Knowledge and Information Systems (2002) 4: 387–412

Knowledge and Information Systems

FindOut: Finding Outliers in Very Large Datasets

Dantong Yu*, Gholamhosein Sheikholeslami† and Aidong Zhang

Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, New York, USA

Abstract. Finding the rare instances or the outliers is important in many KDD (knowledge discovery and data-mining) applications, such as detecting credit card fraud or finding irregularities in gene expressions. Signal-processing techniques have been introduced to transform images for enhancement, filtering, restoration, analysis, and reconstruction. In this paper, we present a new method in which we apply signal-processing techniques to solve important problems in data mining. In particular, we introduce a novel deviation (or outlier) detection approach, termed *FindOut*, based on wavelet transform. The main idea in FindOut is to remove the clusters from the original data and then identify the outliers. Although previous research showed that such techniques may not be effective because of the nature of the clustering, FindOut can successfully identify outliers from large datasets. Experimental results on very large datasets are presented which show the efficiency and effectiveness of the proposed approach.

Keywords: Clustering; Data mining; Outliers; Wavelet

1. Introduction

Data mining is a process of extracting valid, previously unknown, and ultimately comprehensible information from large datasets and using it to make crucial decisions. Modern companies are awash in data on customers, clients, suppliers, and industry trends. But data is of little use without intelligence. Here, intelligence refers to combing through the data to notice patterns, devise rules, and make predictions about the future. Data-mining technology permits organizations to make the most effective use of the vast amounts of data that they have gathered. For example, in the banking industry, data mining can be used in modeling and

* Current address: Brookhaven National Laboratory, Upton, New York, USA.
† Current address: Cisco Systems, Inc., San Jose, California, USA.

Received 7 Sep 2000
Revised 2 Feb 2001
Accepted 31 May 2001

predicting credit fraud, bankruptcy, evaluating risk, performing trend analysis, analyzing profitability, and helping with direct marketing campaigns.

One of the very interesting problems arising recently in the data-mining research community is the deviation (or outlier) detection problem. Here outliers refer to those exceptions which deviate from the values anticipated based on a given (usually statistical) model or from some previously known expectation and norm (Sarawagi et al., 1998). The identification of outliers can lead to the discovery of truly unexpected knowledge in areas such as electronic commerce exceptions, bankruptcy, credit card fraud, and even the analysis of performance statistics of stock exchange (Knorr and Ng, 1998). Such knowledge can lead to detecting abnormal events that happened in the past or predicting potential trends in the future. Such potential trends may become new directions for future investment, marketing, and other purposes.

1.1. Related Work

Most of the previous research in outlier detection has been in the field of statistics (Hoaglin et al., 1983; Johnson, 1992; Barnett and Lewis, 1994). These methods usually make assumptions about data distribution, statistical distribution parameters, type or number of outliers. However, these parameters generally cannot be easily determined, which makes these methods difficult to apply. Ruts and Rousseeuw (1996) proposed a depth-based method to detect the outliers. The data points are organized in layers in the data space according to the value of the point depth.[1] Outliers are expected to be in the layers with smaller depth. Such methods do not have the distribution fitting problem. However, for multidimensional spaces they may not be applicable. Sarawagi et al. (1998) proposed a discovery-driven method in which the search for exceptions is guided by pre-computed indicators of exceptions at various levels of detail in a data cube. They consider a value in a cell of a data cube to be an exception if it is significantly different from an anticipated value based on a statistical model. Although this method can handle hierarchies and ordered dimensions, model selection is still a difficult issue.

Arning et al. (1996) introduced a method for outlier detection which relies on the observation that after seeing a series of similar data an element disturbing the series is considered an outlier. Their method requires a function that can yield the degree to which a data element causes the dissimilarity of the dataset to increase. It looks for the subset of data that leads to the greatest reduction in Kolmogorov complexity for the amount of data discarded (Arning et al., 1996).

Knorr and Ng (1997, 1998) presented the algorithms to detect distance-based outliers. They consider a data point O in a dataset T, a $DB(p, D)$-outlier, if at least a fraction p of the data points in T lies greater than distance D from O. Their index-based algorithm executes a range search with radius D for each data point. If the number of data points in its D-neighborhood exceeds a threshold, the search stops and that data point is declared as a non-outlier, otherwise it is an outlier. In their cell-based approach, they quantize the complete data space

[1] The depth of a point $\mathbf{p}$ to a one-dimensional dataset $X = x_1, x_2, \ldots, x_n$ is the minimum of the number of data points to the left of $\mathbf{p}$ and the number of data points to the right of $\mathbf{p}$. The depth of a d-dimensional data point $\mathbf{p} = \langle p_1, p_2, \ldots, p_d \rangle$ is defined as the smallest depth of p_i in the i-dimensional projection of the dataset, where $1 \leqslant i \leqslant d$.

and assign the data points to the cells. By pruning away a large number of *red* cells which contain too many data points and their immediate neighbors, their approach avoids testing unnecessary cells and speeds up outlier detection. Their experiments show that their cell-based algorithm is the most efficient when the number of dimensions is less than or equal to 4. However, for higher number of dimensions ($\geqslant 5$), the number of cells grows exponentially, and the nested loop which they provided in the same paper outperforms the cell-based algorithm. In addition to the identification aspect for outlier detection, the same authors (Knorr and Ng, 1999) attempted to provide intensional knowledge to explain why an identified outlier is exceptional.

