify outliers in large multi-dimensional data. A cell value in the cube is determined to be an outlier if it is significantly different from the expected value, based on a statistical model. The user can drill down on cells that are flagged as outliers. The model considers variations and patterns in the measure value across all of the dimensions to which a cell belongs.

2.1.5 Density-based Methods

An object is a density-based outlier (10; 30) if a fraction of the objects is at a distance greater than a specified distance. For this method, the fraction of objects and distance are required inputs. Finding good estimates for the fraction of objects and distance can require much trial and error.

LOF (10) works by calculating an outlier score, called a Local Outlier Factor, for each point. It works by first considering the local density of the point which is computed as the inverse of the average distance from its nearest neighbors. The local density for each of the neighbors is computed, and the values are averaged. The ratio between the density of the point and the average density of its neighbors is the average relative density of the

point, and is a measure of whether the point is in a denser or sparser region of the data. This measure is the outlier score of the point, or LOF. Sorting the points according to this measure gives a ranking of them as outliers. An incremental version of LOF that attempts to diagnose outliers in data streams is (43). An extension of LOF that considers cluster structures is (27). A resolution-based outlier algorithm has been proposed (18). Similar to LOF, it computes an outlying score for each data point. The computation of the outlier factors depends on the data clustering which is based on a definition of close neighborhood.

Differing from an outlier score, an innovation of my method is the use of a statistically-based test to compute the strangeness of a point using transduction. Using my method, this test identifies outliers, if any, by setting the confidence level. This contrasts with methods like LOF which only produce a ranking of points according to their outlying factor. In methods like LOF, the only way to decide which points are outliers is to set up a threshold for that outlying factor or to output the highest ranking points according to their outlying factor. The setting of that threshold is difficult and artificial, since there is no guidance on how to set that threshold. In situations where there are no outliers, the existing methods would still find some outliers since they would output the highest outlying points. In contrast, my method is not forced to diagnose any outliers. In a streaming environment where new points are constantly occurring and all points have not been seen, setting an outlying factor threshold is more difficult since there is no idea on how high the new points will score. In contrast, my method will continue diagnosing new points with the statistical test that provides a confidence level.

2.1.6 Distance-based and Density-based Methods

The method of Knorr (31), which is used as a test comparison to my research method, uses distance and density calculations to discover the outliers in a data set. The method aims to answer a nearest neighbor query with radius D for each point in the data set and decide if there are enough neighbors to the point in this D-neighborhood. This decision is controlled by a parameter p, which thresholds the minimum fraction of records that must

be found outside of the D-neighborhood for the point to be an outlier. This method needs two parameters to be adjusted (D and p). However, the code of DB overwrites a badly chosen D by computing a reasonable one using sampling. Therefore, in experiments with this method, the choice of p was varied.

2.1.7 Visualization-based Methods

A technique for spatial outlier detection was proposed by Shekhar (49). The method uses the difference between the attribute value at location x and the average attribute value of x's neighbors to determine if x is an outlier. It works only for univariate data (one attribute besides the spatial coordinates) and assumes a Gaussian distribution for the non-spatial attribute value. Their methodology for finding outliers uses a directed graph, neighborhoods, and comparing the attribute value to the average attribute value of its neighbors. The authors examine a traffic data set from Minneapolis and St. Paul that has 990 nodes/sensors for a 20 square mile section. A neighborhood is defined as the set of traffic sensors adjacent to a given sensor. The neighborhood relationship is based on the directed edges in the underlying spatial graph. Sensors on opposite sides of the highway are not neighbors if the traffic is going in opposite directions. The number of neighbors is user-defined. The non-spatial attribute that is utilized for detecting outliers is traffic volume. Outliers are the locations of traffic stations whose traffic volume measurements are inconsistent with those of their neighbors. The outliers are displayed visually with time on the x-axis and station locations on the y-axis. This work examines only one non-spatial attribute and states that outlier detection in multi-dimensional space using multiple attributes is beyond the scope of this paper, whereas my research technique is capable of incorporating multiple non-spatial attributes.

2.2 Spatial-temporal Data Mining

Previous work on spatial-temporal data mining was focused on two types of patterns: frequent movements of objects over time and evolution of natural phenomena such as forest

coverage. In general, there has been limited work on spatial-temporal data mining, which has been treated as a generalization of pattern mining in time series data, and there has been much less research on determining outliers in spatial-temporal data.

A technique for mining, indexing, and querying historical spatial-temporal data is described in Mamoulis (37). This technique examines spatial-temporal data, and attempts to identify objects that move in periodic patterns. Patterns occur when objects follow the same routes (approximately) over regular time intervals. Objects that follow approximate periodic patterns include transportation vehicles (buses, planes, trains), animal movements, mobile phone users, and people going to work. Given a long sequence and a period, the aim is to discover the most representative trend that repeats itself in every so many timestamps. The patterns are, possibly non-contiguous, sequences of object locations that reappear in the movement history periodically. Since an object is not expected to visit exactly the same locations at every instant of each period, the patterns are not rigid but differ slightly from one occurrence to the next. The pattern occurrences may also be shifted in time (e.g., due to traffic delays or over-sleeping). Their model assumes that the locations of objects are sampled over a long history. The movement of an object is tracked as an n-length sequence of spatial locations. One location is recorded for each timestamp in the history. If the difference between consecutive timestamps is fixed (locations are sampled at regular time intervals), the movement can be represented by a single sequence of locations. Each location is expressed in terms of spatial coordinates. The goal is to discover movement patterns that repeat themselves every T timestamps. It is highly unlikely that an object will repeat an identical sequence of (x,y) locations precisely. Even if the spatial route is precise, the location transmissions at each time stamp are unlikely to be perfectly synchronized. If the first daily locations of the object are very close to each other, they will be treated differently by a standard mining algorithm. To handle the noise in object movement, the exact locations of the objects are replaced by regions (e.g., districts, cells, grid) which contain them. A problem with this approach may occur if the space division is too large or small. The first step in mining the periodic patterns is to obtain frequent 1-patterns. The sequence

of locations is divided into T spatial datasets with one for each offset of the period T. A dense cluster in a dataset corresponds to a frequent pattern. In order to identify the dense clusters for each dataset, a density-based clustering algorithm such as DBSCAN is applied. Clusters with less than a minimun support number of points are discarded, since according to the definition, they are not frequent 1-patterns. From the discovered 1-patterns (clusters for each dataset), a variant of the Apriori algorithm is applied to discover longer patterns. After the patterns of length k have been discovered, the patterns at the next level are found until there are no more patterns at the current level or there are no more levels. For fast and effective candidate generation, minimum bounding rectangles of the pattern regions are used. During candidate pruning, every (k-1) subpattern of the candidate pattern is checked. If there is at least one pattern which agrees with the subpattern and the minimum bounding rectangle-intersection with it is non-empty at all those locations, the pattern is accepted as a candidate pattern. Otherwise the candidate pattern cannot be a valid pattern since some of its subpatterns are not included. The validate-pattern function takes as input a k-length candidate pattern and computes a number of actual k-length patterns from it. Although this algorithm's bottom up approach can find all partial periodic patterns correctly, it can be very slow due to the huge number of region combinations to be joined. If the actual patterns are long, all their subpatterns have to be computed and validated. In addition, a potentially huge number of candidates need to be checked and evaluated, so a top-down method for discovering long patterns is proposed. Instead of finding frequent patterns in a bottom-up approach, the tree is traversed in a top-down, breadth-first order. This spatial-temporal data mining work attempts to find periodic patterns in spatial-temporal data, but does not attempt to find the outliers. This work also appears to only use the spatial location of an object at a recorded time in its computations, but does not examine or use the vector of attributes describing the object at a spatial location at a recorded time.

An approach to detect spatial-temporal outliers is described in Birant (8). The steps in this approach are clustering, checking spatial neighbors, and checking temporal neighbors. The clustering is performed by the density-based DBSCAN clustering algorithm. DBSCAN

is modified to support temporal aspects and to find outliers when clusters have different densities. The two input parameters for DBSCAN are Eps and $MinPts$. Eps is a radius value, and $MinPts$ specifies the minimum number of points that should occur within the Eps radius. This modified algorithm requires an Eps distance parameter for the spatial component and another Eps distance parameter for the temporal component.

Chapter 3: Background

3.1 Transduction Confidence Machines (TCM)

Recently, the field of statistical learning theory (52) has developed an alternative to *induction*. This alternative is known as *transduction*. Instead of induction which uses all the available points to induce a model, transduction can use the data, or usually a small subset of it, to estimate unknown properties of points to be tested (e.g., membership to a class). This idea leads to algorithms that use standard statistical tests to compute the confidence on the estimation. Using transduction, researchers have built Transductive Confidence Machines (TCM) (19) which are able to estimate the unknown class of a point and attach confidence to the estimate. The transductive reliability estimation process has its theoretical foundations in the algorithmic theory of randomness developed by Kolmogorov (35). Unlike traditional methods in machine learning, transduction can offer measures of reliability to individual examples, and uses very broad assumptions. Transduction only assumes that the data points are independent and generated by the same stochastic mechanism. These properties make transduction an ideal mechanism to detect outliers, even though it has never been used before for that purpose. In this research, the ideas of TCM are used to design a test that determines if a point is an outlier and attach a confidence to the estimation. New points are compared to a baseline of "normal"observations. The baseline points are a random sample that may have been organized into a clustering model.

The method this research focuses on uses the concept of *transduction* to find outliers with respect to a series of pre-defined clusters. According to the transduction paradigm (52), unknown estimates for individual points of interest can be made directly from the training data, as opposed to using *induction* to infer a general rule for the points. The estimates that are of interest in learning are the likelihood that a point belongs to a given

cluster in the current clustering model. In that sense, training examples are the points already clustered. Transduction has been previously used to provide confidence measures for the decision of labelling a point as belonging to a set of pre-defined classes (44; 28; 19). TCM (19) introduced the computation of the confidence using Algorithmic Randomness Theory (35). The first proposed application of Algorithmic Randomness Theory to machine learning problems, however, corresponds to Vovk et al. (53). The confidence measure used in TCM is based upon universal tests for randomness, or their approximation. A Martin-Lof randomness deficiency test (35) based on such tests is a universal version of the standard p-value notion, commonly used in statistics. Martin-Lof proved that there exists a universal test for randomness smaller than any other test up to a multiplicative constant. Unfortunately, universal tests are not computable, and have to be approximated using non-universal tests called p-values. In the literature of significance testing, the p-value is defined as the probability of observing a point in the sample space that can be considered more extreme than a sample of data. This p-value serves as a measure of how well the data supports or does not support a null hypothesis. The smaller the p-value, the greater the evidence against the null hypothesis.

Users of transduction as a test of confidence have approximated a universal test for randomness, which in its general form is non-computable, by using a p-value function called strangeness measure (19) or non-conformity score (53). There is more than a single definition of strangeness measure, and in general, its definition depends on the base model used to construct the TCM. The general idea is that the strangeness measure corresponds to the uncertainty of the point being measured with respect to all the other labelled examples of a class. A higher strangeness measure means a higher uncertainty. This research shows how to adapt the definition of the strangeness function of TCMs to the context of finding estimates for points to be outliers. In doing so, points in a given cluster of the model are treated as belonging to the same label or class, even though such class has not been defined. This is possible since points in the same cluster exhibit enough similarity, as measured by the underlying clustering algorithm, among themselves to be commonly labelled as belonging

to a class.

3.2 Strangeness Measure for TCM

In (44), strangeness is measured as the ratio between the sum of the distances from the point u to its k-nearest neighbors inside the class under consideration, y, and the sum of the k-nearest neighbors outside the class. Alternative definitions of strangeness are possible to accomplish the task of detecting outliers.

This definition of strangeness is based on the nearest neighbors to the point in and out of the particular cluster used to test the point against, which is borrowed from the transduction confidence machine that uses k-nearest neighbors(TCM-kNN). Alternative definitions of strangeness are possible, however, and for purposes of the methods my research is focused on, this definition is modified and the modification is described later in the Research Methodolgy section. For TCM, the Euclidean distance is utilized to compute the distance between pairs of points as done in the distance based method (31) which is used as a comparison, and TCM strangeness is the ratio of the sum of the k smallest distances from the same class proposed to put u in, to the sum of the k smallest distances from other clusters where k is the number of nearest neighbors, and measures how strange the point in question is with respect to the cluster proposed to put it in. The larger the value of the strangeness means the stranger the point.

3.3 p-value

The p-value is calculated in (44) as the fraction of points in the class that have strangeness greater than or equal to that of the point. In general, a p-value is the maximum probability under the null hypothesis of the test statistic assuming a value equal to the observed outcome or a value just as extreme or more extreme, with respect to the direction indicated by alternative hypothesis, than the observed outcome. A smaller p-value means a smaller chance that the test statistic could have assumed a value as incompatible with the null

hypothesis if the null hypothesis (class y is a good fit for point u) is true. The algorithm TCM-kNN attempts to place a new point in each class of the problem. While doing that, it may force the updating of some of the strangeness values for the training examples. This happens whenever the distance between the training example and the new point is less than the largest of the k distances that are used to compute the strangeness. It then computes one p-value for each of the attempts (i.e., for each class placement), and then predicts that the point belongs to the class with the largest p-value with a confidence equal to one minus the the second p-value.

Chapter 4: Research Methodology

Since the objective of my research was to identify spatial-temporal outliers, this research was conducted incrementally starting with finding outliers in low dimensional numerical data and concluding with finding outliers in high dimensional spatial-temporal data. The first phase was the design of a technique for identifying outliers in numerical data with a vector of attributes but no spatial or temporal data. This technique utilized transduction to estimate membership in a class and attach confidence to the estimate. For my research method, a strangeness measure is necessary to determine outliers, and the strangeness measure utilized was the Euclidean distance from the object being tested to its nearest neighbors. The final step of the first phase was to calculate the p-value of each data point, and those that have the smallest p-value are declared outliers. The only inputs required are the number of nearest neighbors and the confidence level. My research method does not require an assumption about the distribution of the data and does not require that a certain percentage of the data actually be declared as outliers. It is possible with my research method that none of the data will be declared as outliers. My research method is the foundation of an algorithm called Strangeness based Outlier Detection (StrOUD).

The second phase in my research was to add a capability to detect outliers in spatial data that has a geographic location (x and y coordinates) in addition to the vector of attributes. To accomplish this objective, a spatial kernel was added. A weighted uniform distribution was used for the spatial kernel, and then a weighted exponential decay function was tried as the spatial kernel.

The third phase in my research was to add a capability to detect outliers in spatial-temporal data that has a time stamp when the data was recorded in addition to a geographic location and a vector of attributes. To accomplish this objective, a temporal kernel was added to the outlier algorithm, that resulted in the outlier algorithm having three kernels

with a kernel for the vector of attributes, spatial component, and temporal component. A weighted uniform distribution was used for the temporal kernel, and temporal distances are computed only to a previous time.

The computational processing time to run the spatial-temporal outlier algorithm on spatial-temporal data was often many hours, so kernel density estimation (KDE) was utilized to determine if KDE could produce similar results to the spatial-temporal outlier algorithm but at a much reduced computational time. For my kernel density estimate methodolgy, a kernel density estimate is computed for each data dimension, and a KDE outlier algorithm was developed that utilized the density values from the KDE function to compute the probability for each point, then p-values are computed, and then outliers are determined.

4.1 Strangeness based Outlier Detection (StrOUD)

The ideas of TCM-KNN are adapted by my research (3) for the purposes of determining if a point is an outlier by computing the strangeness of any point with respect to a cluster y, instead of a class y, by considering points in y as the training examples of a class y. However, there is a fundamental difference between my outlier methodology and that solved by TCM. In TCM, the point being examined always belongs to one of the classes. In my outlier methodology, the objective is to determine if the point in question is an outlier, and hence does not belong to any of the clusters or to the entire population if no clustering information is available. If the point u being tested is an outlier, according to the definition provided in (44), the strangeness of u becomes the ratio between two large numbers, since the distances from the point in question, u, to any of the points within clusters are large. In some cases, this ratio will be small enough to be comparable to the strangeness values for points already in the cluster which will lead to false negatives. Experiments show this to be true. Therefore, a modified definition of strangeness is proposed as follows: the strangeness $\alpha_{y,u}$ of a point u with respect to a cluster y is defined as shown in Equation 4.1.

$$\alpha_{y,u} \;=\; \sum_{j=1}^{K} d(z_{yj}, u) \tag{4.1}$$

where z_{yj} represents the j^{th} closest point to u among the points in cluster y and $d(.,.)$ is a distance measure. This definition makes the strangeness value of a point far away from the cluster considerably larger than the strangeness of points already inside the cluster. This definition has been employed by Angiulli (1) as a measure of isolation. My method uses the Euclidean distance to compute the distance between pairs of points. This definition is very different from the one utilized by TCM-KNN, because my method does not divide the sum of the nearest neighbors within the cluster to that of neighbors outside the cluster.

Using Equation 4.1,a series of p-values can be computed for u, with one for each cluster $y = 1, \cdots, c$ where c is the number of clusters. Each p-value is computed as the fraction of points in the cluster, including u, that have strangeness greater than or equal to that of the point u being diagnosed.

$$t_{y,u} \;=\; \frac{|\{z_{yi} : \alpha_{y,z_{yi}} \geq \alpha_{y,u}, i = 1, \ldots, |y|\}| + 1}{|y| + 1} \tag{4.2}$$

where the various $z_{yi} : i = 1, \ldots, |y|$ are all the points in cluster y, $\alpha_{y,z_{yi}}$ is the strangeness of point z_{yi} computed with respect to cluster y, and $|\cdot|$ represents the cardinality of the set given as argument. The additive constant accounts for the point u itself.

Each cluster provides a p-value, so a series of c p-values is obtained, and the largest p-value in this series is called p_{max}. This provides a way of testing the fitness of point u, by testing the null hypothesis H_0^y as "u is fit to be in cluster y." Thus, the alternative hypothesis H_1^y is "u is ill-fit to be in cluster y." Selecting a confidence level $1 - \tau$ which is usually 95%, tests are conducted to determine if $p_{max} \;\leq\; \tau$. If true, all the null hypotheses are rejected and the point is declared an outlier. Otherwise, all the alternative hypotheses are rejected. By choosing τ, the percentage of outliers is not being chosen. Instead, τ

Given a point u under consideration:
Compute the p-values of u with respect to clusters $1, \cdots, c$.
Sort the p-value list in descending order.
Call p_{max} the highest p-value, and p_{next} the next in the list.
If $p_{max} \leq \tau$ then
Reject all the null hypotheses H_0^y, for $y = 1, \cdots, c$
Declare u an outlier with confidence $1 - \tau$
Else
Reject all the alternative hypotheses (u belongs to a cluster in the model).

Figure 4.1: StrOUD: strangeness-based outlier detection algorithm

selects the confidence level for the hypothesis testing. A τ of 5% does not necessarily result in 5% of the points being identified as outliers as the experiments will corroborate, but τ controls when the null hypothesis will be rejected and indicates the maximum error that will be incurred by that decision.

The pseudo-code of the Strangeness based OUtlier Detection (StrOUD) algorithm, that computes the fitness of a point u to be included in existing clusters, is shown in Figure 4.1. It contains a rule to accept or reject alternative hypotheses for the fitness of u to be included in existing clusters. The algorithm is not used to decide the placement of the point in the clusters. That decision is left to the clustering algorithm of choice. If no clustering information is available, the algorithm will test a single null hypothesis H_0 as u is fit to be considered part of the general population of points.