In Ramaswamy et al. (2000), another distance-based outlier detection algorithm was proposed: the top n points with the maximum D^k are considered outliers, where $D^k(\mathbf{p})$ denotes the distance of the k-th nearest neighbor of $\mathbf{p}$. They used a cluster algorithm to partition a dataset into several groups. Pruning and batch processing on these groups could improve efficiency for outlier detection.

Recently, Breunig et al. (2000) proposed the notion of LOF (local outlier factor) to indicate the degree of outlier-ness for each data point in a dataset. A strong outlier is the one which has high LOF value. Their basic idea is to compare the local density $\rho_{\mathbf{p}}$ of each data point $\mathbf{p}$ in a dataset with the local density $\rho_{\mathbf{p}'}$ of each data point $\mathbf{p}'$ from $\mathbf{p}$'s k-nearest neighbor set $NN_k(\mathbf{p})$ in the same dataset. If $\mathbf{p}$'s local density is lower, while its k-nearest neighbors' local densities are higher, then the LOF for $\mathbf{p}$ is higher (i.e., $\mathbf{p}$ is more likely to be an outlier). Their definition of LOF has some benefits: (1) for the data points in a cluster, the LOF value approaches 1; for other data points, the lower and upper bounds of their LOF values can be easily estimated; (2) by intelligently choosing the range of k, the LOF approach has lower computational complexity than the depth-based approaches for large dimensionality; (3) their notion generalizes many ideas from the distribution-based outlier detection algorithms; and (4) it can detect data points that are outliers relative to the densities of their local neighborhoods. These kinds of outliers cannot be detected by previous approaches.

We look at outlier detection from a signal-processing point of view and apply signal-processing techniques to address this problem.

1.2. Our Contributions

In this paper, we present an approach, named *FindOut* (which stands for *Find Outliers*), to efficiently detect outliers from large datasets. The main idea in FindOut is to remove the clusters from the original data and thus identify the outliers. Using the multi-resolution property of wavelet transforms, FindOut can successfully identify various percentages of outliers from large datasets. It can also detect the outliers for complicated data patterns with various densities. Furthermore, FindOut can handle high-dimensional datasets, which have not been addressed by most existing approaches. For instance, WaveCluster (Sheikholeslami et al., 1998) requires modifications to efficiently handle high-dimensional datasets, because it is affected by *curse of dimensionality*. Using a hash table data structure to represent the dataset, FindOut makes intelligent use of available resources to efficiently perform wavelet transform on high-dimensional datasets. This hash-based data representation technique can be applied for any grid-based data-processing approach. We demonstrate that the cost of wavelet transform can be reduced significantly, yet maintaining all the advantages of the wavelet process.

The rest of the paper is organized as follows. Section 2 defines the concept of outliers, introduces wavelet transforms, presents the outliers detection approach, and extends this approach into high dimensions. Section 3 presents the experimental results. Section 4 offers a conclusion.

2. Finding Outliers

The primary motivation for applying signal-processing techniques to spatial datasets comes from the observation that the multidimensional data points can be represented as a d-dimensional signal, and we can apply the following processing primitive on it.

2.1. Linear Space System

From the original dataset, information for detecting outliers is usually hidden and scattered. We introduce the *linear space system* to expose and gather such information. Given an input dataset, we summarize it as a distribution function and input it to the linear space system. We then get a new function which is called a *transformed distribution function*. If the transformed distribution function can make the outliers more salient than other data points, we can then easily detect outliers within the dataset.

Many signal-processing operations can be modeled as a linear space system (Jain et al., 1995). For a d-dimensional space, a linear space (LS_g) system can be completely described by its *kernel* function $g(x_1, x_2, \ldots, x_d)$. The kernel function acts like a processor on the input signal. For such a system, the output $\hat{f}_g(x_1, x_2, \ldots, x_d)$ is defined as the *convolution* of the input signal $f(x_1, x_2, \ldots, x_d)$ with the kernel function $g(x_1, x_2, \ldots, x_d)$. When the kernel is a wavelet function w, we call the linear system the *wavelet transform system*. We denote the output as $\hat{\phi}_w$ when the input is a signal described by function ϕ.

2.2. Problem Formalization

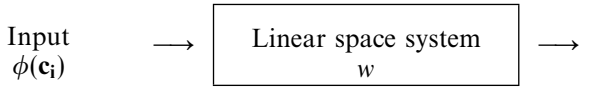
Let $A = \{A_1, A_2, \ldots, A_d\}$ be a set of bounded, totally ordered domains and $S = A_1 \times A_2 \times \ldots \times A_d$ be a d-dimensional numerical space or data space. $A_1, \ldots, A_d$ are referred as dimensions of S. The input dataset is a set of d-dimensional points $\mathbf{O} = \{\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_N}\}$, where $\mathbf{o_i} = \langle o_{i1}, o_{i2}, \ldots, o_{id} \rangle$, $1 \leqslant i \leqslant N$. The j-th component of $\mathbf{o_i}$ is drawn from domain A_j.

We first partition the original data space into non-overlapping hyper-rectangles which we call *cells* to generate the *quantized space* $\mathcal{Q}$. The cells are obtained by segmenting every dimension A_i into m_i number of intervals. Each cell $\mathbf{c_i}$ is the intersection of one interval from each dimension. It has the form $\langle c_{i1}, c_{i2}, \ldots, c_{id} \rangle$ where $c_{ij} = [l_{ij}, h_{ij})$ is the right open interval in the partitioning of A_j. Each cell $\mathbf{c_i}$ has a list of statistical parameters $\mathbf{c_i} \cdot \mathbf{param}$ associated with it.

We say that a point $\mathbf{o_k} = \langle o_{k1}, \ldots, o_{kd} \rangle$ is contained in a cell $\mathbf{c_i}$, if $l_{ij} \leqslant o_{ki} < h_{ij}$ for $1 \leqslant j \leqslant d$. The list $\mathbf{c_i} \cdot \mathbf{param}$ keeps track of the statistical properties such as aggregation, mean, variance, and the probability distribution of the data points contained in the cell $\mathbf{c_i}$. In general, in grid-based approaches by a single pass through the dataset, the containment relations are discovered and appropriate

statistical parameters are computed. Each cell has information about the density of the data contained in the cell. Thus, the collection of $\mathbf{c_i} \cdot \mathbf{param}$ summarizes the dataset.

We choose the number of points contained in each cell as the statistic to be used. That is, we use $\mathbf{c_i} \cdot count$ to be the $\mathbf{c_i} \cdot \mathbf{param}$, which is called *density*. Thus a *density function* $\phi(\mathbf{c_i})$ is specified in the quantized space. A cell in the quantized space with 0 count value is called an *empty cell*. A cell in the quantized space with nonzero count value is called a *nonempty cell*. In our approach we apply wavelet transform on $\mathbf{c_i} \cdot count$ values. The *transformed space* $\mathscr{Q}'$ is the result after wavelet transform on the count values of the cells in the quantized space. The procedure can be easily illustrated by the linear space system. A wavelet transform is described by wavelet function w; thus a linear space system LS_w is defined based on w. A density function ϕ is defined on the quantized space which maps a cell $\mathbf{c_i}$ to $\mathbf{c_i} \cdot \mathbf{param}$. The result of LS_w is a new density function $\hat{\phi}_w$ on the transformed space which maps $\mathbf{c_i}$ to the transformed value of $\mathbf{c_i} \cdot \mathbf{param}$, as follows:



We introduce the following definitions to be used in the rest of the paper.

Definition 1. (Significant cell) Given a linear space system LS_w and density function ϕ, a cell $\mathbf{c}$ is a *significant* cell if $\hat{\phi}_w(\mathbf{c}) \geqslant \tau$, where $\tau = p \times \mathscr{V}$, p is an input parameter and $\mathscr{V}$ is a statistical value of $\hat{\phi}_w$.

The statistical value $\mathscr{V}$ can be the average value, maximum value, or summation of $\hat{\phi}_w$.

Definition 2. (ϵ-neighbor) A cell $\mathbf{c_1}$ is an ϵ-neighbor of cell $\mathbf{c_2}$ if both are significant cells in the transformed space and $D(\mathbf{c_1}, \mathbf{c_2}) \leqslant \epsilon$, where D is an appropriate distance metric and $\epsilon > 0$.

When the cells are ϵ-neighbor, they are ϵ-connected.

Definition 3. (Cluster) A cluster $\mathscr{C}$ is a set of significant cells $\{\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_m}\}$ which are ϵ-connected in the transformed space.

Informally, we say that a data point is an outlier if it does not belong to any clusters. We give a precise definition of the outliers as follows:

Definition 4. (Outlier) Given a linear space system LS_w and density function ϕ, a data point in a cell $\mathbf{c}$ is an outlier if $\hat{\phi}_w(\mathbf{c}) < \tau$, where τ is given by Definition 1.

Our definition can be related to the outliers as defined by Knorr and Ng (1998). For a dataset, it defines a density function ϕ. We define a kernel function g as follows:

$$g(\mathbf{c_i}) = \begin{cases} 1 & \text{if } D(\mathbf{c_i}, \vec{\mathbf{0}}) \leqslant \epsilon \\ 0 & \text{otherwise} \end{cases}$$

where $\vec{\mathbf{0}}$ is the origin in the data space and D is a distance function. A linear system LS_g is defined based on g. A data point $\mathbf{o_i}$ in LS_g is an outlier if $\hat{\phi}_g(\mathbf{o_i}) < p \times N$, where N is the size of the data points. Thus the outlier under this linear system is similar to the one defined in Knorr and Ng (1998).