This test is a novel aspect of my research methodology, since it permits the determination of which points are outliers, if any, by setting the confidence level. This is in contrast to methods like (1; 10) that only produce a ranking of points according to their outlying factor (strangeness). In those methods, the only way to decide which points are outliers is to set up a threshold for that outlying factor or to output the highest k ranking points according to their outlying factor. The setting of that threshold is a difficult proposition and an artificial one, since there is no guidance on how to set that threshold. In situations where there are no outliers, those methods would still identify the highest k outlying points as outliers.

In my method, if there are no outliers at the confidence level which is input, StrOUD is not forced to diagnose any point as an outlier. Another important difference is in a data streaming environment where new points are constantly occurring. Since all the points have not been seen, it is impossible to determine the k highest ranking points. Also, setting an outlying factor threshold becomes even more difficult, since there is no idea on how high newer points will score. In contrast, my methodology will continue diagnosing new points with the principled, statistical test that gives a confidence level.

The operation of changing the strangeness values for some of the examples in the clusters whenever the new point is closer to the example than at least one of the example's k nearest neighbors is potentially a costly one. Instead of just keeping the strangeness values for all the points in the clusters, it requires maintaining the information of the k-nearest neighbors for each point, so when a new point is under consideration, these lists can be examined and the distances to the k-nearest neighbors compared to the distance to the new point. However, since the only interest is in diagnosing whether the new point is an outlier or not, all this information can be removed and that operation dropped, instead working with the original strangeness values of the clustered points. In doing so, the only risk is working with some value of strangeness that is larger than it really should be. To understand this, consider a point u to be tested. Suppose u is close to at least one point i in cluster y, so that there exists a point j in y, among the k-nearest neighbors of i, that satisfies $d(i,j) > d(i,u)$. In this case, u will replace j among the nearest neighbors of i when tested in cluster y, making the $\alpha_{y,i}$ smaller than the value previously calculated without the inclusion of u in the cluster. This replacement would make i less strange than previously calculated. As a consequence, if $\alpha_{y,i}$ is not modified, the algorithm will consider i more strange than it really is, thus increasing the p-value for u. There is a risk in lowering the chance of declaring the new point u an outlier. However, this only happens if the new point is close enough to some of the neighbors in the cluster, and by definition, that point should not qualify as an outlier. There are points already in the cluster that are farther away than this one. Precision can be sacrificed in the calculation of the p-values to gain speed in diagnosing the

outliers. Experimental results have shown that this change has no effect on the correctness of the algorithm for detecting true outliers. This argument only holds if every new point is considered in isolation. If the new point is incorporated in any of the clusters, the strangeness values of the other points already there may need to be recomputed. The only interest here is in processing every point in a new batch separately, and decide individually whether each point is an outlier under the current clustering model. Once this decision is made for the whole batch of points, a process occurs by which the new batch of points is incorporated to the clustering model which is not the objective of this research. During that process, the strangeness values of the points in the clustering model will be re-computed.

StrOUD works as follows: given a batch of new points, determine for each one of them separately whether the point is an outlier; then determine if the point u is an outlier by testing c null hypotheses of the form " u is fit to belong in cluster y" $(y = 1, \cdots, c)$; and if all these c hypotheses can be rejected, the point is declared an outlier.

If clustering information is available and multiple clusters exist, multiple tests are performed with one for every cluster. The actual confidence value will be $(1 - \tau)^c$ which is smaller than $1 - \tau$. In order to obtain the desired confidence level δ, a value of τ is selected such that $(1 - \tau)^c = \delta$.

When no clusters are available, the test can be administered to the data as a whole as if it all belonged to one cluster. Doing that requires a single α_i per point as opposed to computing one per cluster, and the τ used directly reflects the confidence level that is required $(\delta = 1 - \tau)$.

Regardless if cluster information is available, the basic operation is to compare the strangeness of the point being tested with that of the points in a baseline. These points can be organized in clusters or treated as a whole. As such, StrOUD does not require that the baseline points to all be normal, but rather makes the assumption that a vast majority of these points conform to what is understood as normal. This is a very realistic assumption that goes to the core of what is commonly understood as normality. Given a random sample of points, one expects the vast majority of the points to be normal. Otherwise,

abnormal behavior would be common and would cease, by definition, to be abnormal. This assumption makes it easy to obtain baseline data for the test, by just sampling whatever data is available.

The setting of τ does not force the number of outliers to be exactly $100*\tau\%$ of the points. In fact, as will be shown in the experiments, the number of outliers is often lower than this percentage. For instance, $\tau = 5\%$ does not result in 5% of the points being flagged as outliers, but generally results in a lower percentage of outliers. This distinguishes StrOUD from those described in (1; 10) which require the user to set exactly the percentage or the number of outliers that will be flagged. Those techniques simply rank the points according to their degree of 'outlierness' and let the user decide the cutoff point. StrOUD aims to discover the true number of outliers present in the dataset given the level of confidence desired by the user. To see why StrOUD does not necessarily flag $100 * \tau\%$ of the points as outliers, consider the following extreme case. A dataset in which 99.99% of the points have the same sum of distances to their k nearest neighbors for a given value of k, and the remainder of the points (0.01 %) have much greater sum of distances to their k nearest neighbors. A sample of such a dataset is very likely to contain only normal points. When subjected to the StrOUD algorithm, normal points will not be flagged as outliers, while 0.01 % of the points will be flagged as outliers regardless of what value of τ is chosen. So, for instance, for $\tau = 0.05$, only 0.01 % of the points will be declared as outliers. The techniques in (1; 10) will simply rank the points according to their strangeness. For them, if the cutoff is set at 5 %, then 5 % of the data will be declared to be outliers, and most of which will belong to the first kind of points with the smallest sum of distances to their k nearest neighbors.

4.2 Computational Enhancements

A problem encountered when using the large and complex data sets, such as the bookstore and texture data sets, was the computational time required to calculate the nearest neighbor distances. The computation time for these nearest neighbor distance calculations when

using these large data sets was often many hours or a few days. With the goal of reducing these computation times, representatives based on sampling and one-class support vector machines were tried. Random sampling only was tried and then a combination of one-class support vector machines and random sampling was tried.

Since the StrOUD method requires finding K nearest neighbors for each cluster, $O(m)$ distance computations are needed for each point to be diagnosed, where m is the number of data points in the normal data set. To diagnose n points, the complexity would be $O(mn)$. To find the distribution of α values for the normal data set, $O(m^2)$ comparisons are required. This step is done off-line, and only once, before starting the diagnosing of outliers. However, if m is very large, the off-line computation may be very costly. In this case, to alleviate the complexity, one may sample the data set and perform comparisons only with the sampled data. Experiments are performed that show this is a reasonable approach to handle large data sets with little or no significant deterioration of the results. To reduce the number of computations, two methods have been devised to select a set of representatives from the original normal data, and then these representatives are used in the outlier computations.

4.2.1 Random Sampling

The first computational enhancement method is to select a random sample from the data and perform the computation only on the sampled data to find the distribution of strangeness values for the normal set and to diagnose the outliers. A random sampling function such as the one in Weka (54) can be used to collect a random sample of the data.

4.2.2 One-class Support Vector Machine

The second computational enhancement method is to find a hypersphere that contains all the points of the typical set, and select some points in the margin of the sphere together with points located in the interior of the sphere. To find the points in the margin, a one-class Support Vector Machine (SVM) procedure is used on the typical set to obtain the

support vectors that are located on the margin of the hypersphere. Simple sampling of the remaining points is used to obtain the interior points. The interior points are necessary, because if interior points are not sampled, new outliers may be diagnosed inside the contour as outliers, thus generating false positives. It is possible to utilize an algorithm that detects whether a point is inside the margin, but this is easily done only if the margin defines a convex hull, which may not be the case for some sets. Therefore, the interior is filled with sampled representatives from the set itself. One-class SVMs are used in my method as a data reduction technique and not as an outlier detection technique.

A one-class SVM (6) fits a hypersphere in a transformed space to include most of the given data. The hypersphere tries to capture the support within which the data is clustered in an effort to separate them from the rest of the world. This approach is well suited for scenarios in which it is reasonable to assume that the given data cluster in a certain way (i.e., the given examples are "alike" according to some similarity measure). At the same time, the "rest of the world" does not follow a probability distribution that can be estimated. For example, the given data can be a collection of faculty webpages, and the goal is to train a classifier that discriminates faculty webpages from other webpages. Clearly, a non-faculty webpage can belong to an arbitrary class.

More specifically, given the data $x_1, \ldots, x_n \in \Re^d$, let $\phi : \Re^d \to \Re^D$ be a function that maps the input data into a higher dimensional feature space. Then, a one-class SVM solves the following constrained optimization problem:

$$\min_{r \in \Re, \zeta \in \Re^n, c \in \Re^D} r^2 + \frac{1}{\nu n} \sum\nolimits_i \zeta_i \tag{4.3}$$

such that $\|\phi(x_i) - c\|^2 \leq r^2 + \zeta_i$, $\zeta_i \geq 0$ $\forall i = 1, \ldots, n$. The parameter $\nu \in [0, 1]$ sets the trade-off between the radius r of the hypersphere and the number of points the hypersphere can include where $c \in \Re^D$ is the center of the hypersphere. This optimization problem can be solved by introducing the Lagrange multipliers β_i, $i = 1, \ldots, n$, which gives the solution $c = \sum_i \beta_i \phi(x_i)$. To optimize the βs, it can be shown that the dual objective function can

be written, using the kernel trick, in terms of a kernel function. The optimization of the dual problem gives the optimal β_i values, with $0 \leq \beta_i \leq \frac{1}{\nu n}$. The data points associated to non-zero β_i values are the support vectors. To estimate the margin of the discovered clusters, select the non-bounded support vectors ($0 < \beta_i < \frac{1}{\nu n}$). In equation 4.3, if ν is set to a small value then $1/\nu$ is large, and large ζ_i values are penalized and more data are put inside the hypersphere. If ν is set to a large value, then $1/\nu$ is small, and larger ζ_i values are encouraged and the size of the hypersphere is reduced.

Experiments were conducted to compare the two techniques based on random sampling and one-class SVMs, with the percentage of representatives equal in both techniques to guarantee a fair comparison.

4.3 Global vs. Local Outliers

The complex nature of real data is that, in some cases, anomalous objects can be identified only when the distribution of the surrounding objects is considered. A global view of the data would fail to detect these anomalies. Such anomalous objects are called local outliers. StrOUD is capable of properly detecting local outliers in contrast to previously developed algorithms for outlier detection based on global parameters.

A main problem in outlier detection algorithms such as those described in Knorr (31) is that their decisions are based on global parameters. As pointed out in Breunig (10), such algorithms fail to distinguish points that are at similar distances from dense and sparse clusters, since they either diagnose all these points as outliers or as non-outliers. A point close to a sparse cluster can have its distance to neighbors comparable to those in the cluster, and therefore should be diagnosed as a non-outlier. A similarly close point to a dense cluster is likely an outlier, since the points in that cluster are closer to each other than they are to that point. In the dense cluster case, the point is rare compared to those in the cluster.

StrOUD solves this problem by correctly comparing the strangeness of the point α to the

distribution of strangeness for the cluster points. Given the confidence level, if the point's α is greater than a fraction of cluster strangeness that surpasses the confidence, the point would be declared an outlier. This is precisely what would happen in the dense cluster case. In the sparse cluster, the strangeness of the point would not surpass a large fraction of the strangeness of the cluster's points, thereby declaring the point as a non-outlier.

The Local Outlier Factor (LOF) algorithm (10) solves this problem by assigning to points a measure of relative density with respect to the density of their neighbors and ranking them according to that measure. StrOUD makes the decision by using hypothesis testing on a measure of strangeness. While not being done, it is entirely plausible to use the measure of relative density of LOF as the strangeness measure of StrOUD. A comparison of results from StrOUD and LOF is in the experimental section.

4.4 Kernel-based StrOUD

To incorporate the effects of space and time into the outlier calculation, three kernels are embedded into StrOUD (4). One kernel is for the feature measurements, the second kernel is for the spatial coordinates, and the third kernel is for time. The weighted kernel distance function is $\alpha_1(d_f) + \alpha_2(d_s) + \alpha_3(d_t)$ where $\alpha_1 + \alpha_2 + \alpha_3 = 1$, d_f = feature distance, d_s = spatial distance, and d_t = temporal distance. The user is able to input each the weights (as long as the sum of the three weights equal one) based on their domain knowledge, subject matter expertise, or a desire to give the most weight to that which is most important to their analysis. The user can also weight each kernel equally if they want each kernel to have equal importance. The total distance is the sum of the feature distance, spatial distance, and temporal distance. This total distance is then utilized as the distance measurement in StrOUD. The algorithm for Kernel-based StrOUD is shown in Figure 4.2.

4.4.1 Feature Measurements Kernel

This kernel incorporates a vectorial representation for the feature measurements at a given location and time, and a distance measure over these measurements. There is no limit to

Let there be m training examples
For $i = 1$ to m do
    Calculate and normalize D_{fi}^m, D_{si}^m and D_{ti}^m
    $D_i^m = \rho(D_{fi}^m) + \beta(D_{si}^m) + \delta(D_{ti}^m)$
    (where $\rho + \beta + \delta = 1$)
For $j = 1$ to n do
    For every test example,
    Calculate and normalize D_{fi}^n, D_{si}^n and D_{ti}^n
    $D_i^n = \rho(D_{fi}^n) + \beta(D_{si}^n) + \delta(D_{ti}^n)$
    If $D_i^n < D_i^m$
        Calculate α values for the vector of features, spatial coordinates, and time
        $\alpha = \alpha_f + \alpha_s + \alpha_t$
    Compute the p-value of the point using α
Output as confidence one minus the p-value

Figure 4.2: The Kernel-based StrOUD algorithm

the number of feature measurements.

4.4.2 Spatial Kernel

A spatial kernel was added to StrOUD to incorporate the impact of spatial location on outlier detection. Each data instance was recorded at a spatial location. The spatial distance is the distance between the x,y coordinates of the spatial locations. First, a weighted uniform distribution was used for the spatial kernel, and then a weighted exponential decay function was used. A uniform distribution is a distribution that has constant probability. The graph of a uniform distribution is a rectangle. The uniform distribution defines equal probability over a given range for a continuous distribution. A variable is uniformly distributed over the interval (0,1) if its probability density function is given by $f(x)= 1$ if $0<x<1$ and 0 otherwise. In exponential decay functions, the independent variable is t, representing time, and the general form of the exponential decay function is $f(t) = Ae^{-kt}$ where A is a constant and k is a positive constant. The constant k is called the decay rate.

4.4.3 Temporal Kernel

After the addition of the spatial kernel, a temporal kernel was added to StrOUD to incorporate the impact of time on outlier detection. Each data instance was recorded at a spatial location at a specific time. The year, month, day, and time fields were reduced to a single field (hours). The temporal distance was computed as the number of hours from a specific time to a previous time. A weighted uniform distribution was used for the temporal kernel.

4.5 Kernel Density Estimation

The computational processing time of distance-based outlier detection can sometimes be prohibitive. This is due to the fact that finding nearest neighbors is costly. For large datasets, like spatial-temporal datasets, this means unreasonable waiting time to diagnose the data.

As an alternative, a method that uses a different definition of strangeness was conceived. This method utilizes the 'surprise function' as strangeness. The surprise function is -$logP$, where P is the likelihood of a point.

In order to compute the likelihood of points, the probability density is estimated that supports the data being analyzed. In order to not make assumptions about the form of this density, a non-parametric method is utilized. Kernel Density Estimation (KDE) (14) was utilized for this method. For that purpose, a kernel needs to be utilized. In this case, a Gaussian kernel was selected.

Kernel density estimation methods are similar to histograms in that they are both built with a number of individual kernels centered on the sampled data points. The kernel function (4.4) is usually a symmetric probability density function (pdf) which is non-negative over its domain and integrates to 1 over the defined range. The value of the density at some arbitrary value x is dependent on the distance between the observed data and the shape of the kernel. The kernel function K determines the shape of the kernel. The parameter h

determines the width of the kernel and the smoothness of the estimation. The kernel density estimator can be considered to place small bumps at each observation as determined by the kernel function. The estimator consists of a sum of the bumps and usually becomes a smooth curve as a result. For this likelihood-based method, the pdf of the sample is extrapolated to the entire population.

The steps in this likelihood-based method are to input weights for each of the three kernels, the number of bins for density estmation, and the sample size to be utilized. Following the inputs, utilize the KDE function to generate the density distribution for each data dimension. The likelihood of a point is calculated using the density distribution. The distribution of surprises = -$logP$ are then calculated. StrOUD is then applied using the distribution of surprises as the strangeness values. The algorithm for Outliers KDE is shown in Figure 4.3.

$$f_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K(\frac{x - x_i}{h}) \tag{4.4}$$

```
Let m be the number of training examples and col be the number of columns in the data set
Let ρ be the features weight, β be the spatial weight, and α be the temporal weight
∑(ρ + β + α) = 1
Let n be the number of bins for density estimation
Let ss be the sample size and rs be a random sample of the data of size ss
for c = 1 to col
    [bandwidth,density,xmesh] = kde(data sample,n,min(data sample),max(data sample));
    density=density/sum(density);
    range=max-min;
    width=range/(n-1);
    for i = 1 to m
        for j = 1 to col
            for k = 2 to n
                if xmesh(k-1) <= data sample(i,j) and data sample (i,j) < xmesh(k)
                  bin = k-1;
                  probability of training point (i,j) = density(bin);
                  strangeness of training point (i,j) = -log(probability of training point (i,j);
    for u = 1 to ss
        for j = 1 to col
            for k = 2 to n
                if xmesh(k-1) <= data sample(u,j) and data sample (u,j) < xmesh(k)
                  bin = k-1;
                  probability of sample point (u,j) = density(bin);
                  strangeness of sample point (u,j) = -log(probability of sampe point (u,j);
for r = 1 to m
    combined strangeness of training point (r) = ρ ∑strangeness of feature training points;
        + β ∑strangeness of spatial training points + α ∑strangeness of temporal feature point;
for v = 1 to ss
    combined strangeness of sample point (v) = ρ ∑strangeness of feature sample points;
        + β ∑strangeness of spatial sample points) + α ∑strangeness of sample temporal point;
for w = 1 to m
    strange = 0;
    for p = 1 to ss
        if combined strangeness of training point(w) > combined strangeness of sample point(p)
            strange = strange + 1;
    pvalue = strange/ss;
    if pvalue > confidencelevel
        declare Outlier;
```

Figure 4.3: Outliers Kernel Density Estimate algorithm

Chapter 5: Empirical Evaluation

Experiments were conducted with StrOUD on real and synthetic data. In all the experiments, the data sets were used as they were encoutered without any pre-processing. For each succeeding experiment, the size and complexity of the data increased. The data sets varied from a synthetic two-dimensional data set with two clusters and the Fisher Iris data set which has three clusters and 150 records to an e-commerce bookstore data set with twelve dimensions, three clusters, and 65,536 records and texture data from the Elena project that has forty dimensions, ten clusters, and 4250 records. Experiments were conducted using the cluster information and then were conducted without the cluster information by assuming the training data was all in one cluster. For the majority of these and other experiments, the number of nearest neighbors was five and a confidence level of 95% was utilized.

It cannot be assumed that data being tested does not contain any outliers. Since most data sets are not free from outliers, it is a reasonable assumption that the typical data is contaminated. Experiments were conducted to see if StrOUD could successfully detect outliers with data contaminated with outliers and not require a noise-free data set. For these experiments, StrOUD was utilized to clean data sets corrupted by noise, and then StrOUD was utilized to detect new outliers using a corrupted, uncleaned data set. These experiments were also conducted with sampling using one-class SVMs. For these experiments, the typical data was contaminated with a number of outliers from the class chosen to act as the outliers.

Comparisons are performed with other methods, such as the distance-based(31) algorithm denoted as DB, and Local Outlier Factor(10) denoted as LOF. The LOF method does not really output outliers, but rather ranks the points according to their density measure. Therefore to conduct a fair comparison, LOF is used as the strangeness measure. LOF values of the baseline are computed and are sorted in descending order. Then the largest $m = \lceil \tau \times n \rceil$ values are considered where n is the size of the baseline and τ is the threshold

used in StrOUD for the p-value. If the LOF measure of the test point is greater or equal than the smallest of these m LOF values, the test point is declared an outlier, otherwise it is not an outlier. This process is repeated for each of the test points.

5.1 Empirical Evaluation with Numerical Data

5.1.1 Two-dimensional Data with Clusters of Different Densities

To demonstrate that StrOUD can handle clusters of different density and properly diagnose outliers which is the common problem of algorithms that use a global distance or a global density to find outliers, a two-dimensional data set with two clusters was constructed with one dense cluster and one sparse cluster. The dense cluster is centered at $(10, 10)$ and contains 1,100 points following a Gaussian distribution with covariance matrix equal to the identity matrix. The sparse cluster is centered at $(0, 0)$ and contains 30 points that follow a Gaussian distribution with covariance matrix equal to the identity matrix. The test data consists of points lying on rings surrounding the two clusters (64 points around cluster 0, and 100 points around cluster 1). The rings are designed in such a way that the sum of the distances of a point on the ring to the 5 nearest neighbors follows a very narrow distribution with mean 5.1 and standard deviation 1.2. The data, normal and test, can be seen in Figure 5.1. When running StrOUD in this dataset, the clustering information was not utilized. The dataset is considered to be in one single cluster for StrOUD calculations. This is done to be similar to LOF which does not use any clustering information. StrOUD correctly diagnosed the 64 points on the ring around the sparse cluster (centered at $(0, 0)$) as normal, and the 100 points on the ring around the dense cluster (centered at $(10, 10)$) as outliers. The same results were determined by LOF.

5.1.2 A Synthetic Data Set

A data set was constructed consisting of points grouped into three clusters, according to Gaussian distributions, with diagonal covariance matrices. The points have four dimensions

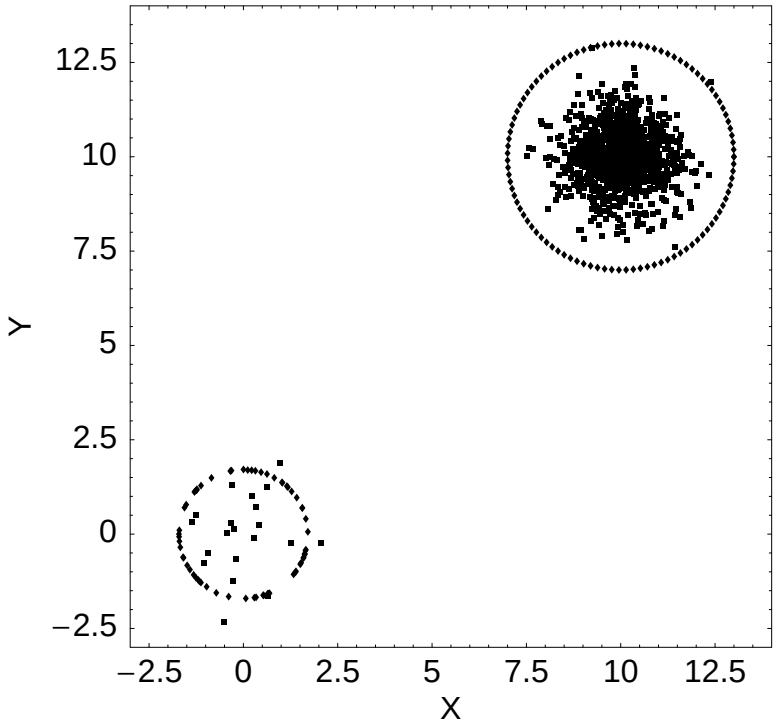

Figure 5.1: Two-dimensional Data with Clusters of Different Densities

Table 5.1: Synthetic Data Results

StrOUD							
	3 clusters						no clusters
	$\delta = 90\%$			$\delta = 95\%$			$\delta = 95\%$
k	2	5	10	2	5	10	5
TP	100%	100%	100%	100%	100%	100%	100%
FP	0%	0%	0%	0%	0%	0%	0.2%
DB				LOF			
p	0.7	0.8	0.9				
TP	100%	100%	100%	100%			
FP	2%	0%	0%	2%			

and each cluster has 500 points. The centroids of the three clusters are $(1, 1, 1, 1)$, $(7, 1, 1, 1)$, and $(3, 3, 3, 3)$, and the standard deviations used to generate the points are $(1, 2, 1, 2)$, $(2, 1, 2, 1)$, and $(1, 1, 2, 3)$, respectively. A test set was constructed consisting of 50 points that were more than 3 standard deviations away from the centroids (outliers) and 50 points that follow the same distribution as the cluster points (non-outliers). The results of the experiments with the synthetic data are summarized in Table 5.1 showing the True Positive (TP) and False Positive (FP) rates for StrOUD, DB, and LOF. Across a range of choices for k (2, 5, 10) and global confidence (90%, 95%, which result from choosing $\tau = 0.035$, and 0.017 respectively), all the outliers (100% true positives) are flagged. The same result is obtained when no cluster information is used with 0.2% false positives determined. For DB, one of the parameters of their method was varied (i.e., p, the minimum fraction of tuples lying outside the D-neighborhood of an outlier) from 0.7 to 0.9. For all the cases, the true positives remained at 100%. The false positives were 0, except for the case of $p = 0.7$, which determined 2% false positives. The results are comparable to those obtained by StrOUD. LOF also obtains comparable results.

Table 5.2 shows the running times for StrOUD using the synthetic data. For each experiment, a different number of baseline points were used which are shown as size in the table. A total of 100 test points were used for the test.

Table 5.2: Running times for StrOUD using synthetic data.

size	StrOUD (sec.)
125	2.2
250	2.7
500	3.4
1,000	8.2
1,500	12.6

5.1.3 National Hockey League data

A test was performed using NHL player's statistics, that can be obtained from web sites such as nhlstatistics.hypermart.net for the 1994 season. The reason for this test is to compare the behavior of StrOUD with DB, and the NHL 1994 data set is one of the case studies the authors of DB investigated. A four-dimensional description is used for each player with the following attributes: the plus-minus statistic that indicates how many more event-strength goals were scored by the team when the player was on the ice, the number of penalty minutes, the percentage points, and goals scored. After tuning the parameters D and p, where D is the radius and p is the minimum fraction of records that must be found outside of the D-neighborhood, for the distance-based outliers code, seven outliers were determined by the distance-based code. Added to these seven records were the records of 27 other players selected at random to perform a test with StrOUD. The remaining 834 players in the data set were grouped into three clusters using K-means. A value of $\tau = 0.025$ was used for a total confidence of 95%. StrOUD found the seven outliers and added three more to the list for a total of ten outliers out of the 34 records tested.

Figure 5.2 shows the histogram of distances to the closest centroid for the normal points in the NHL data. Figure 5.3 shows the histogram of distances to the closest centroid for the outliers in the NHL data. Each bar in the histogram shows the percentage of points in the group that have distances to the closest centroid in the range indicated. These figures show that the outliers found by StrOUD are radically different with respect to the points declared non-outliers which justifies the finding of the extra three outliers.

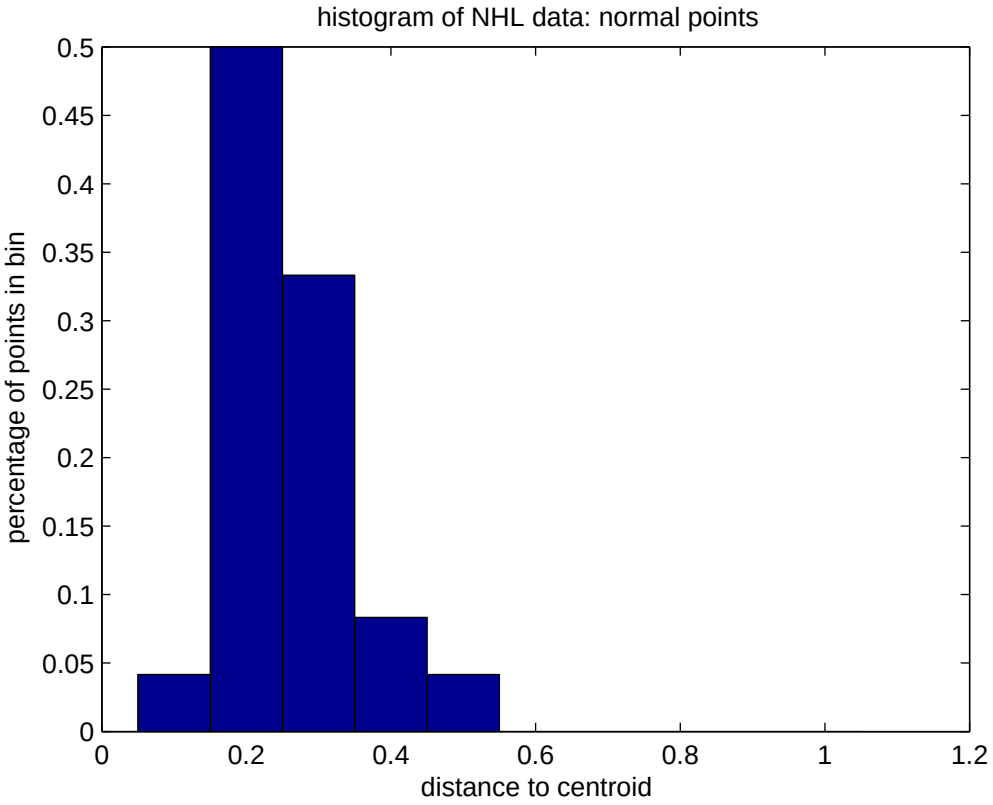

Figure 5.2: Histogram of normal points in NHL data

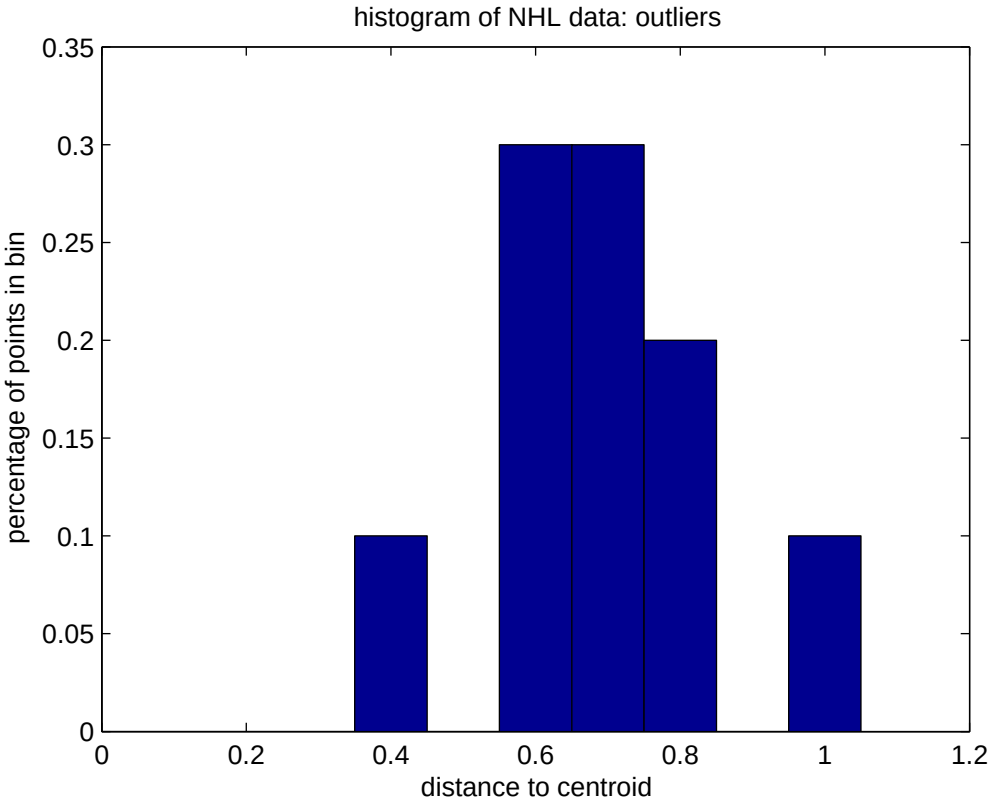

Figure 5.3: Histogram of outliers in NHL data

5.1.4 Fisher Iris Data

The Fisher-Iris data is a widely used data set for supervised learning which can be found at the UCI machine learning repository (51). The data set contains three classes of records, and each class corresponds to a type of Iris plant. There are 50 records of each class in the set. This set was chosen because one of the classes (Iris Setosa) is clearly separable from the other two. Taking away the class attribute, two clusters are formed with 45 records from each of the two classes Virginica and Versicolor. The records of the Setosa class plus the remaining records of the other two classes (5 each) are used to test the StrOUD detection algorithm. A good outlier detection technique should be able to recognize the Setosa records as outliers without giving false positives. Results of this experiment declared 100% of the Setosa records as outliers. The other 10 records were not flagged as outliers, giving a 100% true positive rate and a 0% false positive rate. For each Setosa point the distance to the closest centroid of the two clusters representing the Versicolor and Virginica classes was calculated and these distances are in a small range between 2.86 to 3.79, with a mean of 3.24, and a small standard deviation of 0.20. The range for the "normal" data (Versicolor and Virginica points) is 0.23 to 1.80, with a mean of 0.73, and a standard deviation of 0.37.

To maintain a confidence of 95%, the $\tau = 0.025$ ($0.975^2 \approx 0.95$). These results hold even when the clustering information is not utilized (i.e., all the 90 records of Virginica and Versicolor are put in one cluster) using $\tau = 0.05$. For both experiments, $k = 5$ was utilized. For this experiment, LOF also achieves a 100% true positive rate and a 0% false positive rate.

5.1.5 Bookstore Data

This data set was generated from an e-commerce workload and has been used previously in (39). The logs correspond to some weekdays in which a large number of HTTP requests were processed. Entries corresponding to images, errors, etc., were deleted and the URLs of the remaining entries in the log were mapped to one of 12 e-business functions such

Table 5.3: Bookstore Data Results for Different Values of k.

k	1	5	10	20	LOF
TP	98.33%	99.44%	99.44%	99.72%	99.0%
FP	0.23%	0.56%	0.59%	0.54%	4.5%

as "add item to cart," and "pay." A session vector was generated for each session. This vector indicates the number of times that each of the 12 different functions (e.g., Search, Browse, Add) was invoked during the session (12 dimensions). The "ground truth" in this experiment consists of the domain knowledge that robot sessions are those with a total number of "clicks" greater than or equal to 50. Those records were separated from the data set along with some non-outliers used to test the algorithm. The basic clustering model was found using K-means clustering over a set of records that do not contain any robot sessions. The members of each cluster differ on the intensity and characteristics of the sessions. The first cluster contains records with low activity (few clicks), the second cluster contains records with moderate activity, and the third cluster contains records with high activity. The last two clusters contain records of sessions in which the "Add" function was performed considerably more often than in the sessions represented by the records in the first cluster. (e.g., cluster 3 sessions correspond to heavy buyers in the bookstore). A total of 101,808 records were used to form the original cluster model. The technique was tested with 65,536 records of which 360 were known to be outliers and the remaining 65,176 records were non-outliers.

Table 5.3 shows the results of running StrOUD on this data set with different values of k using $\tau = 0.017$ to obtain a total confidence level of 0.95 ($0.983^3 \approx 0.95$). A large percentage of the outliers are detected by the test while the false positive rate is kept low. The results throughout the range of k are stable. The table also shows the results of running LOF on this data. For LOF, the TP is comparable to that obtained by StrOUD. The false positive rate is much higher than StrOUD.

Figure 5.4 shows the histogram of distances to the closest centroid for the non-outliers

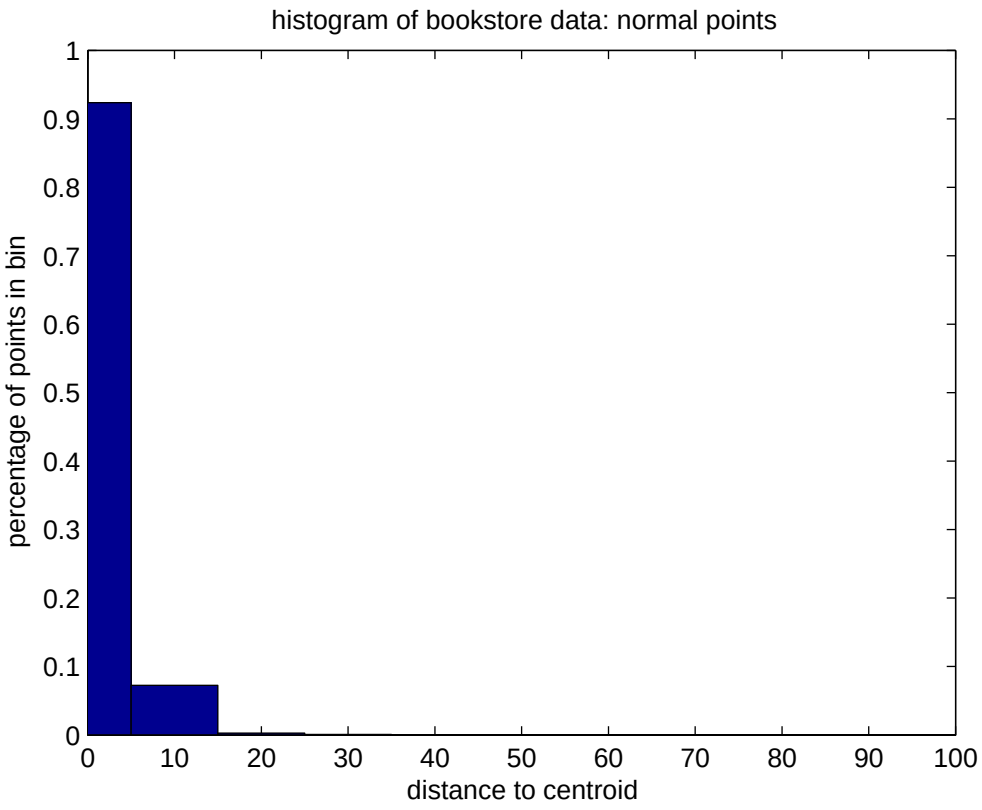

Figure 5.4: Histogram of non-outliers in the bookstore data

in the bookstore data. Figure 5.5 shows the histogram of distances to the closest centroid for the outliers in the bookstore data. Each bar shows the percentage of points within each group of non-outliers or outliers with distances to the closest centroid in the range indicated. The distributions are very different with the bulk of the outliers concentrating at distances greater than 7 while most of the non-outliers have distances less than 7.

Experiments were conducted varying the clustering model. Using K-Means and three different seeds, different clusterings were obtained for the baseline data. The diagnoses for each clustering were compared, and on average, 98% of the test points received the same determination in the presence of two different clusterings of the same baseline data. This proves that the method is very robust with respect to changes in the clustering model.