Given a set of N data points $\mathbf{O} = \{\mathbf{o_1}, \mathbf{o_2}, \ldots, \mathbf{o_N}\}$ in the d-dimensional data space, our goal is to find the outliers as defined above. From the clustering point of view, outliers are the objects not located in the clusters of the dataset, usually called noise. For example, in gene research, a natural basis for organizing gene expression data is to group together genes with similar patterns of expression. Most regular gene expressions can be grouped into clusters. These gene expressions show a similar pattern. But there are some irregular gene expressions which cannot be grouped into any existing clusters because they have some attribute values which are distinct from others. This kind of gene belongs to outliers which may indicate new important patterns. It is difficult to detect this kind of gene because they mix with a vast amount of clustered gene expressions. After we remove the clustered gene expressions, the remaining outliers can be uncovered and explored carefully. We first propose FindOut to detect outliers in very large datasets with low dimensions. Then we extend it to efficiently handle datasets with high dimensions by incorporating the carefully designed data structure.

2.3. Wavelet Transform on Multidimensional Data

We first discuss the relationship between multidimensional data and multidimensional signals and show how to use wavelet transform to detect the inherent relationships in the data. We propose to look at the multidimensional data space from a signal-processing perspective. The collection of data in the multidimensional data space composes a d-dimensional signal. The high-frequency parts of the signal correspond to the regions of the data space where there is a rapid change in the distribution of data; they are the boundaries of clusters. The low-frequency parts of the d-dimensional signal which have high amplitude correspond to the areas of the data space where the data are concentrated. For example, in a two-dimensional data space, each row or column can be considered as a one-dimensional signal, so the whole data space will be a two-dimensional signal. Boundaries and edges of the clusters constitute the high-frequency parts of this two-dimensional signal, whereas the clusters themselves correspond to the parts of the signal which have low frequency with high amplitude. When the number of data point is high, we can apply signal-processing techniques to find the high-frequency and low-frequency parts of the d-dimensional signal which represents the data points, resulting in detection of the clusters. The key idea is to apply signal-processing methods to transform the space and find the dense regions in the transformed space. The data points in the remaining non-dense regions will be the outliers.

Wavelet transform is a signal-processing technique that decomposes a signal into different frequency sub-bands (for example, high-frequency sub-band and low-frequency sub-band). A one-dimensional signal s can be filtered by convolving the filter coefficients c_k with the signal values

$$\hat{s}_i = \sum_{k=0}^{M-1} c_k s_{i+k-\frac{M}{2}}, \tag{1}$$

where M is the number of coefficients in the filter and $\hat{s}$ is the result of convolution. Wavelet transform provides us with a set of interesting filters. For example, Fig. 1 shows the Cohen–Daubechies–Feauveau (2,2) biorthogonal wavelet (Uytterhoeven et al., 1997).

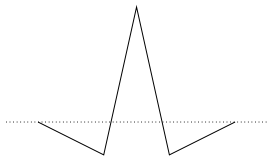


Fig. 1. Cohen–Daubechies–Feauveau (2,2) biorthogonal wavelet.

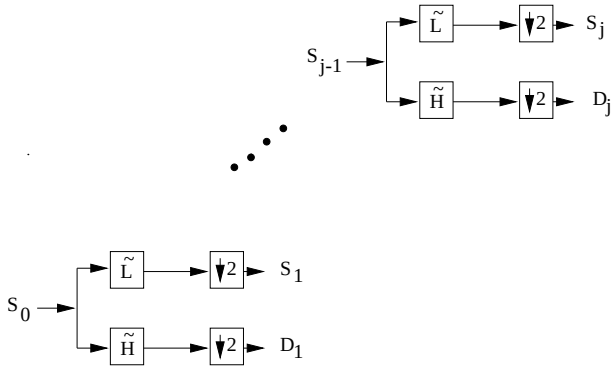


Fig. 2. Block diagram of multi-resolution wavelet transform.

We now briefly review wavelet-based multi-resolution decomposition. More details can be found in Mallat (1989). To have multi-resolution representation of signals we can use discrete wavelet transform. We can compute a coarser approximation of the one-dimensional input signal S_0 by convolving it with the low-pass filter $\tilde{L}$ and down-sampling the signal by two (Mallat, 1989). All the *approximation signals* S_j, $1 < j < J$ (J is the maximum possible scale) can thus be computed from S_0 by repeating this process. Figure 2 illustrates the method.

D_j denotes the difference between approximations S_j and S_{j-1} and is called the *detail signal* at the scale j. We can compute the detail signal D_j by convolving S_{j-1} with the high pass filter $\tilde{H}$ and returning every other sample of output. The wavelet representation of a discrete signal S_0 can therefore be computed by successively decomposing S_j into S_{j+1} and D_{j+1} for $0 \leqslant j < J$. This representation provides information about signal approximation and detail signals at different scales.

We can easily generalize the wavelet model to d-dimensional data space in which a one-dimensional transform can be applied multiple times. For example, in two-dimensional data space, we can represent the data space as an image where each pixel of an image corresponds to one cell in the data space, and the gray value of each pixel represents the number of data points which are in the corresponding cell. Wavelet transform can be applied along the axes x and y. It decomposes an image into an *average signal* (LL) and three *detail signals* which are directionally sensitive: LH emphasizes the horizontal image features, HL the vertical features, and HH the diagonal features. Figure 4 shows the wavelet representation of the image in Fig. 3 at three scales. At each level, LL is shown in the upper left quadrant, LH is shown in the upper right quadrant, HL is displayed in the lower left quadrant, and HH is in the lower right quadrant.