In order to compare StrOUD with DB for this data set, Principal Component Analysis

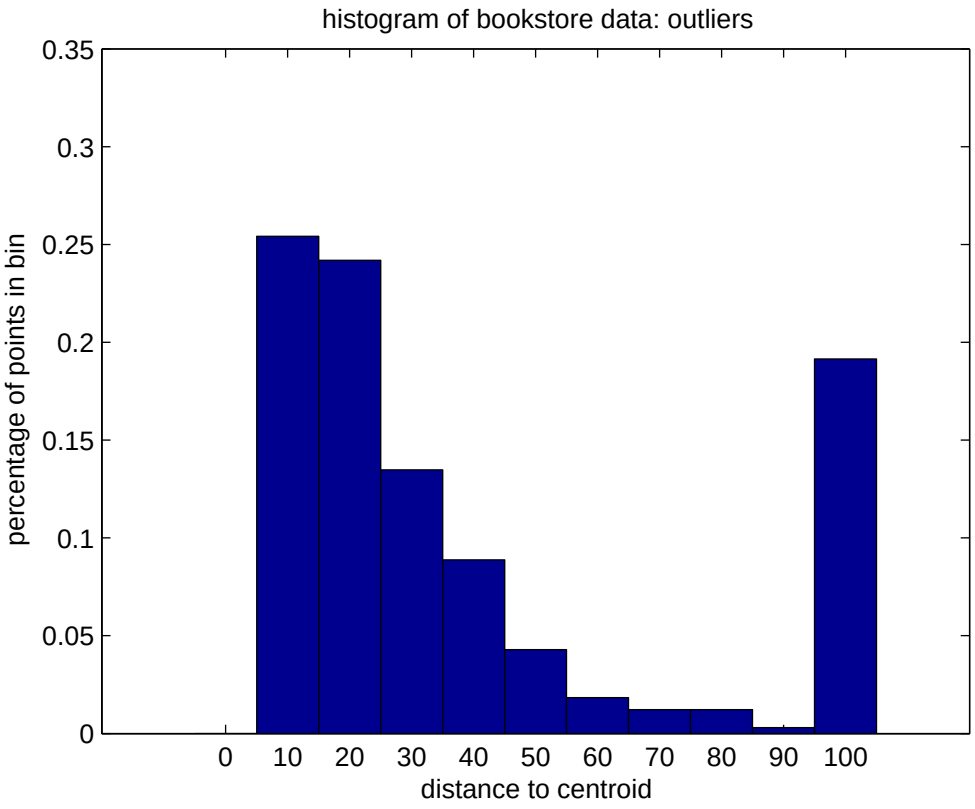

Figure 5.5: Histogram of outliers in the bookstore data

Table 5.4: Bookstore Data PCA Results

	StrOUD			DB		
	$k = 2$	$k = 5$	$k = 10$	$p = 0.7$	$p = 0.8$	$p = 0.9$
TP	98.06%	98.61%	97.78%	6.39%	2.22%	2.22%
FP	0.13%	0.26%	0.30%	0%	0%	0%

was utilized by representing the data using the first four principal components, because the code received for DB only worked for four dimensions. Table 5.4 shows the results for StrOUD and DB. The results are comparable to those shown in Table 5.3. The fraction of true positives is around 1% smaller than when the entire data set is used, and the fraction of false positives is about the same. For DB, when the value of p was decreased, it was possible to identify only a small fraction of the true outliers (less than 7%). The technique however, does not introduce any false positives for this data set.

Table 5.5 shows the results of experiments, that do not use the clustering information, on the original dataset, the PCA-reduced dataset, and the sampled dataset. In the first column which is labelled original in the Table, the test was conducted on the original bookstore data without the cluster information. The True Positive rate (99.72%) is comparable to the one obtained using the clusters (99.44%), while the False Positive rate is larger but still very manageable (4.63%). The second test uses the 4-dimensional data obtained by PCA. The results are comparable to the original data (TP=100%, FP=4.56%). The last two tests were conducted by using a sample of the original data as the typical set. The sample sizes were 1% and 10% respectively. The results show perfect recognition of the outliers in both cases (TP=100%), with a very slight increase of the False Positive rate (5.16% to 5.74% for the 1% and 10% samples respectively). These experiments demonstrate that when the data set is large, it is possible to use a sample of the normal data to capture outliers without significant loss of accuracy.

Table 5.5: Bookstore Data Results without Cluster Information (k=5).

	original	PCA	1% sample	10 % sample
TP	99.72%	100%	100%	100%
FP	4.63%	4.56%	5.16%	5.74%

5.1.6 Texture data

This is one of the real data sets of the Elena project which can be found in (16). The data set was generated by the Laboratory of Image Processing and Pattern Recognition (INPG-LTIRF) in Grenoble, France, using as the original source the material from (11) and referenced in (21) and (22). The data set contains 11 classes and 40 dimensions. The assigned labels for the 11 classes are non-consecutive integers: 2, 3, 4, 6, 7, 8, 9, 10, 12, 13, and 14. Class 13 is the held-out class used as outliers. The original aim was to distinguish between 11 different textures (Grass lawn, Pressed calf leather, Handmade paper, Raffia looped to a high pile, Cotton canvas, etc.) with each pattern (pixel) being characterized by 40 attributes built by the estimation of fourth order modified moments in four orientations: 0, 45, 90 and 135 degrees. Again, the class attribute is discarded. The objective is to detect the points in one class as outliers with respect to the points in other classes. This experiment used all the records of the texture data as the initial points with the exception of 500 points that belong to one of the classes and a sample of 5% (223 records) of other classes that were then included in the test set. Each of the other classes was represented by a cluster, for a total of 10 clusters and 4,250 records. The sample size is 4,250 x 0.05 = 212.5 plus a rounding error of 1 for each of the 10 clusters that resulted in 223 sampled records. With $k = 5$ and $\tau = 0.005$ (to reach a final confidence of 95%, as $0.995^{10} \approx 0.95$), a true positive rate of 99.8% was obtained with no false positives. When the clustering information is ignored, the true positive rate is 99.8% with 2.2% false positives. See Table 5.9 under All Data. The results for LOF are 99.60% for the true positive rate and 5.86% for the false positive rate.

Figure 5.6 shows the histogram of distances to the closest centroid for the non-outliers

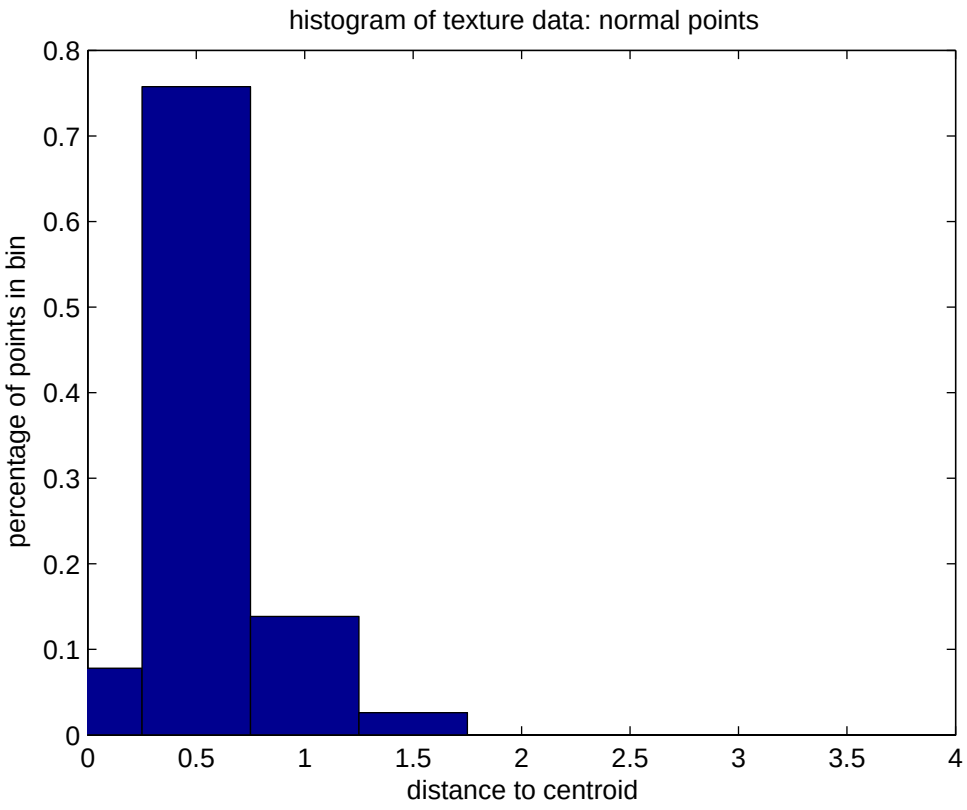

Figure 5.6: Histogram of non-outliers in the texture data

in the texture data. Figure 5.7 shows the histogram of distances to the closest centroid for the outliers in the texture data. Each bar shows the percentage of points within each group of non-outliers or outliers with distances to the closest centroid in the range indicated. The two distributions are very different with little overlap on the distances.

In order to compare StrOUD with DB for this data set, Principal Component Analysis was performed and the data represented using the first four principal components. Table 5.6 shows the results for StrOUD and DB. Although the percentage of true positives (outliers) captured is smaller than that obtained when using the entire data set, StrOUD still captures a large fraction of the outliers with no false positives to report. For DB, since the 500 points of class 13 are somewhat close to each other, it is necessary to lower the value of p to capture them as outliers, but by doing so, false positives are introduced.

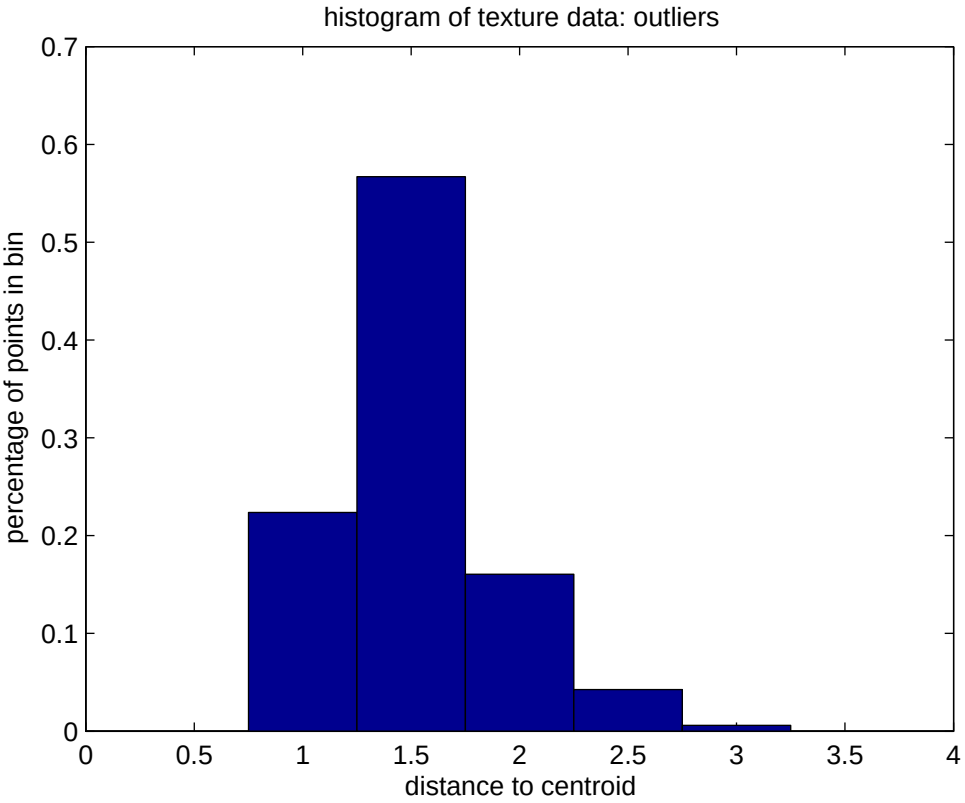

Figure 5.7: Histogram of outliers in the texture data

Table 5.6: Texture Data PCA Results

	StrOUD			DB		
	$k = 2$	$k = 5$	$k = 10$	$p = 0.7$	$p = 0.8$	$p = 0.9$
TP	95.4%	92.7%	90.9%	100%	14.8%	0.4%
FP	0%	0%	0%	42.74%	12.87%	0.22%

5.1.7 Shuttle data

Another data set utilized in the experiments was the NASA Shuttle data set. The shuttle data set contains nine attributes all of which are numerical, and seven classes labeled from one to seven. Approximately 80% of the data belongs to class 1. The shuttle data has been utilized to build supervised models to predict each of the classes. The data consists of $43,500$ instances and a test set of $14,500$ instances. Since Class 3 only has 132 instances in the training data set and 39 instances in the test data set for a total of 171 outliers, Class 3 was selected to be the outliers in the shuttle data experiments. Thus 132 instances of Class 3 were removed from the training data resulting in a set of 43368 instances that were used as the baseline. The 132 instances of Class 3, that were removed, were placed in the test set to form a test dataset of 14632 instances of which 171 were true outliers. Each original class, except for class 3, in the training dataset was placed in a cluster except for the no cluster experiment. Table 5.7 lists the results of experiments to enhance the performance using sampling and One-class SVM (k=5) with clusters. The 171 instances of cluster 3 are the outliers. Results using the entire data set are reported in the first column, results using sampling are shown in the second column, and results using one-class SVM are in the third column. For each technique, the percentages of True Positive (TP) and False Positive (FP) are given. For one-class SVM, the number of support vectors used with the corresponding percentage per cluster, and the number of sampled non-support vectors with corresponding percentage per cluster are shown. Similar results were obtained with and without cluster information. In both cases 100% TP rate was achieved. Without cluster information, the false positive rate is 4.2%, and with cluster information, it is a slightly lower 2.7%. The results for LOF are TP=100% and FP=4.5%. A comparison with DB was not made for this dataset due to the code limitations of DB.

Table 5.7: Shuttle Data Results

	All Data	Sampling	One-class SVM
% Data	100%	10%	10%
Cluster 1			
SVs			256 (0.8%)
Non-SVs			3128 (9.2%)
Cluster 2			37 (100%)
Cluster 4			
SVs			64 (1.0%)
Non-SVs			603 (9.0%)
Cluster 5			
SVs			25 (1.0%)
Non-SVs			220 (9.0%)
Cluster 6			6 (100%)
Cluster 7			11 (100%)
TP	100%	100%	100%
FP	4.2%	5.0%	2.7%

5.1.8 Choosing Representatives

Experiments were conducted to choose representatives based on the sampling and one-class SVM methods. The LIBSVM (6) library for the one-class SVM was utilized. A Gaussian kernel was used in all experiments. Results obtained with no clustering information are given in Table 5.8. Tables 5.9, 5.10, and 5.7 show the results when clustering information is used.

Table 5.8 details the results of experiments to enhance the performance using one-class SVM (k=5) without clustering information and sampling. Results using the entire data set are also reported. For each technique, the percentages of True Positive (TP) and False Positive (FP) are provided. For One-class SVM, the results for different percentage combinations of support vectors (SVs) and non-support vectors (Non-SVs) are shown. The value of ν (i.e., the coefficient that controls overfitting) and the width of the Gaussian kernel were varied, so that at least 10% of the entire data set was selected as non-bounded support

vectors. A fraction of non-bounded support vectors and non support vectors were randomly selected as representatives. All percentage combinations are listed in Table 5.8, which also lists the results for random sampling and for the entire data set. All results (TP and FP) are averaged over three independent runs of random sampling. One-class SVM provided very high TP and low FP rates for all percentage combinations of SVs and non-SVs. This result shows that this approach is robust with respect to the range of tested percentages. All combinations of non-bounded SVs and non-SVs provided a very good estimation of the underlying data distribution. All experiments with one-class SVM, except for all the Texture data, resulted in lower FP rates than sampling and using all the data. This is probably because false positives are likely to be points located close to the contour of the clusters and the use of non-bounded support vectors improves their proper diagnosis.

Tables 5.9, 5.10, and 5.7 describe the results when clustering information is used for Texture, Bookstore, and Shuttle data, respectively. For each cluster, approximately 1% of non-bounded SVs were sampled. Other combinations are reported for Texture (Table 5.9), Bookstore (Table 5.10), and Shuttle data (Table 5.7). Again, the one-class SVM helps reduce the FP rates with respect to random sampling. The TP rates are always very high across all percentages.

Table 5.9 details the results of experiments to enhance the performance using sampling and one-class SVM (k=5) with clusters. Results using the entire data set are also reported. For each technique, the percentages of True Positive (TP) and False Positive (FP) are given. For One-class SVM the number of support vectors used with the corresponding percentage per cluster, and the number of sampled non-support vectors with corresponding percentage per cluster are shown.

Table 5.10 details the results of experiments to enhance the performance using sampling and One-class SVM (k=5) with clusters. The outliers (robot sessions) are those with a total number of "clicks" greater than or equal to 50. Results using the entire data set are also reported. For each technique, the percentages of True Positive (TP) and False Positive (FP) are given. For One-class SVM,the number of support vectors used with the corresponding

Table 5.8: Results using One-class SVM (k=5)

Texture data

Method	*SVs*	*Non-SVs*	*TP*	*FP*
SVM w/o clusters	10%	0%	99.8%	3.1%
SVM w/o clusters	8%	2%	99.8%	4.0%
SVM w/o clusters	6%	4%	99.8%	3.6%
SVM w/o clusters	4%	6%	99.8%	4.0%
SVM w/o clusters	2%	8%	99.8%	2.7%
SVM w/o clusters	1%	9%	99.8%	3.4%
All Data	–	–	98.8%	2.2%
10% Sampling	–	–	99.8%	7.2%

Bookstore data

Method	*SVs*	*Non-SVs*	*TP*	*FP*
SVM w/o clusters	10%	0%	99.72%	3.1%
SVM w/o clusters	8%	2%	99.72%	2.47%
SVM w/o clusters	6%	4%	99.72%	1.62%
SVM w/o clusters	4%	6%	99.72%	0.73%
SVM w/o clusters	2%	8%	99.72%	0.82%
SVM w/o clusters	1%	9%	99.72%	0.64%
All Data	–	–	99.72%	4.63%
10% Sampling	–	–	99.72%	5.74%

Shuttle data

Method	*SVs*	*Non-SVs*	*TP*	*FP*
SVM w/o clusters	10%	0%	98.27%	2.9%
SVM w/o clusters	8%	2%	100%	3.7%
SVM w/o clusters	6%	4%	100%	2.4%
SVM w/o clusters	4%	6%	100%	3.1%
SVM w/o clusters	2%	8%	100%	2.9%
SVM w/o clusters	1%	9%	100%	2.7%
All Data	–	–	100%	4.2%
10% Sampling	–	–	100%	5.0%

percentage per cluster, and the number of sampled non-support vectors with corresponding percentage per cluster are shown.

Table 5.11 reports the running times of StrOUD when applied on the entire texture, bookstore, and shuttle data sets and on 10% of the data obtained either by random sampling or by means of one-class SVMs. Table 5.12 reports the running times when keeping a baseline of 10% of the bookstore data and varying the size of the bookstore test set. The computation of the p-values was implemented in Python. The sampling process was carried out in Weka (54). The computation time to perform sampling is negligible, and therefore not included in the table. The experiments were performed on a Dell Inspiron 8600 laptop computer running Windows XP, with a 2 GHz processor, 512 MB main memory, and 60 GB hard drive.

SVM without cluster information provides the best running times for all three data sets. The case of 10% sampling is fastest for the bookstore and shuttle data, but SVM w/clusters is very slightly faster than 10 % sampling for the texture data. The running time for SVM with and without clusters is about the same for the texture, bookstore, and shuttle data, and is about 10 times faster than all the texture data, about 200 seconds faster than all the bookstore data, and about 20 times faster than all the shuttle data.

5.1.9 Cleaning the Data

The objective of these experiments is to clean a data set containing outliers without relying on the presence of an outlier-free training file. In this section, the results of experiments aimed to clean a data set containing outliers are shown without relying on the presence of an outlier-free sample. The first data cleaning experiment used the texture data, and contaminated the normal data with a number of outliers from the class chosen to act as outliers. Then, a sample of the dataset was used as the normal set, and the entire dataset was used the test set. The only difference is, that in order to be transductive, whenever a data point is tested, the strangeness values of the training set are computed without including that point. Table 5.13 details the results of cleaning the texture data contaminated with

Table 5.9: Texture Data Results

		Sampling		One-class SVM	
	All Data	(1)	(2)	(1)	(2)
% Data	100%	5%	10%	10%	10%
Cluster 0					
SVs				6(1.4%)	23(5.4%)
Non-SVs				37(8.7%)	21(5.0%)
Cluster 1					
SVs				6(1.4%)	22(5.0%)
Non-SVs				37(8.5%)	21(4.8%)
Cluster 2					
SVs				6(1.4%)	22(5.3%)
Non-SVs				36(8.6%)	21(5.0%)
Cluster 3					
SVs				5(1.2%)	22(5.3%)
Non-SVs				36(8.7%)	20(4.9%)
Cluster 4					
SVs				6(1.4%)	23(5.2%)
Non-SVs				37(8.4%)	21(4.8%)
Cluster 5					
SVs				7(1.6%)	22(5.1%)
Non-SVs				37(8.6%)	21(4.9%)
Cluster 6					
SVs				6(1.4%)	22(5.0%)
Non-SVs				37(8.5%)	21(4.8%)
Cluster 7					
SVs				6(1.5%)	21(5.2%)
Non-SVs				36(8.9%)	20(4.9%)
Cluster 8					
SVs				5(1.2%)	22(5.3%)
Non-SVs				36(8.7%)	20(4.8%)
Cluster 9					
SVs				6(1.4%)	23(5.4%)
Non-SVs				37(8.7%)	21(4.9%)
TP	99.8%	99.8%	99.8%	99.8%	99.8%
FP	2.2%	7.4%	7.0%	3.4%	3.6%

Table 5.10: Bookstore Data Results

					One-class SVM	
	All Data	Sampling	(1)	(2)	(3)	(4)
% Data	100%	10%	16.8%	10%	10.1%	10%
Cluster 0						
SVs			493 (1.0%)	493 (1.0%)	493 (1.0%)	493 (1.0%)
Non-SVs			4182 (9.0%)	4182 (9.0%)	2325 (5.0%)	4182 (9.0%)
Cluster 1						
SVs			0	32 (2.0%)	507 (28.4%)	51 (2.9%)
Non-SVs			1787 (100%)	163 (9.1%)	565 (31.6%)	139 (7.8%)
Cluster 2						
SVs			0	32 (1.5%)	406 (19.3%)	41 (1.9%)
Non-SVs			2105 (100%)	188 (8.9%)	794 (37.7%)	184 (8.7%)
TP	99.44%	100%	100%	100%	100%	100%
FP	0.56%	5.7%	0.44%	0.60%	0.54%	0.68%

Table 5.11: Total Running Times (in seconds)

Texture	SVM model(s)	Total time
All Data	–	378.92
10% Sampling	–	32.49
SVM w/o clusters	1.14	36.02
SVM w/ clusters	0.20	30.37
Bookstore	SVM model(s)	Total time
All Data	–	1242.86
10% Sampling	–	198.20
SVM w/o clusters	19.75	974.28
SVM w/ clusters	41.77	1084.82
Shuttle	SVM model(s)	Total time
All Data	–	20799.11
10% Sampling	–	1261.37
SVM w/o clusters	457.9	1310.81
SVM w/ clusters	84.82	1327.62

Table 5.12: Bookstore Data Test Set Running Times (in seconds)

Number of test records	Test time	Total time
65	4.599	76.568
650	17.108	89.167
6500	76.519	148.578
65536	806.425	878.484

a percentage of outliers. The number of outliers in the data set, and its corresponding percentage with respect to the entire data set are shown. The results shown in Table 5.13 indicate that almost all the outliers present in the data set can effectively be determined when the noise (outliers) injected to the data is kept at a low level. Each block of the table corresponds to a level of contamination. The number in the noise row is the number of outliers put into the data set and its percentage of all the data. The sample utilized as baseline is 10% of the data, and the results are obtained by averaging results of 10 samples.

A much larger number of outliers in the data (e.g., more than 10%) brings about a decrease of the true positive rate (less than 40%), but, 10% of noise is a large contamination. The false positive rate is kept very low throughout the experiments (between 1.8% and 4%).

Data cleaning experiments were also run with the bookstore and shuttle data sets. Table 5.14 shows the results of cleaning the bookstore data contaminated with a percentage of outliers. The number of outliers in the data set, and its corresponding percentage with respect to the sampled data set are shown. These bookstore data cleaning experiments use all the data and a set representatives obtained by the one-class SVM method. Similarly, Table 5.15 shows the results of cleaning the shuttle data contaminated with a percentage of outliers. The number of outliers in the data set, and its corresponding percentage with respect to the sampled data set are shown. These shuttle data cleaning experiments use class 3 as the outliers. Since not enough outliers were available in the bookstore and shuttle data sets, in order to maintain the levels of contamination used in the texture data experiments, the typical data is undersampled. The results obtained are similar to those from the texture cleaning experiments.

5.1.10 Using a Noisy Data Set

This section describes what happens if the normal data is contaminated with outliers and is later used to further detect other outliers. This is different from the experiments in the previous section. The goal of the experiments in this section is to diagnose outliers using contaminated data instead of cleaning the contaminated data.

Table 5.13: Results of Cleaning the Texture Data

Noise	10 (0.24%)	10 (0.24%)	10 (0.24%)	10 (0.24%)	10 (0.24%)
10% sample	426	426	426	426	426
k	5	10	20	40	100
TP	100%	99.8%	99%	97.8%	95.8%
FP	2.7%	2.7%	2.2%	1.8%	1.8%
Noise	20 (0.47%)	20 (0.47%)	20 (0.47%)	20 (0.47%)	20 (0.47%)
10% sample	427	427	427	427	427
k	5	10	20	40	100
TP	99.8%	98.6%	97%	94.2%	91%
FP	3.1%	3.1%	3.1%	2.7%	2.7%
Noise	50 (1.16%)	50 (1.16%)	50 (1.16%)	50 (1.16%)	50 (1.16%)
10% sample	430	430	430	430	430
k	5	10	20	40	100
TP	99.8%	98.2%	95.8%	90.2%	86.4%
FP	3.6%	3.6%	3.1%	3.1%	3.1%
Noise	100 (2.3%)	100 (2.3%)	100 (2.3%)	100 (2.3%)	100 (2.3%)
10% sample	435	435	435	435	435
k	5	10	20	40	100
TP	47%	47.4%	49%	52.4%	55%
FP	1.8%	1.8%	1.3%	1.3%	0.9%
Noise	300 (6.59%)	300 (6.59%)	300 (6.59%)	300 (6.59%)	300 (6.59%)
10% sample	455	455	455	455	455
k	5	10	20	40	100
TP	21.4%	27.4%	29.8%	40%	49.2%
FP	4%	3.6%	3.6%	3.1%	2.7%
Noise	500 (10.53%)	500 (10.53%)	500 (10.53%)	500 (10.53%)	500 (10.53%)
10% sample	475	475	475	475	475
k	5	10	20	40	100
TP	19.8%	21.6%	24.8%	33.2%	39.8%
FP	3.1%	3.1%	3.1%	2.7%	2.7%

Table 5.14: Results of Cleaning the Bookstore Data

Noise	10 (0.23%)	10 (0.23%)	10 (0.23%)	10 (0.23%)	10 (0.23%)
10% sample	426	426	426	426	426
k	5	10	20	40	100
TP	100%	99%	96.8%	94.8%	92.7%
FP	3.4%	3.2%	2.1%	1.9%	1.7%
Noise	20 (0.47%)	20 (0.47%)	20 (0.47%)	20 (0.47%)	20 (0.47%)
10% sample	427	427	427	427	427
k	5	10	20	40	100
TP	99.2%	97.6%	96.4%	93%	90.2%
FP	3.7%	3.5%3.4%	2.5%	2.4%	
Noise	50 (1.16%)	50 (1.16%)	50 (1.16%)	50 (1.16%)	50 (1.16%)
10% sample	430	430	430	430	430
k	5	10	20	40	100
TP	96.2%	95%	92.7%	86.6%	82.7%
FP	4.4%	3.9%	3.1%	3.0%	2.9%
Noise	100 (2.3%)	100 (2.3%)	100 (2.3%)	100 (2.3%)	100 (2.3%)
10% sample	435	435	435	435	435
k	5	10	20	40	100
TP	65.9%	65.5%	64.7%	63.2%	63%
FP	1.7%	1.7%	1.6%	1.5%	1.5%
Noise	230 (6.59%)	230 (6.59%)	230 (6.59%)	230 (6.59%)	230 (6.59%)
10% sample	349	349	349	349	349
k	5	10	20	40	100
TP	44.7%	45.4%	49.4%	51.2%	52.1%
FP	4.4%	4.2%	4%	3.7%	3.7%
Noise	360 (10.53%)	360 (10.53%)	360 (10.53%)	360 (10.53%)	360 (10.53%)
10% sample	342	342	342	342	342
k	5	10	20	40	100
TP	22.7%	32.6%	34.2%	39.1%	40.9%
FP	3.2%	3.1%	3%	2.7%	2.6%

Table 5.15: Results of Cleaning the Shuttle Data

Noise	10 (0.23%)	10 (0.23%)	10 (0.23%)	10 (0.23%)	10 (0.23%)
10% sample	426	426	426	426	426
k	5	10	20	40	100
TP	100%	100%	99.8%	97.9%	93.8%
FP	3.7%	3.7%	3.6%	3.6%	3.3%
Noise	20 (0.47%)	20 (0.47%)	20 (0.47%)	20 (0.47%)	20 (0.47%)
10% sample	427	427	427	427	427
k	5	10	20	40	100
TP	100%	100%	97.9%	92.5%	90%
FP	3.8%	3.7%	2.4%	2.2%	1.9%
Noise	50 (1.16%)	50 (1.16%)	50 (1.16%)	50 (1.16%)	50 (1.16%)
10% sample	430	430	430	430	430
k	5	10	20	40	100
TP	97.8%	96.9%	94.1%	88.8%	85.9%
FP	4.9%	4%	2.8%	2.4%	2.3%
Noise	100 (2.3%)	100 (2.3%)	100 (02.3%)	100 (2.3%)	100 (2.3%)
10% sample	435	435	435	435	435
k	5	10	20	40	100
TP	68.7%	67.9%	67%	65.2%	64.3%
FP	2.3%	2.2%	2%	1.2%	1.1%
Noise	135 (6.59%)	135 (6.59%)	135 (6.59%)	135 (6.59%)	135 (6.59%)
10% sample	205	205	205	205	205
k	5	10	20	40	100
TP	44.4%	45.5%	50.9%	53.9%	54%
FP	4.9%	4.8%	4.7%	4.6%	4.5%
Noise	171 (10.53%)	171 (10.53%)	171 (10.53%)	171 (10.53%)	171 (10.53%)
10% sample	162	162	162	162	162
k	5	10	20	40	100
TP	27.4%	30.1%	40.2%	47.4%	49.6%
FP	2.9%	2.8%	2.7%	2.2%	2.1%

The first experiment was conducted using the texture data and the typical data was contaminated with outliers from the chosen class. This data set was then used as the basis for diagnosing the test set without the outliers that were already used to contaminate the typical data. As a result, the test set contains the remainder of the data from the class that was not included in the typical set (outliers) plus 223 non-outliers. Table 5.16 shows the results of these texture data experiments using a contaminated typical set. The number of outliers in the typical data set and its corresponding percentage with respect to the entire set are shown. The number of outliers, labeled as Noise, used to contaminate the typical set is given in the first row of the Table. Again, a high percentage (greater than 98%) of outliers in the test data was diagnosed except for the cases (k=10 and noise=50; and k=40 and noise=100) with a low false positive rate (less than 3%) when a value of k commensurable with the number of contaminants is used. Since this value is not known, a "rule of thumb" value for k must be used, but the experiments show that a relatively low value of k (100, or 2.3% of the normal data set examples) can effectively be used to diagnose a large number of the outliers. Experiments with contaminated data were also run with the bookstore and shuttle data sets using k=5. Table 5.17 shows the results of diagnosing the bookstore data using a contaminated typical set with (k=5). The number of outliers in the typical data set and its corresponding percentage with respect to the entire set are shown. Table 5.18 shows the results of diagnosing the shuttle data using a contaminated typical data set with (k=5). The number of outliers in the typical data set and its corresponding percentage with respect the entire set are shown.. For the Shuttle data, class 3 was used as the outliers class. Results are consistent across the three data sets.

Table 5.19 shows the results of performing an experiment in which the texture data is first cleaned using the procedure explained in Section 5.1.9, and the resulting dataset without diagnosed outliers, is utilized to diagnose the same test set used for the experiments in Table 5.16. Table 5.19 shows the results of diagnosing the texture data by first cleaning a contaminated typical data set, and then using it to diagnose outliers on a test set. The number of outliers in the typical data set and its corresponding percentage with respect

Table 5.16: Texture Data Results Using a Contaminated Typical Dataset

Noise	10 (0.23%)		20 (0.46%)		50 (1.16%)			100 (2.29%)	
k	5	10	5	10	10	20	100	40	100
TP	100%	100%	100%	100%	95.5%	98.1%	100%	85.4%	98.8%
FP	2.9%	2.7%	2.8%	2.7%	2.1%	2.8%	1.6%	2.0%	1.7%

Table 5.17: Bookstore Data Results Using a Contaminated Typical Dataset

Noise	36 (10%)	90 (25%)	180 (50%)
TP	99.8%	97.0%	97.1%
FP	2.9%	2.9%	2.9%

to the entire set are shown. The results show a large TP rate, even as the percentage of noise in the original set increases. The FP rate remains stable throughout the experiments. These results are the average of 10 experiments with different noise sets. This demonstrates that StrOUD can be trusted to diagnose outliers even if the purity of the original data is not certain.

5.2 Empirical Evaluation with Spatial-temporal Data

Experiments to detect spatial outliers were conducted on one-dimensional crime data that recorded a count of the number of crimes at a specific location, and then on two-dimensional earthquake data from the southwestern United States that contained the intensity and depth of an earthquake at a specific location. Experiments to detect spatial-temporal outliers were conducted on five-dimensional buoy data recorded from approximately forty buoys located in the Gulf of Mexico during 2005 which was the year when many severe hurricanes, such as Katrina, moved through the Gulf of Mexico. The buoys were located at a specific geographic

Table 5.18: Shuttle data Results Using a Contaminated Typical Dataset

Noise	17 (10%)	34 (20%)	51 (30%)
TP	100%	100%	100%
FP	3.2%	3.2%	3.2%

Table 5.19: Detecting Outliers Using a Cleaned Dataset

Noise	50 (1.16%)			100 (2.29%)	
k	10	20	100	40	100
TP	99.8%	99.8%	99.8%	99.8%	99.6%
FP	2.7%	2.8%	2.8%	2.7%	2.9%

location in the Gulf of Mexico and record wind direction, wind speed, barometric pressure, air temperature, and water temperature on an hourly basis.

5.2.1 Crime Data

This data is a count of crimes that occurred in a geographic area. The data has one non-spatial attribute which is the count of the number of crimes. The data also has a spatial location (x coordinate and y coordinate) for each recorded count. This data set contains 464 instances. The location of the crimes is shown in Figure 5.8 with the magnitude of the crime count at the location indicated by the size of the circle.

For the experiments with the crime data, the one non-spatial attribute and the geographic information are utilized. Each individual geographic location and its non-spatial attribute for that location is an individual test point, and the remaining geographic locations with their non-spatial attribute are the training data. The data is looped through, so that each geographic point and its non-spatial attribute will be a test point once.

Crime Data Experiments

For the crime data, there are 464 instances. The minimum crime count value was 1 and the maximum value was 10. For each instance, three experiments were conducted. For the first experiment the non-spatial attributes and the spatial coordinates are weighted so that the sum of the weights always equals one. The weight for the non-spatial attributes decreases from 1 to 0 (1, .95, .75, .5, .25, .05, 0) while the weight of the spatial coordinates increases from 0 to 1(0, .05, .25, .5, .75, .95, 1). For the second experiment, the non-spatial attributes always have a weight equal to 1, and an exponential kernel is used for the spatial

Figure 5.8: Location of Crimes

coordinates that is weighted and the weights vary from 1 to 0 (1, .95, .75, .5, .25, .05, 0). For the third experiment, an exponential kernel is used for the non-spatial attributes that is weighted and the weights vary from 1 to 0 (1, .95, .75, .5, .25, .05, 0) while an exponential kernel is also used for the spatial coordinates that is weighted and weights vary from 0 to 1 (0, .05, .25, .5, .75, .95, 1).

The crime data experiment results were compared with the results of using the Local Moran's module in ArcGIS. The neighborhood in the Local Moran's analysis is the number of points within a radius of the point to be tested with 4000 feet being used as the radius for this experiment. The number of points within a 4000 foot radius of the test point were the number of neighbors used in the Local Moran's calculations and in the StrOUD calculations with the spatial kernel.

Crime Data Results

The experimental results using an exponential kernel showed that an exponential kernel was not very effective in determining outliers. In almost all test cases, using the exponential kernel at all weights produced a Not Outlier result. The uniform distribution was more effective in calculating outliers.

Using the spatial coordinates and one non-spatial attribute (the count of the number of crimes), and varying the weights of the non-spatial attributes and spatial coordinates, but always making them equal to one, 6 of the 464 instances were determined to be outliers when a weight of .95 was given to the count and a weight of .05 was given to the spatial coordinates. The location of the 6 outliers determined by StrOUD is shown in Figure 5.9. Local Moran's determined that 7 of the 464 instances were outliers. The location of the 7 outliers determined by Local Moran's is shown in Figure 5.10. The point that Local Moran's determined to be an outlier and this method determined to not be an outlier had a crime count of 3 and is located in a very dense crime count area. For this point, there are 174 of the 464 points within its 4000ft radius. Its z value is -2.56, and for a point to be an outlier with Local Moran's, its z value must be less than -2.0, so this point is just below

Table 5.20: Crime Data Point Neighborhood Count Distribution

Count	Number of points
1	140
2	22
3	8
4	1
5	2
6	1

the threshold. Table 5.20 shows The distribution of the 174 neighbors. As such, 2.3% of the points in the neighborhood of 174 points have a count greater than 3, and 6.9% of the points in that neighborhood have a count greater than or equal to 3.

5.2.2 Earthquake Data from the Southwestern United States

This data is for earthquakes that occurred in the southwestern United States. The data has two non-spatial attributes which are the earthquake's magnitude and depth. The data also has a spatial location (x coordinate and y coordinate) for each recorded earthquake. This data set contains 557 earthquake instances.

For the experiments with the earthquake data, the non-spatial attributes and the geographic information are utilized. Each individual geographic location and its non-spatial attributes for that location is an individual test point, and the remaining geographic locations with their non-spatial attributes are the training data. The data is processed in a loop, so that each geographic point and its non-spatial attributes will be a test point once.