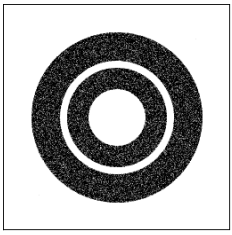


Fig. 3. A sample two-dimensional data space.

(a) (b) (c)

Fig. 4. Multi-resolution wavelet representation at (a) scale 1; (b) scale 2; (c) scale 3.

As shown in Fig. 4, LH shows the horizontal boundaries, HL shows the vertical boundaries, and HH shows the diagonal boundaries of the image.

2.4. Wavelet in High-Dimensional Space

In a high-dimensional space, it is expected that data will be sparse and most of the cells in the quantized space will be empty. An efficient way of storing only the nonempty cells in the quantized space is expected to drastically reduce the space complexity. We use a hash table approach to keep track of the nonempty cells only. The main idea is to efficiently represent high-dimensional data in limited memory and perform wavelet transform on this representation. But performing a convolution operation such as wavelet transform on this representation is a nontrivial problem. We present an accumulative approach to calculate wavelet transform in high-dimensional spaces.

2.4.1. Data Space Representation

In the quantized space, every cell $\mathbf{c_i}$ has the form of $\langle c_{i1}, c_{i2}, \ldots, c_{id} \rangle$ which is called the *key* or *index* for $\mathbf{c_i}$, where $c_{ij} = [l_{ij}, h_{ij})$ is the right open interval in the partitioning of dimension A_j. The address of a cell in the hash table can be calculated by applying appropriate hash function on the index of a cell. A hash table requires much less storage than a direct-address table, which was used in Sheikholeslami et al. (1998). Specifically, the storage requirement can be reduced from $O(m^d)$ to $\Theta(N' \times d)$, where N' is the number of nonempty cells in the quantized feature space. With hashing, a cell $\mathbf{c_i} = \langle c_{i1}, c_{i2}, \ldots, c_{id} \rangle$ is stored in the hash bucket $h(\mathbf{c_i})$.

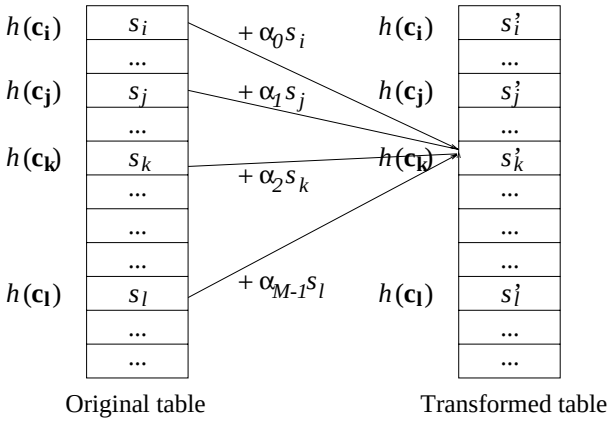


Fig. 5. Traditional approach of calculating wavelet transform.

2.4.2. Calculating Wavelet Transform on Hashed Feature Space

Wavelet transform is applied on the hashed representation of the quantized space to generate a new hash table consisting of only the nonempty cells. The neighboring cells in the original quantized space are scattered in the hash table because the hash table might not preserve the spatial locality. By scanning through the hash table, recovering the original quantized space and convolving the wavelet filter with each cell and its neighbors, we generate new cells in the transformed space. Directly recovering the original space is extremely time-consuming. We propose the 'accumulative' wavelet transform without directly recovering the original space from the hash table. For any nonempty cell $\mathbf{c_i} = \langle c_{i1}, c_{i2}, \ldots, c_{id} \rangle$, the cells which will contribute to its value in the transformed space along dimension A_j are, $\mathbf{c_k} = \langle c_{i1}, c_{i2}, \ldots, c_{ij} + k, \ldots, c_{id} \rangle$, where $\frac{M}{2} \leqslant k \leqslant \frac{M}{2}$ and M is the width of the wavelet filter. All the cells stored in the hash table will get new values after the wavelet transform is applied (see Fig. 5). Also, because of the convolution operation in the wavelet transform, some of the previously empty cells will become nonempty by receiving contributions from their neighboring cells. We call each potential nonempty cell a *receiver* and each old nonempty cell a *contributor*. In traditional implementation of the wavelet transform, each receiver knows which cells to ask for contributions. Thus, the algorithm scans through all the potential receivers. In a multidimensional array implementation, every cell is considered as a potential receiver. So the algorithm has to scan through the entire space of cells which we should try to avoid in high-dimension cases because of the exponential growth in the number of cells. But in the case of hashed implementation we only have information about the cells which are nonempty. Thus there are many potential receivers for which we have no knowledge of their hashed location. Therefore, it is not possible to use traditional scanning algorithms directly on the hashed and quantized feature space.