Earthquake Experiments

For the earthquake data, there are 557 instances. For each instance, three experiments were conducted. For the first experiment the non-spatial attributes and the spatial coordinates are weighted so that the sum of the weights always equals one. The weight for the non-spatial attributes decreases from 1 to 0 (1, .98, .95, .75, .5, .25, .05, .02, 0) while the weight of the spatial coordinates increases from 0 to 1 (0, .02, .05, .25, .5, .75, .95, .98, 1). For

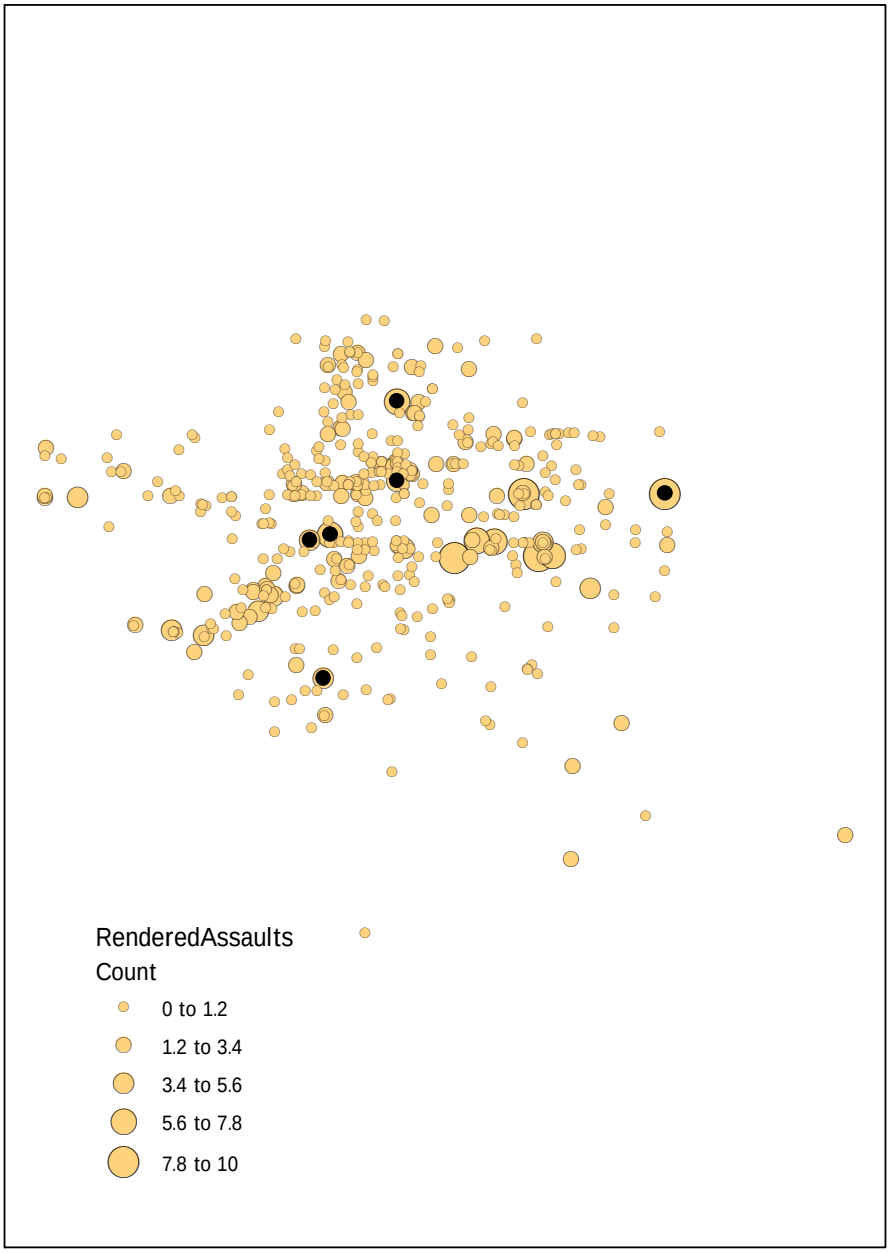

Figure 5.9: Location of Outliers using StrOUD

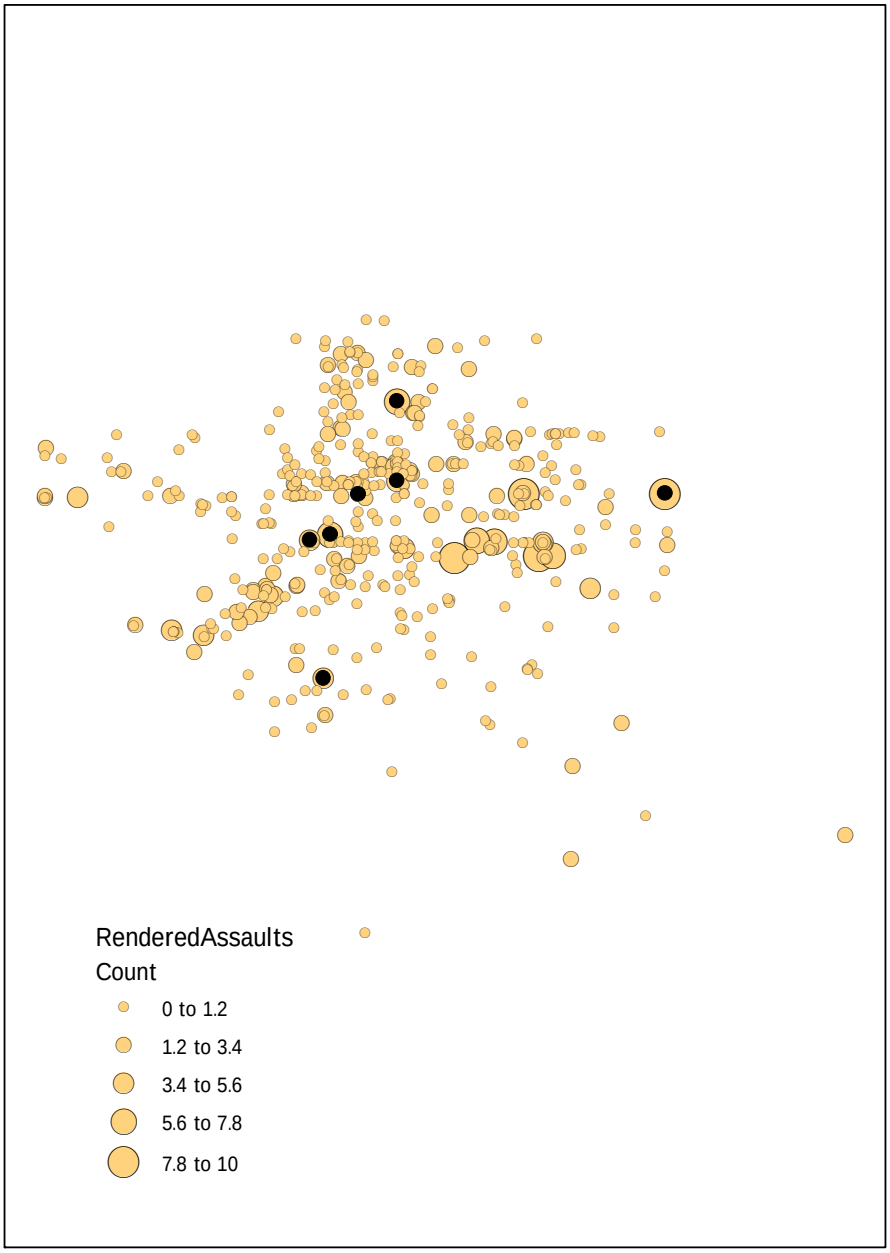

Figure 5.10: Locations of Outliers using Local Moran's

the second experiment, the non-spatial attributes always have a weight equal to 1, and an exponential kernel is used for the spatial coordinates that is weighted and the weights vary from 1 to 0 (1, .98, .95, .75, .5, .25, .05, .02, 0). For the third experiment, an exponential kernel is used for the non-spatial attributes that is weighted and the weights vary from 1 to 0 (1, .98, .95, .75, .5, .25, .05, .02, 0) while an exponential kernel is also used for the spatial coordinates that is weighted and weights vary from 0 to 1 (0, .02, .05, .25, .5, .75, .95, .98, 1).

Earthquake Results

For each test case of the 557 instances, the script completed in approximately 10.5 seconds. The experimental results using an exponential kernel showed that an exponential kernel was not very effective in determining outliers. In almost all test cases, using the exponential kernel at all weights, produced a Not Outlier result.

Using the spatial coordinates and two non-spatial attributes, and varying the weights of the non-spatial attributes and spatial coordinates, but always making their sum equal to one, StrOUD determined that data from 16 of the 557 sites were outliers. The results of the earthquake data experiments are shown in Figure 5.11. 12 of the 16 outliers were due primarily to the values of the non-spatial attributes and are shown in red in the figure. 2 of the 16 outliers were due primarily to their spatial location and are shown in purple in the figure. 1 of the 16 outliers was always an outlier regardless of the weights and is shown in orange in the figure. 1 of the 16 outliers needed contributions from both the non-spatial attributes and the spatial coordinates to be an outlier and is shown in blue in the figure. This point, shown in blue, was not determined to be an outlier when the weight of the non-spatial attributes or the spatial coordinates was equal to 1, but was determined to be an outlier when the weight of the non-spatial attributes varied between .95 and .25 and the weight of the spatial coordinates varied between .25 and .95. The remaining 541 sites were determined to not be outliers and are shown in green in the figure.

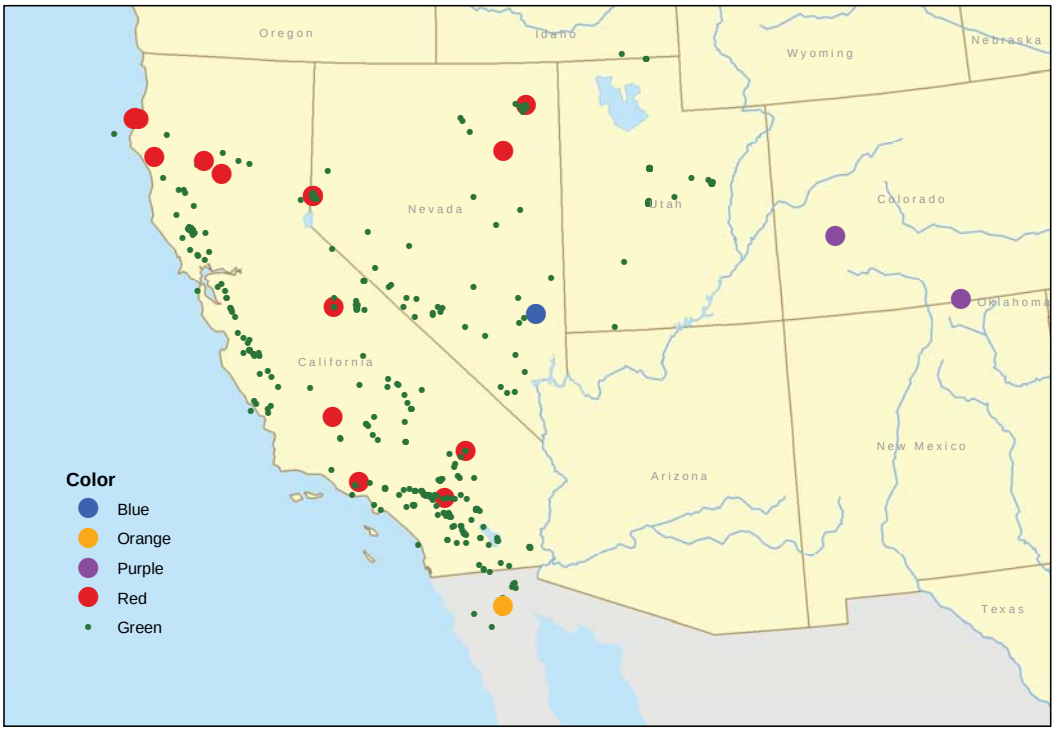

Figure 5.11: Earthquake Data Results

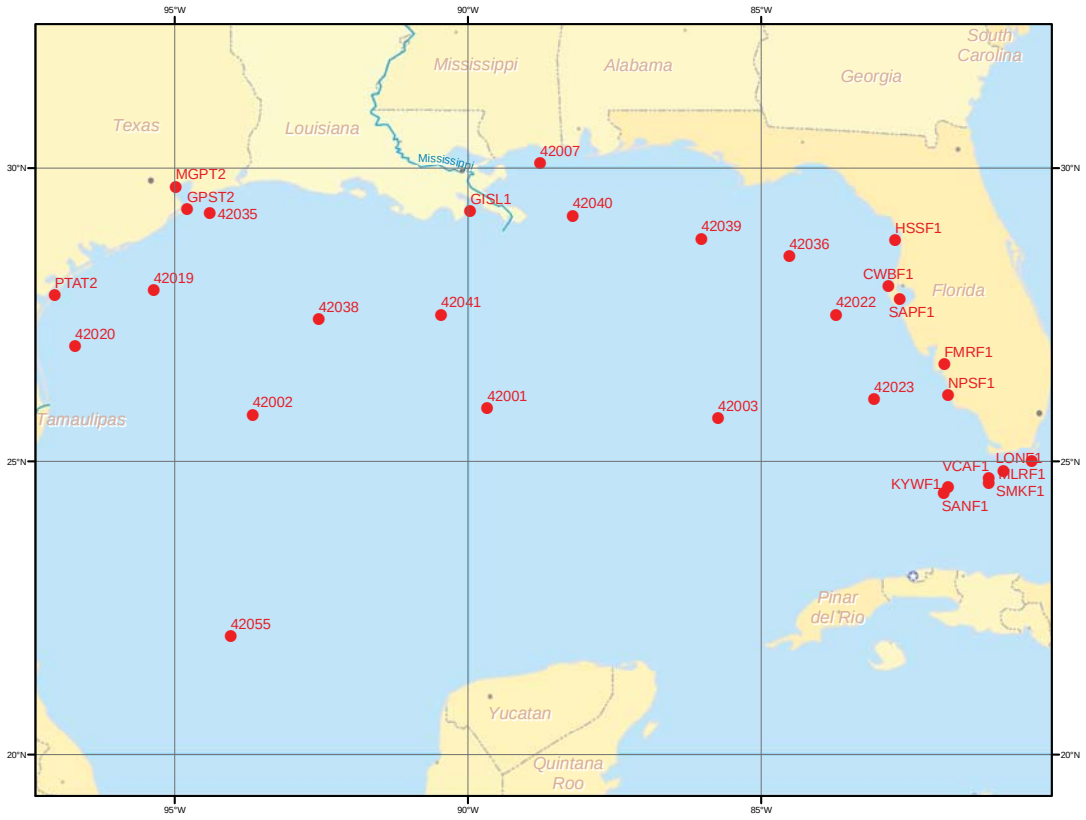

Figure 5.12: Locations of Buoys in the Gulf of Mexico

5.2.3 Buoy Data from the Gulf of Mexico

This data is weather data recorded from 30 buoys located in the Gulf of Mexico during 2005. The data has five features which are wind direction, wind speed, barometric pressure, air temperature, and water temperature. Each buoy has a spatial location (x coordinate and y coordinate). This data set contains 163849 instances. Each instance lists the time (year, month, day, and hour) the feature data was recorded at the spatial location. The location of the buoys in the Gulf of Mexico is shown in Figure 5.12.

For the experiments with the buoy data, the vector of features, the geographic information, and the temporal information are utilized. Each individual geographic location, the time the data was recorded, and its vector of features for that specific location and time is

an individual test point. The remaining geographic locations with their vector of features and the time the data was recorded are the training data. The data is processed in a loop, so that each point in the data set will be a test point once.

Buoy Experiments

The first experiment was to run the buoy data with the original StrOUD script using only the vector of features (non-spatial and non-temporal). The second experiments were to run the StrOUD script with the spatial kernel represented by a uniform distribution using the vector of features and spatial data. The third experiments were to run the StrOUD script with the temporal kernel represented by a uniform distribuion using the vector of features and the temporal data. The fourth experiments were to run the StrOUD script with the spatial and temporal kernels using the vector of features, the spatial data, and the temporal data. All kernels were represented by uniform distributions with $\alpha_1(d_f) + \alpha_2(d_s) + \alpha_3(d_t)$ and $\alpha_1 + \alpha_2 + \alpha_3 = 1$ where d_f = feature distance, d_s = spatial distance, and d_t = temporal distance.

Buoy Results

For the first experiment, a weight of 1 was given to the vector of features and a weight of 0 was given to the spatial and temporal components, and 437 outliers were identified.

For the second set of experiments, the features and the spatial coordinates were weighted and the sum of their weights equaled 1, and a weight of 0 was given to the temporal component. For a weight of .95 for the features and a weight of .05 for the spatial coordinates, 373 outliers were identified, and all 373 outliers were in the group of 437 outliers found in the first experiment that did not include any spatial or temporal data. For a weight of .90 for the features and .10 for the spatial coordinates, 90 outliers were identified, and all 90 outliers were in the group of 373 outliers found when a weight .95 was given to the features and a weight of .05 was given to the spatial coordinates.

For the third set of experiments, the features and the temporal coordinates were weighted

and the sum of their weights equaled 1, and a weight of 0 was given to the spatial coordinates. For a weight of .95 for the features and a weight of .05 for the time, 401 outliers were identified, and 398 of those 401 outliers were in the group of 437 outliers found in the first experiment that did not include any spatial or temporal data. For a weight of .90 for the features and .10 for the temporal data, 110 outliers were identified, and all 110 outliers were in the group of 401 outliers found when a weight of .95 was given to the features and weight of .05 was given to time.

The fourth, and final, experiment was using the feature information, the spatial coordinates, and the temporal information all together. For this experiment, a weight of .90 was given to the features, a weight of .05 was given to the spatial coordinates, and a weight of .05 was given to time. For this combination of weights equal to 1, 123 outliers were identified. Of these 123 outliers, 120 of them were in the group of 437 outliers found in the first experiment that did not include any spatial or temporal data, 120 of them were in the group of 373 outliers found in the second experiment when a weight of .95 was given to the attributes and weight of .05 was given to the spatial coordinates, and all 123 outliers were in the group of 401 outliers found in the third experiment when a weight of .95 was given to the attributes and a weight of .05 was given to time.

Table 5.21 shows the results of the experiments with the buoy data. Except for three outliers found when a weight is given to the temporal kernel, all of the other outliers are in the set of 437 outliers resulting from the experiment when all the weight is given to the attributes and no weight is given to the spatial or temporal components. Also, all of the outliers resulting from a weight of .90 for the features and a total weight of .10 given to the spatial and temporal components are a subset of the outliers resulting from a weight of .95 for the features and a weight of .05 given to the spatial or temporal components.

Strong outliers and weak outliers are defined such that strong outliers have a p-value between 0 and 0.02 and weak outliers have a p-value between 0.02 and 0.05. Giving weight to the spatial and temporal kernels in addition to the vector of features kernel resulted in identifying only the strong outliers. The identified outliers resulted from a hurricane or a

Table 5.21: Gulf of Mexico Buoy Data Results

weights			Number of
attributes	spatial	time	Outliers
1.0	0	0	437
.95	.05	0	373
.90	.10	0	90
.95	0	.05	401
.90	0	.10	110
.90	.05	.05	123

strong cold front. Figure 5.13 shows the location of buoys producing weak outliers resulting from the location of Hurricane Katrina at 2am on August 29, 2005. Figure 5.14 shows the location of buoys producing both strong outliers and weak outliers resulting from the location of Hurricane Katrina at 7am on August 29, 2005. Figure 5.15 shows the location of buoys producing outliers resulting from a cold front that moved across Texas into the Gulf of Mexico at 12pm on December 8, 2005.

5.2.4 Kernel Density Estimation (KDE) experiments

These experiments utilized a one-dimensional KDE function from Matlab (9) with the goal of significantly decreasing the computational time required by StrOUD for identifying outliers in large spatial-temporal databases. A Gaussian kernel is assumed and the bandwidth is chosen automatically by the KDE function. Inputs for the KDE function are the data from which the density estimate is constructed, the number of mesh points used in the uniform discretization which must be a power of two, the minimum value of each data feature, and the maximum value of each data feature. The number of mesh points is equivalent to the number of bins. A kernel density estimate is computed for each feature in the vector of features, the x value of the spatial data, the y value of the spatial data, and for the time. The output of the KDE function is the optimal bandwidth, the density values, and the grid over which the density estimate is computed. The density values are used to compute the probability for each point, then p-values are computed from the probabilities, and outliers are determined based on the p-values and the confidence level.

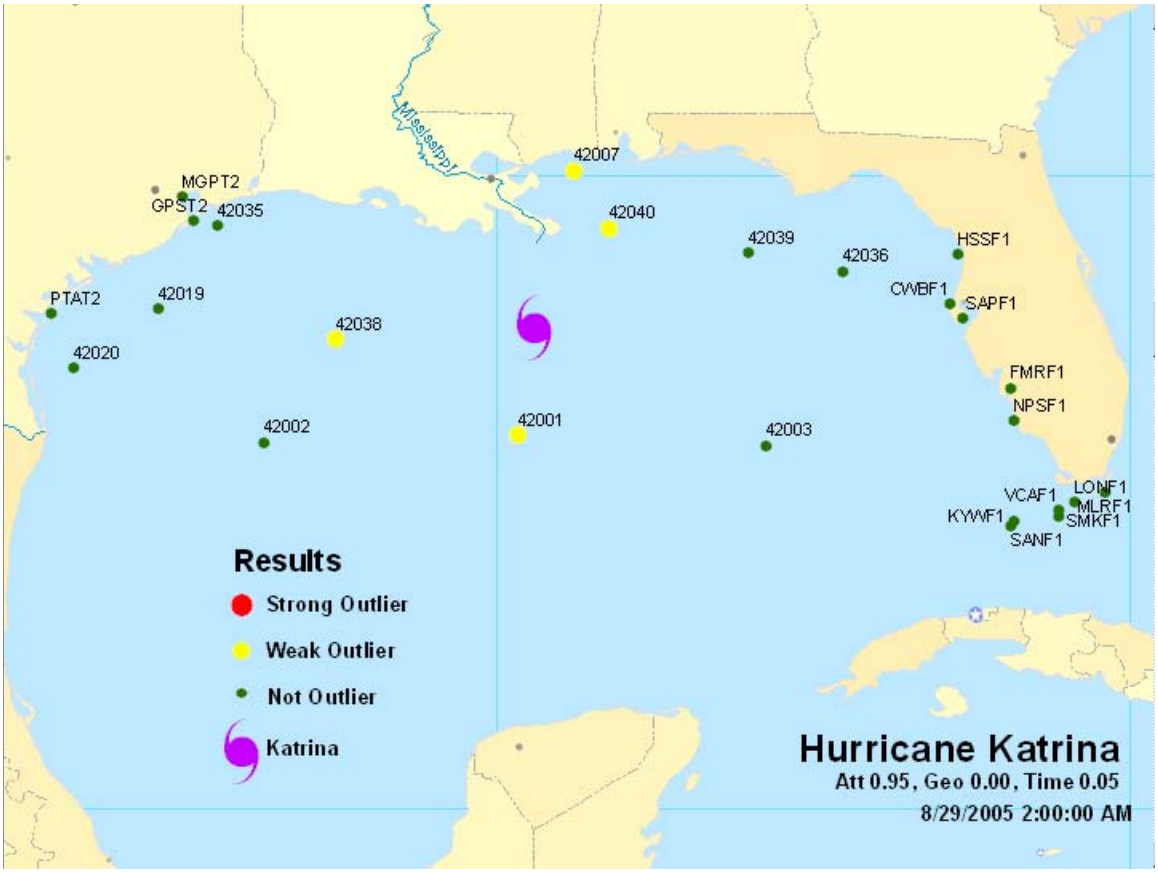

Figure 5.13: Locations of Buoys with Weak Outliers Due to Hurricane Katrina

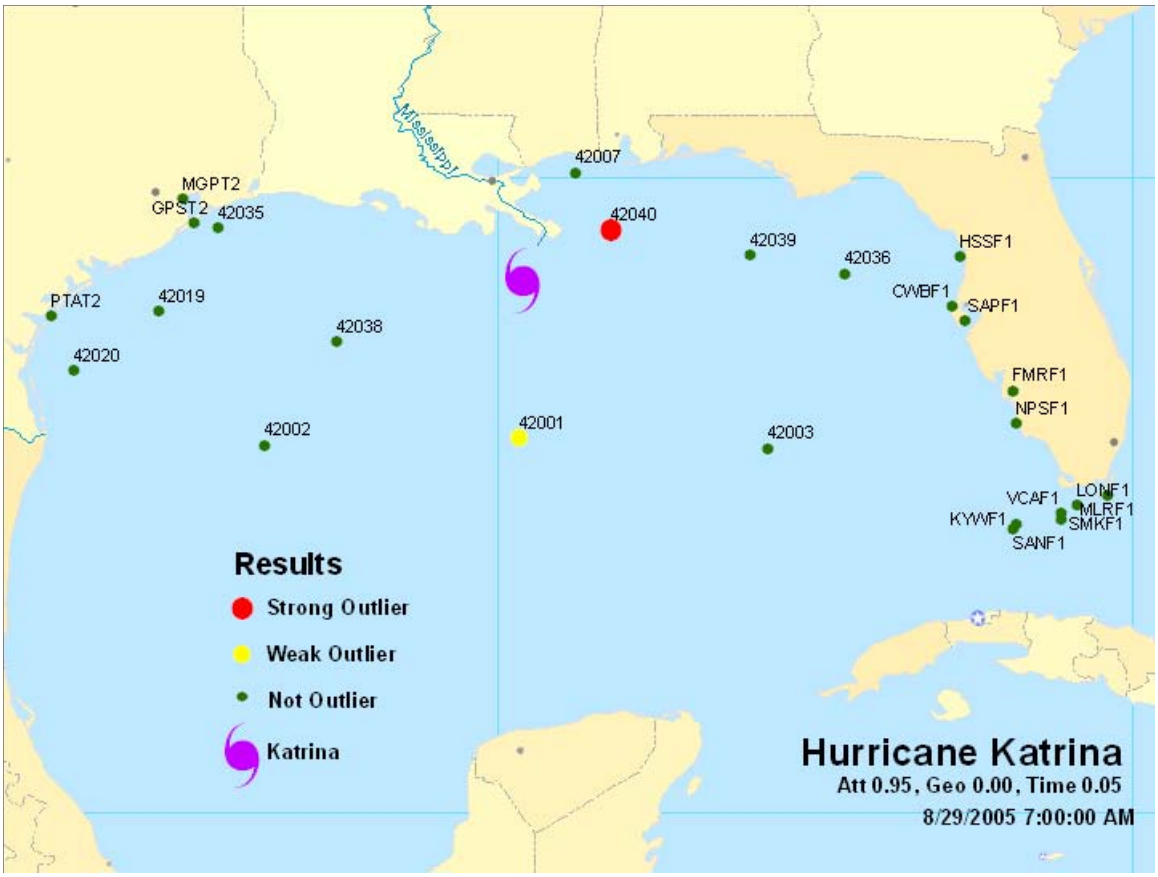

Figure 5.14: Location of Buoys with Strong and Weak Outliers Due to Hurricane Katrina

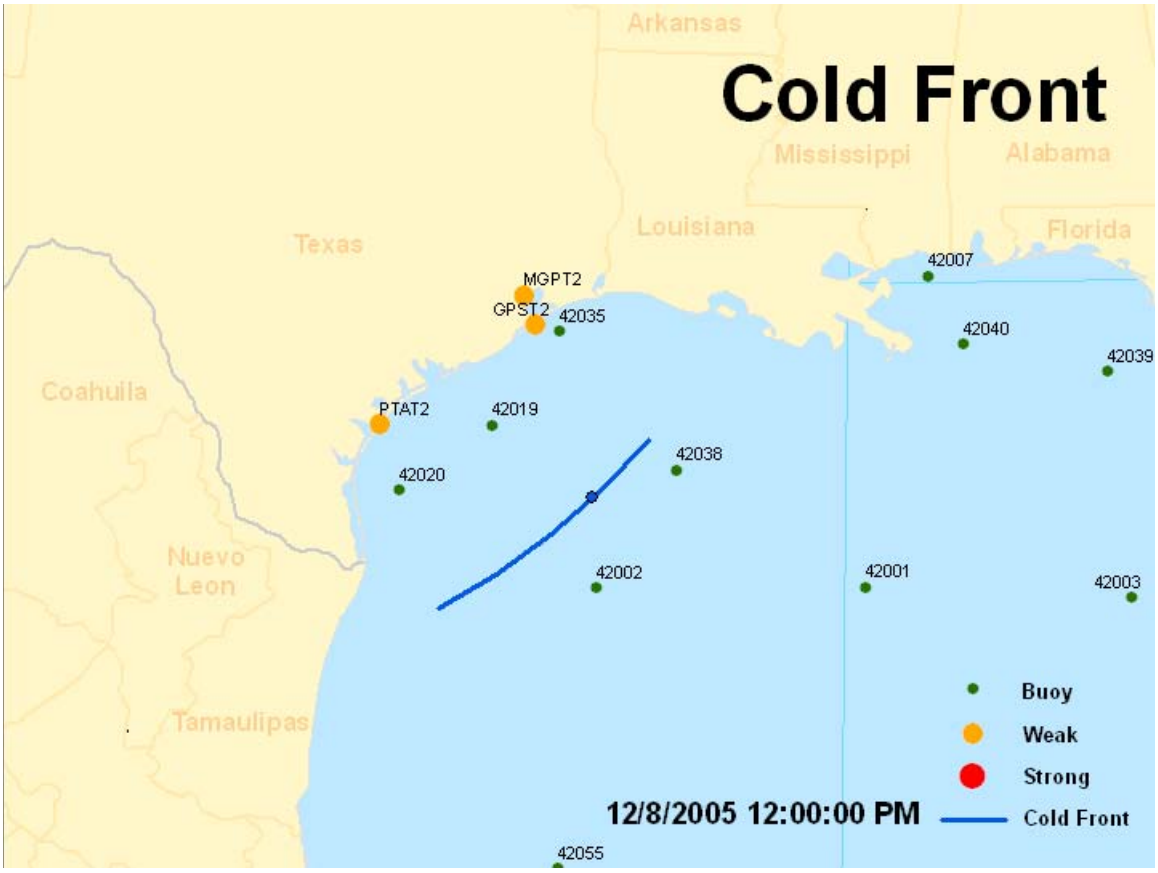

Figure 5.15: Location of Buoys with Outliers Resulting from a Cold Front

KDE Experiments

The first experiments with the Outlier KDE algorithm were on the earthquake data set (section 5.2.2), and the next experiments were on the buoy data set (section 5.2.3). For the Outlier KDE experiments with the earthquake data set, only feature (intensity and depth) and spatial (x and y coordinates) data were utilized. The feature data and the spatial coordinates are weighted so that the sum of the weights always equals one. The weight for the feature data decreases from 1 to 0 (1, .98, .95, .75, .5, .25, .05, .02, 0) while the weight of the spatial coordinates increases from 0 to 1 (0, .02, .05, .25, .5, .75, .95, .98, 1).

For the experiments with the buoy data the sample size was 10000, and the value of the number of mesh points was varied from 8 to 8192 by a power of 2. Four different combinations of weights were utilized for each value of the number of mesh points. The four combinations are feature weight = 1 and spatial and temporal weight = 0; feature weight = 0.9 and spatial and temporal weight each = 0.05; feature weight = 0.8 and spatial and temporal weight each =0.1; and feature weight = 0.2 and spatial and temporal weight each = 0.4. Experiments were conducted using all the data as the sample with a feature of weight of 0.9 and spatial and temporal weight each = 0.05, in order to compare the number of outliers determined using all the data with the number of outliers determined using a sample size of 10000 which is about 6 % of the entire data. Experiments were conducted using a sample size of 10000 with a feature weight of 0.9 and a spatial and temporal weight each = 0.05 while varying the number of mesh points from 8 to 8192 by a power of 2 in order to compare the number of outliers and execution time with kernel-based StrOUD with feature weight = 0.9 and and spatial and temporal weight each = 0.05. The final experiment with the buoy data was taking the outliers determined by the Outlier KDE algorithm and using this set as the test data set for kernel-based StrOUD.

KDE Results

For the experiments with the earthquake data, the Outlier KDE algorithm identified more outliers than StrOUD for each weight. The results are shown in Table 5.22. The table

shows the weights given to the features and the spatial coordinates, the number of outliers identified by the Outliers KDE algorithm, the number of outliers identified by the Outliers KDE algorithm that were also identified by the StrOUD algorithm, and the number of outliers identified by the Outliers KDE algorithm that were not identified by the StrOUD algorithm. For all weights, all the outliers identified by the StrOUD algorithm were included in the set of outliers determined by the Outliers KDE algorithm. In all cases, the outliers from the Outliers KDE algorithm are a superset of the outliers from the StrOUD algorithm. In all experiments with the earthquake data, the intersection of the set of outliers from the Outliers KDE algorithm and the set of outliers from the StrOUD algorithm is the entire set of outliers from the StrOUD algorithm.

For the experiments with the buoy data, the Outlier KDE algorithm identified more outliers than kernel-based StrOUD for every combination of weights and for every value of the number of mesh points, but in all cases, the execution time for the Outlier KDE algorithm was much faster than the execution time for kernel-based StrOUD. The execution time for the Outlier KDE algorithm is about 2% of the execution time for the kernel-based StrOUD algorithm. The results of the Outlier KDE algorithm experiments are shown in Table 5.23 for a sample size of 10000, where n is number of mesh points/bins, kde time is the time to run the one-dimensional Matlab kde function, Outlier KDE time is the total execution time for the Outlier KDE algorithm to include the execution time for the one-dimensional Matlab kde function, and kernel-based StrOUD time is the execution time for the kernel-based StrOUD algorirhm. For a small n, the one-dimensional KDE function from Matlab underfits the data, and for a large n, the one-dimensional KDE function from Matlab overfits the data.

A comparison of the number of outliers and execution time between using a sample size of 10000 and using all the buoy data is shown in Table 5.24 where n is varied from 8 to 8192 by a power of 2 and the weights stay constant (feature weight = 0.9 and spatial and temporal weight are each = 0.05). Using all the data, the number of outliers determined by Outlier KDE stayed relatively constant as n increased at approximately 13000 outliers,

Table 5.22: Earthquake Data Results Using the Outlier KDE Algorithm

feature weight	spatial weight	Outliers	Outliers in StrOUD	Outliers not in StrOUD
1	0	25	21	4
0.98	0.02	28	23	5
0.95	0.05	28	24	4
0.75	0.25	28	24	4
0.5	0.5	28	25	3
0.25	0.75	26	24	2
0.05	0.95	28	23	5
0.02	0.98	28	23	5
0	1	16	13	3

while the number of outliers for a sample size of 10000 increased from approximately 13000 to approximately 16000 as n increased from 8 to 8192 bins.

The results of the experiment where the outliers from the Outliers KDE algorithm are used as test data for the kernel-based StrOUD algorithm are shown in Table 5.25. All the outliers identified by the kernel-based StrOUD algorithm were included in the set of outliers determined by the Outliers KDE algorithm. In all cases, the outliers from the Outliers KDE algorithm are a superset of the outliers from the kernel-based StrOUD algorithm. In all experiments with the buoy data, the intersection of the set of outliers from the Outliers KDE algorithm and the set of outliers from the kernel-based StrOUD algorithm is the entire set of outliers from the kernel-based StrOUD algorithm. When the output of the Outlier KDE algorithm was used as input for kernel-based StrOUD algorithm, the exact same outliers were detected as when all the buoy data was used, but the execution time was approximately 20 to 30 times faster than using all the data. The total execution time using the outliers from the Outliers KDE algorithm as test data for the kernel-based StrOUD algorithm was less than one hour while the execution time for the kernel-based StrOUD algorithm using all the data was more than one day.

Table 5.23: Buoy Data Results Using the Outlier KDE Algorithm

n	feature weight	spatial weight	time weight	Outlier KDE outliers	kde time(sec)	Outlier KDE time(sec)	StrOUD outliers	StrOUD time(sec)
8	1	0	0	12850	8.2	1295.5	123	93851.09
8	0.9	0.05	0.05	13099	8.3	1253.84	123	93851.09
8	0.8	0.1	0.1	13229	8.3	1264.0	123	93851.09
8	0.2	0.4	0.4	11146	8.2	1245.33	123	93851.09
16	1	0	0	12941	8.4	1218.74	123	93851.09
16	0.9	0.05	0.05	13270	8.4	1222.69	123	93851.09
16	0.8	0.1	0.1	12960	8.5	1248.27	123	93851.09
16	0.2	0.4	0.4	11919	8.4	1241.1	123	93851.09
32	1	0	0	14222	8.7	1217.92	123	93851.09
32	0.9	0.05	0.05	13381	8.6	1234.1	123	93851.09
32	0.8	0.1	0.1	13511	8.6	1220.17	123	93851.09
32	0.2	0.4	0.4	13826	8.6	1278.25	123	93851.09
64	1	0	0	13129	8.8	1240.76	123	93851.09
64	0.9	0.05	0.05	13527	8.9	1248.13	123	93851.09
64	0.8	0.1	0.1	13485	8.8	1241.05	123	93851.09
64	0.2	0.4	0.4	14331	8.8	1292.18	123	93851.09
128	1	0	0	13849	8.9	1253.21	123	93851.09
128	0.9	0.05	0.05	13741	9.1	1292.98	123	93851.09
128	0.8	0.1	0.1	13577	9.0	1270.16	123	93851.09
128	0.2	0.4	0.4	15271	9.0	1325.72	123	93851.09
256	1	0	0	14249	9.2	1301.27	123	93851.09
256	0.9	0.05	0.05	14108	9.2	1312.73	123	93851.09
256	0.8	0.1	0.1	14047	9.2	1289.11	123	93851.09
256	0.2	0.4	0.4	15589	9.2	1352.81	123	93851.09
512	1	0	0	14650	9.5	1392.14	123	93851.09
512	0.9	0.05	0.05	14449	9.4	1388.74	123	93851.09
512	0.8	0.1	0.1	14413	9.6	1401.0	123	93851.09
512	0.2	0.4	0.4	16267	9.6	1412.89	123	93851.09
1024	1	0	0	14907	9.7	1557.49	123	93851.09
1024	0.9	0.05	0.05	14732	9.7	1544.02	123	93851.09
1024	0.8	0.1	0.1	15104	9.7	1539.52	123	93851.09
1024	0.2	0.4	0.4	18186	9.7	1556.8	123	93851.09
2048	1	0	0	16241	9.8	1896.41	123	93851.09
2048	0.9	0.05	0.05	16144	9.8	1828.33	123	93851.09
2048	0.8	0.1	0.1	15923	9.9	1914.59	123	93851.09
2048	0.2	0.4	0.4	17909	9.8	1848.04	123	93851.09
4096	1	0	0	15854	10.1	2441.11	123	93851.09
4096	0.9	0.05	0.05	16287	10.2	2431.45	123	93851.09
4096	0.8	0.1	0.1	16422	10.0	2538.24	123	93851.09
4096	0.2	0.4	0.4	18493	10.2	2465.6	123	93851.09
8192	1	0	0	14908	10.7	3731.93	123	93851.09
8192	0.9	0.05	0.05	16632	10.7	3661.57	123	93851.09
8192	0.8	0.1	0.1	15530	10.5	3656.27	123	93851.09
8192	0.2	0.4	0.4	18422	10.6	3651.35	123	93851.09

Table 5.24: Comparison of Results Using a Sample and all the Buoy Data

n	sample size	Outlier KDE outliers	kde time(sec)	Outlier KDE time(sec)
8	10000	13099	8.3	1253.84
8	163848	13189	12.1	1521.08
16	10000	13270	8.4	1222.69
16	163848	12763	12.2	1586.72
32	10000	13381	8.6	1234.1
32	163848	13297	12.6	1595.12
64	10000	13527	8.9	1248.13
64	163848	13446	12.8	1581.77
128	10000	13741	9.1	1292.98
128	163848	13235	12.9	1601.48
256	10000	14108	9.2	1312.73
256	163848	12877	13.3	1610.79
512	10000	14449	9.4	1388.74
512	163848	12932	13.4	1642.83
1024	10000	14732	9.7	1544.02
1024	163848	12939	13.7	1655.42
2048	10000	16144	9.8	1828.33
2048	163848	13029	14.0	1740.11
4096	10000	16287	10.2	2431.45
4096	163848	12997	14.2	1872.95
8192	10000	16632	10.7	3661.57
8192	163848	13004	14.4	2152.13

Table 5.25: Results Using the OutlierKDE Output as the kernel-based StrOUD Test Data

n	KDE time(sec)	Outlier KDE time(sec)	Outlier KDE outliers	StrOUD time(sec) all data	StrOUD time(sec) using Output KDE as test data	StrOUD outliers
8	8.3	1253.84	13099	93851.09	1427.62	123
16	8.4	1222.69	13270	93851.09	1431.91	123
32	8.6	1234.1	13381	93851.09	1440.09	123
64	8.9	1248.13	13527	93851.09	1442.11	123
128	9.1	1292.98	13741	93851.09	1447.79	123
256	9.2	1312.73	14108	93851.09	1459.33	123
512	9.4	1388.74	14449	93851.09	1466.56	123
1024	9.7	1544.02	14732	93851.09	1473.26	123
2048	9.8	1828.33	16144	93851.09	1514.45	123
4096	10.2	2431.45	16287	93851.09	1520.06	123
8192	10.7	3661.57	16632	93851.09	1552.51	123

Chapter 6: Conclusions

6.1 Summary

Outlier detection and identification is an important data mining task used by business, medical, security, and defense organizations. For the defense organizations, outlier detection has become more critical as asymmetic warfare has become more prevalent and terrorist activities become more common. For the defense community, identifying outliers, both spatially and temporally, is important in trying to predict and prevent attacks and maneuvers.