In our approach, each *contributor knows* its *receivers* and the addresses of the *receivers* can be calculated by using the hash function in time $O(l)$. So, instead

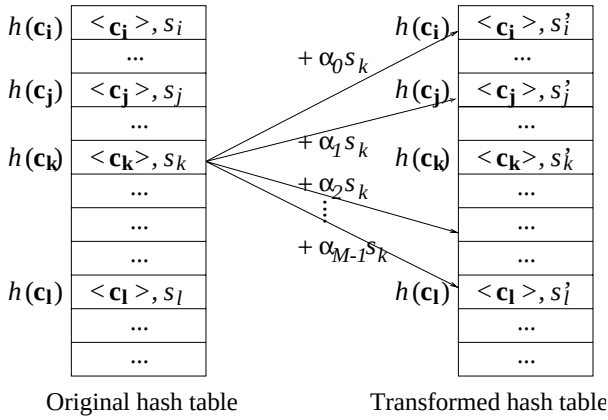


Fig. 6. Accumulative approach of calculating wavelet transform.

of receivers asking for values, contributors distribute values to receivers. Thus, it is sufficient to just scan through the contributors which are already saved in the hash table. With slight modifications, we can rewrite Formula 1 as

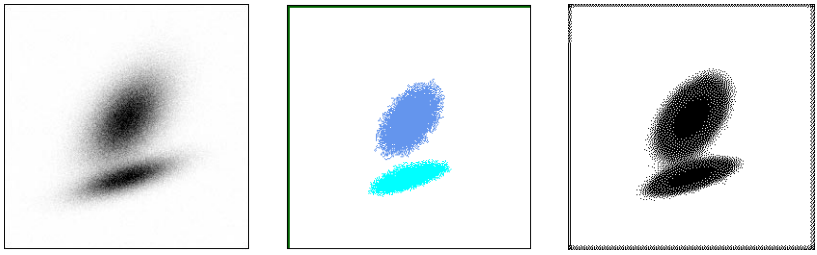
$$s'_{i+j} = \alpha_{\frac{M}{2}-j} s_i + \sum_{k=0}^{\frac{M}{2}-j-1} \alpha_k s_{i+j-\frac{M}{2}+k} + \sum_{k=\frac{M}{2}-j+1}^{M-1} \alpha_k s_{i+j-\frac{M}{2}+k}, \tag{2}$$

where $-M/2 \leqslant j < M/2$.

Using this formula while scanning the hash table, each old nonempty cell or contributor is multiplied by a coefficient and the result is accumulated into its *receiver* cells which are hashed into a new table (see Fig. 6).

2.5. Selection of Wavelet Filters

The motivation for using wavelet transform and thereby detecting outliers in the transformed space is drawn from the observations on the properties of outliers. From the signal point of view, outliers are localized high-frequency phenomena. They show significant changes when compared with their neighbors. If they occur as anything other than isolated aberrations in the data, they would be considered as part of the signal's structure, not outliers. For many real-world datasets which exhibit a more complex structure, some outliers are very close to the boundaries of the clusters. Therefore, they might be misclassified as normal data points. While usual Fourier methods do well at picking out frequencies from a signal, they are incapable of dealing properly with a signal that changes over time or space. These facts greatly limit the capability of Fourier methods in outlier detection. Wavelet is a refinement of the Fourier analysis, and it is well suited to this type of problem because wavelets can capture local changes. When wavelet transforms are applied to signals, local features (such as discontinuities and turning points) are preserved, while still discerning outliers (Donoho, 1992; Donoho et al., 1995). Greenblatt (1995) introduced the theory of applying wavelet to time series to detect outliers.



(a) (b) (c)

Fig. 7. (a) Original data space; (b) transformed space with a wavelet filter; (c) transformed space with a Gaussian filter.

Applying a discrete wavelet transform on a dataset, the wavelet transform must satisfy the following properties:

- orthogonality;
- adaptation to a finite quantization with flexible size;
- ease of implementation.

The orthogonality property is an important property when a wavelet transform is applied on a dataset with outliers, since it ensures mutual independence of wavelet coefficients generated from the normal data points and those generated from outliers. When we remove the wavelet coefficients generated from outliers, the majority of the data points will not be affected. Cohen et al. (1992) designed a *hat-shaped* wavelet (one of which is shown in Fig. 1) that satisfies the above three requirements.

The Cohen–Daubechies–Feauveau filter (shown in Fig. 1) can effectively smooth the dataset and remove noise. It emphasizes regions where points cluster, but simultaneously tends to suppress weaker information in their boundary. Intuitively, dense regions in the original space act as *attractors* for the nearby points and at the same time as *inhibitors* for the points that are not close enough. This means the majority of the data points automatically stand out and clear regions around them, so that they become more distinct. Figure 7 shows the experimental results when we apply wavelet and Gaussian filters on a dataset with two clusters. This dataset contains 500,000 data items in the two clusters plus 25,000 randomly distributed noise data items. As Fig. 7(b) shows, when we apply the wavelet transform the regular data points in the transformed space are more salient and thus easier to find. Figure 7(c) shows the result of using a Gaussian filter of a size similar to our wavelet filter. The two clusters in the original dataset merge in the transformed dataset when we apply a Gaussian filter. The outliers between these two clusters cannot be detected. When we try to raise the threshold to separate the two clusters, many regular data points will be misclassified as outliers. Therefore, the wavelet filter shows better performance than the Gaussian filter on dataset cleaning.

The multi-resolution property of wavelet transform can help detecting the clusters at different levels of detail. As we showed in Fig. 4, wavelet transform provides multiple levels of decomposition, which results in clusters at different scales from fine to coarse. Figure 4(a) shows the first level of decomposition, and its *LL* sub-band shows two circles with clear boundaries. Figure 4(b) shows the second level of decomposition, and its *LL* sub-band shows that the bound-

aries between the two circles become blur. Figure 4(c) shows the third level of decomposition, and its *LL* sub-band shows that the two circles merge together. The appropriate scale for choosing clusters can be decided based on the user's needs. Also, since applying wavelet transform is very fast, it makes our approach cost-effective. As it will be shown later, applying wavelet transform on very large datasets takes only a few seconds. Using parallel processing we can get even faster responses.

2.6. Outlier Detection

As discussed in Section 2.3, wavelet transform has two filters: a high-pass filter and a low-pass filter. In this section, we show the characteristics of its low-pass filter and how it can be applied in outlier detection. One step of wavelet transform is to convolve its input with a low-pass filter. A low-pass filter has the property of removing the outliers (noise) from the input. It considers the influence of the neighboring data points in computing its output. For example, Fig. 8(a) presents a one-dimensional low-pass *CDF filter*. Figure 8(b) shows a one-dimensional dataset (signal) containing some noise, with *amplitude* on the y-axis and *time* on the x-axis. The amplitude near the origin is high. It approaches zero as *time* approaches infinity in both directions, which corresponds to the noise part of the signal. Figure 8(c) shows the signal after convolving with the low-pass filter $g(t)$. The amplitude rapidly approaches zero when $time \geqslant 20$ or $time \leqslant -20$. It shows how the low-pass filter removes the outliers and smoothes the input data.

WaveCluster takes advantage of this property of low-pass filters and removes the outliers that may cause problems in detecting the clusters. However, in FindOut, our goal is to *find* the outliers and not to remove them. The main idea in FindOut is to remove the clusters from the original data and thus identify the outliers. We now present the definition of $\ominus$ to subtract quantized spaces.

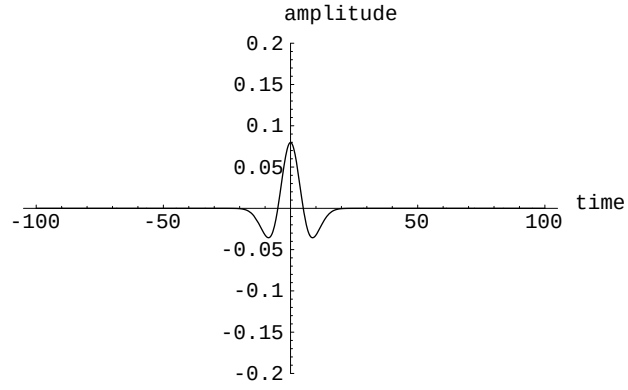
Definition 5. (Scale compatibility): Given two quantized spaces $\mathcal{Q}_1 = A_1 \times A_2 \ldots \times A_d$ and $\mathcal{Q}_2 = B_1 \times B_2 \ldots \times B_d$, let m_i be the number of cells along dimension A_i and n_i be the number of cells along B_i. $\mathcal{Q}_1$ and $\mathcal{Q}_2$ are *scale-compatible* if

$$\forall i \quad 1 \leqslant i \leqslant d \quad \exists s_i \in Z^+ \mid m_i = s_i n_i, \quad or \quad n_i = s_i m_i$$

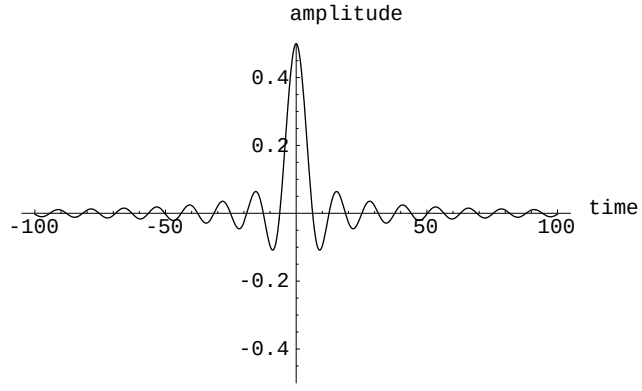
Based on Definition 5, we know that the original space $\mathcal{Q}$ and the transformed space $\mathcal{Q}'$ are scale-compatible. If the wavelet transform is applied once, each cell $\mathcal{Q}'(i, j)$ in $\mathcal{Q}'$ corresponds to four cells in the original space $\mathcal{Q}$: $\mathcal{Q}(2i, 2j)$, $\mathcal{Q}(2i, 2j+1)$, $\mathcal{Q}(2i+1, 2j)$ and $\mathcal{Q}(2i+1, 2j+1)$. Similarly, we can map the cells in the transformed space to the cells in the original space.

Definition 6. (Subtraction of quantized spaces): Given two scale-compatible spaces $\mathcal{Q}_1$ and $\mathcal{Q}_2$, $R = \mathcal{Q}_1 \ominus \mathcal{Q}_2$ is the result of their subtraction. If $\mathcal{Q}_1(a_1, a_2, \ldots, a_d)$ is nonempty and $\mathcal{Q}_2(b_1, b_2, \ldots, b_d)$ is empty then $R(c_1, c_2, \ldots, c_d)$ is nonempty, otherwise $R(c_1, c_2, \ldots, c_d)$ is empty, where $c_i = s_i a_i$ or $c_i = s_i b_i$.