As a member of the DoD workforce working in an organization with a geospatial intelligence mission, I decided to focus my research on a methodology for identifying spatial-temporal outliers. I believe spatial-temporal outlier detection is a priority requirement for the defense community and has important military applications. The amount of spatial-temporal data being received, processed, stored, and analyzed has significantly increased, and this vast volume of spatial-temporal data makes it extemely difficult, if not impossible, for a human to analyze and identify outliers. The goal for my research was to devise a statistical methodology for identifying spatial-temporal outliers.

Some research has been conducted on identifying outliers with data that has a vector of attributes with no spatial or temporal information. Much of this previous outlier research assumed a probability distribution for the data that was usually a Gaussian distribution, and computed the likelihood of a point fitting the distribution or required a percentage of the data to be declared as outliers. My research methodology is an improvement over previous methods, because my methodology does not require an assumption on the data distribution and does not require a certain percentage of the data to be declared as outliers. To date, a lesser amount of research has been conducted on spatial-temporal outliers, and I believe that my research addresses this important and emerging research area with a robust

methodology that requires no assumptions about the data and does not require thresholds. The only input required for my methodology, is the number of neighbors and the confidence level, and the confidence level is the only parameter of significance.

My research was conducted incrementally starting with finding outliers in low dimensional numeric data with no spatial-temporal information included, and continued with finding outliers in high dimensional spatial-temporal data. My research used transduction to estimate membership in a class and attach a confidence to that estimate, devised a strangeness measure, calculated p-values, and those having the smallest p-values declared outliers based on the confidence level. This research methodology is the foundation of an algorithm called Strangeness based Outlier Detection (StrOUD). Spatial and temporal kernels were added to StrOUD to incorporate the impact of spatial and temporal information on outlier detection. The strangeness between two data objects was computed as a weighted combination of the vector of features, geographical, and temporal distance between the objects. Using these three kernels, StrOUD successfully diagnosed outliers with respect to their local spatial-temporal neighborhoods. Weighting the kernels permits a user to determine which kernel will have the greatest importance in the outlier calculation.

Many empirical evaluations were conducted with my methodology and consistently produced results with a high percentage of true positives and a low percentage of false positives. Using extensive experimentation, the results show that my methodology is robust with the respect to the choice of neighbors, k, and obtains good results for the usual choice of 95% confidence level. Experiments were conducted using clustering information and not using clustering information (assuming all the data was in one cluster), and the results were very comparable. The results with the clustering information are better, but not significantly better than the results without clustering information. Discarding the clustering information generally increases the false positive percentage. Comparisons were done with other methods, such as DB and LOF, and StrOUD was never out peformed by these methods. The extensive experimentation provided empirical evidence about the capabilities of my outlier detection methodology. The empirical evaluations demonstrated the performance of

StrOUD, kernel-based StrOUD, and Outlier KDE, and validated the claims made in this document. Since most data sets are not completely clean and contain noise, experiments were conducted with StrOUD on noisy/contaminated data. StrOUD was able to effectively clean a data set with outliers to provide a sample of data that is mostly free of outliers. If the data set is contaminated with outliers, StrOUD can effectively identify the outliers in the data.

For large data sets, the computation time required to run StrOUD can be lengthy. To try and reduce the computation time, random sampling, support vector machine with random sampling, and kernel density estimation were utilized. Each of these alternatives reduced computation time while obtaining similar results to using all the data.

The main contribution of my research is the creation of a novel, principled, statistically-based test, that does not make assumptions about the data distribution, to determine outliers with data that has multiple attributes, spatial coordinates, and temporal information, that is based on transduction to compute strangeness and utilizes kernels for the vector of attributes, spatial coordinates, and temporal information and permits the user to input the weights for each of these kernels.

6.2 Future Work

In my research, the distance metric used was Euclidean distance. Additional distance measures should be investigated to see the impact on outlier results and computation time. To speed the search for k-nearest neighbors, multi-dimensional search structures, such as Kd-trees (7) could be examined. Another options to avoid the costly k-nearest neighbor computations for each point is to employ the space-filling curves technique of Angiulli (1), based on maintaining lists of the points ordered by their position on a series of Hilbert-curves, each shifted or rotated, as proposed by Liao (36).

My research methodology identified outliers from data recorded at a spatial location. The next step beyond finding these outliers from point data is to find outliers for trajectories. Trajectories are spatial paths generated by connecting points over time. The

trajectories could be modeled using Hidden Markov Models, a similarity matrix would be generated, distances determined from the similarity matrix, and then, using these distances, the StrOUD methodology could be utilized to determine the outliers.

Another area of future work would be to identify area outliers. Some research has been conducted on identifying outliers for points, but research is needed for determining area outliers. An area is an enclosed geographic region that is larger than a point. Much spatial-temporal information is recorded as areas in Geographic Information Systems, so it would be a logical step to devise a methodology for identifying outliers for areas.

As the applications may differ in their objectives, it may me worthwhile to try and incorporate specific application information into my methodology. Incorporating specific application information could be accomplished by creating a kernel function for each specific application and then replacing the existing kernels with the application specific kernels.

Since the applications and datasets tested with my application will differ, there is a need to incorporate domain specific knowledge into my methodology. Mechanisms for incorporating domain specific knowledge into my methodology will be investigated, created, and tested to determine the impact and possible improvement to my methodology.

The usage of cameras, sensors, mobile devices (e.g., cell phones) and satellites are generating spatial-temporal data streams. These spatial-temporal data streams create a dynamic environment for detecting outliers. An outlier at one or more moments of time may not be an outlier at a later time. One of the benefits of my methodology is that the introduction of a new test point from a data stream does not require the re-computation of all the previous strangeness values, but only to examine where the strangeness of the new test point fits in the distribution of existing strangeness values. The dynamic evolution of outliers from spatial-temporal data streams will require further experimentation.

To date, the largest number of dimensions tested with my methodology has been 40 dimensions. The impact of dimensionality on computation time and performance needs to be examined. Synthetic data sets can be generated for high-level dimensions, and the impact on computation time and performance can be evaluated. The goal would be to determine

the number of dimensions that cause the true positive percentage to substantially decrease and the false positive percentage to substantially increase.

Future research should also include the impact of incorporating normal mixtures, adapative kernel mixtures, and multivariate kernel estimating techniques into my methodology to determine the impact of each on computation time and performance. Mixture models assign instances to clusters or classes probabilistically instead of deterministically. Methods using a mixture model assign different probability distributions to each cluster or class. The adaptive kernel mixtures technique provides consistency for a large class of kernel functions with low compuational complexity associated with finite mixture models. Normal mixtures are consistent only if the mixture density is correct an the number of terms is correct. Adaptive mixtures have a create rule to add terms as necessary. Once established, mixture models often reduce computation time for estimating probabilities. Multivariate kernel estimating techniques use a multivariate kernel function to estimate the probability density function for mulitple variables that can be utilized to estimate their probabilities. These techniques could replace the kernel functions currently incorporated into my methodology to determine if they can reduce computation time while maintaining very good results.

BIBLIOGRAPHY

Bibliography

[1] Angiulli, F. and Pizzuti, C. (2005) Outlier mining in large high-dimensional data sets. IEEE Transactions on Knowledge and Data Engineering, 17(2): 203-215.

[2] Anselin, L. (1995) Local indicators of spatial association - LISA. *Geographical Analysis*, **27(2)**, 93-115.

[3] Barbara, D., Domeniconi, C., Rogers, J. (2006) Detecting Outliers Using Transduction and Statistical Significance Testing. 12th SIGKDD International Conference on Knowledge Discovery and Data Mining, 55-64.

[4] Barbara, D., Domeniconi, C., Rogers, J. (2009) Detecting Spatio-Temporal Outliers with Kernels and Statistical Testing. 17th International Conference on Geoinformatics, 1-6.

[5] Bay, S.D., and Schwabacher, M. (2003) Mining distance-based outliers in near linear time with randomization and a simple pruning rule. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining,* 29-38.

[6] Ben-Hur, A. and Horn, D. and Siegelmann, H.T. and Vapnik, V. (2001) Support Vector Clustering. Journal of Machine Learning Research, 2(2): 125-137.

[7] Bentley, J.L. (1975) Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9):509-517.

[8] Birant, D. and Kut, A. (2006) Spatio-Temporal Outlier Detection in Large Databases. 28th International Conference on Information Technology Interfaces, 179-184.

[9] Botev, Z. (2007) Kernel density estimation script. MATLAB 7.4(R2007a).

[10] Breunig, M., Kriegel, H., Ng, R., Sander, J. (2000) LOF: Identifying Density-Based Local Outliers. *Proc. of the ACM SIGMOD Conference on Management of Data,* 427-438.

[11] Brodatz, P. (1966) Textures: A Photographic Album for Artists and Designers, Dover Publications,Inc.,New York.

[12] Chandola, V., Banerjee, A., Kumar, V. (2009) Anomaly Detection: A Survey. ACM Computing Surveys. Vol. 41, No. 3, Article 15, 1-58.

[13] Chatzis, S., Kosmopoulous, D., Varvarigou, T. (2009) Robust Sequential Data Modeling using an Outlier Tolerant Hidden Markov Model. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 31, no 9, pp 1657-1669.

[14] Duda, R., Hart, P., Stork, D. (2001) Pattern Classification. Wiley.

[15] Edgington, E. (1980) Randomization Tests. New York: Marcel Dekker.

[16] Elena project data. ftp://ftp.dice.ucl.ac.be/pub/neural-nets/ELENA/databases/

[17] Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. *Proc. of the 2nd Intl. Conference on Knowledge Discovery and Data Mining.* 226-231.

[18] Fan, H., Zaiane, O., Foss, A., and Wu, J. (2009) Resolution-based outlier factor: detecting the top-n most outlying data points in engineering data. Knowledge and Information Systems. 19:31-51.

[19] Gammerman, A., and Vovk, V. (2002) Prediction algorithms and confidence measures based on algorithmic randomness theory. Theoretical Computer Science. 287: 209-217.

[20] Ghoting, A., Parthasarathy, S., Otey, M. (2008) Fast Mining of Distance-based Outliers in High-dimensional Datasets, Data Mining and Knowledge Discovery, Volume 16, Number 3, pages 349-364.

[21] Guerin-Dugue, A., and Aviles-Cruz, C. (1993) High Order Statistics from Natural Textured Images, *ATHOS workshop on System Identification and High Order Statistics*. Sophia-Antipolis, France, September 1993.

[22] Guerin-Dugue, A. et al., (1995) Deliverable R3-B4-P - Task B4: Benchmarks, Technical report, Elena-NervesII "Enhanced Learning for Evolutive Neural Architecture", ESPRIT-Basic Research Project Number 6891, June 1995

[23] Guha, S., Rastogi, R., and Shim, K. (1998) CURE: an efficient clustering algorithm for large databases. *Proc. of ACM SIGMOD Conf. on Management of Data,* 73-84.

[24] Han, J., and Kamber, M. (2001) Data Mining Concepts and Techniques, Academic Press.

[25] Hardin, J., and Rocke, D.M. (2004) Outlier detection in the multiple cluster setting using the minimum covariance determinant estimator. Computational statistics and data analysis, Vol 44, pp. 625-638, 2004.

[26] Hawkins, D. (1980) Identification of Outliers. Chapman and Hall, London.

[27] He, Z. and Xu, X. and Deng, S. (2003) Discovering cluster-based local outliers. Pattern Recognition Letters 24(9): 1641-1650.

[28] Ho, S.S., and Wechsler, H. (2003) Transductive Confidence Machine for Active Learning, *Int. Joint Conf. on Neural Networks*, Portland, OR.

[29] Hubert, M., Rousseeuw, P.J, and Van Aelst, S. (2005) Multivariate Outlier Detection and Robustness. In *Handbook of Statistics*, Vol. 24, C.R. Rao, E. Wegman, J. Solka, editors. Elsevier, 2005.

[30] Jin, W., Tung, A., and Han, J. Mining (2001) Top-n local outliers in large databases. *Proc. of the Intl. Conference on Knowledge Discovery and Data Mining,* 293-298.

[31] Knorr, E., Ng, R. (1998) Algorithms for Mining Distance-Based Outliers in Large Datasets. *Proceedings of the 24th Intl. Conference on Very Large Databases,* 392-403.

[32] Knorr, E., Ng, R., and Tucakov, V., (2000) Distance-Based Outliers: Algorithms and Applications, *VLDB Journal*, 8(3-4):237-253.

[33] Kou, Yufeng, Lu, Chang-Tien, and Chen, Dechang, (2006) Spatial Weighted Outlier Detection. *2006 SIAM Conference on Data Mining,* 613-617.

[34] Lewis, B.V. (1994) Outliers in Statistical Data. John Wiley.

[35] Li, M., and Vitanyi, P. (1997) Introduction to Kolmogorov Complexity and its Applications. 2nd Edition, Springer Verlag.

[36] Liao, S., Lopez, M.A., and Leutenegger, S.T. (2001) High dimensional similarity search with space filling curves. Proc. International Conference on Data Engineering, 615-622.

[37] Mamoulis, N., Cao, H., Kollios, G., Hadjieleftheriou, M., Tao, Y., and Cheung, D. (2004) Mining, Indexing, and Querying Historical Spatiotemporal Data, 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 236-245.

[38] Markou, M. and Singh, S. (2003) Novelty Detection: A Review, Signal Processing.

[39] Menasce, D., Abraho, B., Barbará, D., Almeida, V., Ribeiro, F. (2002) Fractal Characterization of Web Workloads. *Proceedings of the "Web Engineering" Track of WWW2002, Honolulu, Hawaii, USA*, 7-11.

[40] Moore, A. Clustering with Gaussian Mixtures. www.cs.cmu.edu/ awm.

[41] Neville, J., Jensen, D., Friedland, L., and Hay, M. (2003) Learning Relational Probability Trees, *Proc. of the 9th ACM-SIGKDD Intl. Conference on Knowledge Discovery and Data Mining,* Washington, DC, 625 - 630.

[42] Ng, R., and Han, J. (1994) Efficient and effective clustering methods for spatial data mining. *Proceedings of the 20th Intl. Conference on Very Large Data Bases*, 144-155.

[43] Pokrajac, D. and Lazarevic, A. and Latecki, L.J. (2007) Incremental local outlier detection for data streams. *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining* 61-75.

[44] Proedru, K., Nouretdinov, I., Vovk, V., Gammerman, A. (2002) Transductive confidence machine for pattern recognition. *Proc. 13th European conference on Machine Learning.* 2430:381-390.

[45] Ramaswamy, S., Rastogi, R., and Shim, K. (2000) Efficient algorithms for mining outliers from large data sets. *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 427-438.

[46] Sagan, H. (1994) Space Filling Curves. Springer-Verlag.

[47] Scott, D. (2004) Outlier Detection and Clustering by Partial Mixture Modeling. COMPSTAT 2004 Symposium.

[48] Sheikholesami, G., Chatterjee, S., and Zhang, A. (1998) WaveCluster: a multi-resolution clustering approach for very large spatial databases. *Proc. of the 24th Intl. Conference on Very Large Databases.* 428-439.

[49] Shekhar, S., Lu, C.T., and Zhang, P. (2003) A Unified Approach to Spatial Outliers Detection, *Geoinformatica*, 7(2), June, 2003.

[50] Tang, J., Chen, Z., Fu, A., and Cheung, D. (2002) Enhancing Effectiveness of Outlier Detection for Low Density Patterns. *Proc. of PAKDD'02*, 535-548.

[51] UCI Machine Learning Repository. http://www.ics.uci.edu/ mlearn/MLRepository.html/

[52] Vapnik. V. (1998) Statistical Learning Theory, New York: Wiley.

[53] Vovk, V., Gammerman, A., and Saunders, C. (1999) Machine learning applications of algorithmic randomness. *Proceedings of the 16th Intl. Conference on Machine Learning.* 444-453.

[54] Weka: University of Waikato Data Mining Software. http://www.cs.waikato.ac.nz/ml/weka/

Curriculum Vitae

James P. Rogers

Education
B.S. Civil Engineering, North Carolina State University, 1983.
M.S. Civil Engineering, Computer-Aided Design, George Washington University, 1991.

Work Experience
2001 to Present - Associate Technical Director. Currently TEC's Basic Research coordinator, TEC's Small Business Innovative Research program coordinator, and TEC's Science and Engineering Apprentice Program coordinator. Also, currently the primary investigator for the Statistical Determination of Spatial Outliers and Modelling of Spatio-Temporal Co-occurrence Patterns basic research projects.
2000 to 2001 - Special Projects Officer for Future Combat Systems.
1997 to 2000 - Chief, Information Applications and Technologies Branch.
1995 to 1997 - Chief, Technical Plans and Programs Office.
1994 to 1995 - Program Manager, Imagery Exploitation System.
1992 to 1994 - Program Manager, Total Distribution Advanced Technology Demonstration.
1987 to 1992 - Project Engineer for the Digital Topographic Support System.
1983 to 1987 - Project Engineer for terrain visualization projects, a 3D stereoscopic video system, and a Digital Laser Platemaking System.

Licenses
Licensed Professional Engineer in Virginia since 1991.

Society Memberships
Chi Epsilon engineering honor society

Publications
"Detecting Outliers using Transduction and Statistical Testing", submitted to Journal of Intelligent Information Systems, August 2010. With D. Barbara and C. Domeniconi.
"Mixed-Drove Spatio-Temporal Co-occurrence Pattern Mining", IEEE Transactions on Knowledge and Data Engineering (TKDE), October 2008. With M. Celik, S. Shekhar, and J. Shine.
"Detecting Outliers Using Transduction and Statistical Testing", Proceedings of the 12th ACM=SIGKDD International Conference on Data Mining and Knowledge Discovery, August 2006, Philadelphia, PA, USA. With D. Barbara and C. Domeniconi.