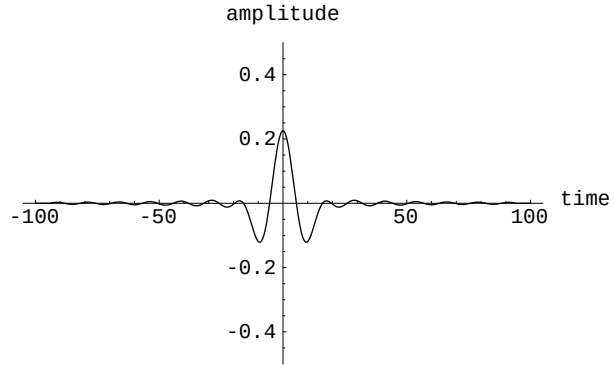
The main steps of FindOut's algorithm are listed in Algorithm 1. Based on the definition of cells in Section 2.2, the data points will be assigned to the cells in step 1.1 of the algorithm. We denote this quantized space as $\mathcal{Q}_{orig}$. Applying wavelet transform (step 1.2) on $\mathcal{Q}_{orig}$ results in a set of sub-bands $\mathcal{Q}_{l,t}$ for each level l ($1 \leqslant l \leqslant J$) of the transform. For each level l, $\mathcal{Q}_{l,t}$ consists of one approximation space (signal) $\mathcal{Q}_{l,A}$ and a set of detail spaces (signals) $\mathcal{Q}_{l,D}$ (as defined in Section 2.3). The clusters are detected in the transformed spaces $\mathcal{Q}_{l,A}$ in step 1.3. In step 1.4, the clusters will be mapped to the original data and will be represented in the



(a)



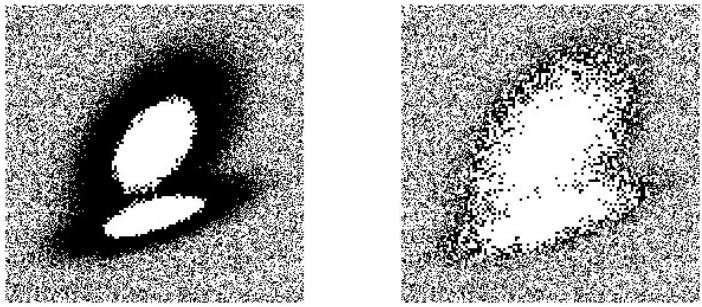
(b)



(c)

Fig. 8. (a) Low-pass filter $g(t)$; (b) original signal $f(t)$; (c) output signal $f'(t)$, $f'(t) = f(t) \star g(t)$.

spaces $\mathcal{Q}_{l,c}$. In step 2 of FindOut, the subtraction results of spaces will be stored in $\mathcal{Q}_{l,out}$, which only contain the outliers at different levels l. Since in wavelet transform we always down-sample the signals by two, all the above spaces are scale-compatible.



(a) (b)

Fig. 9. (a) Original data $\ominus$ clusters; (b) outliers detected by FindOut.

Algorithm 1. FindOut

Input: Multidimensional data points
Output: Outliers

1. Apply WaveCluster.
   1.1. Quantize the data space. /* $\mathcal{Q}_{orig}$ */
   1.2. Apply wavelet transform. /* $\mathcal{Q}_{l,t} = \{\mathcal{Q}_{l,A}, \mathcal{Q}_{l,D}\}$ */
   1.3. Find the connected components (clusters) in $\mathcal{Q}_{l,A}$.
   1.4. Map original data to the clusters. /* $\mathcal{Q}_{l,c}$ */
2. $\mathcal{Q}_{l,out} \leftarrow \mathcal{Q}_{orig} \ominus \mathcal{Q}_{l,c} \ominus \mathcal{Q}_{l,D}$.

From Fig. 7, if we remove the pixels from the original space which correspond to the cluster cells, we will get the space $\mathcal{Q}_{orig} \ominus \mathcal{Q}_{1,c}$, shown in Fig. 9(a). For better visualization, all nonempty cells in Fig. 9(a) (and 9(b)) are shown in black instead of gray scale. Figure 9(a) shows that in addition to the outliers, the points in the boundaries of clusters are also included in the result. The goal of WaveCluster is to detect the core of the main clusters in the dataset. Thus it applies those wavelet transforms that can clear the area around the clusters to make them more distinct. However, such points should not be considered as outliers. To detect the clusters we use the information in the low- frequency sub-bands (approximation signals) $\mathcal{Q}_{l,A}$. The other detail sub-bands $\mathcal{Q}_{l,D}$ have information about the boundaries of the clusters. We can apply this information and remove the cluster boundary points from the set of outliers. To do so, we subtract $\mathcal{Q}_{l,D}$ from the spaces in the previous step and get the results shown in Fig. 9(b).

2.6.1. Handling Complex Datasets

Real-world datasets exhibit more complex structures (Breunig et al., 2000). For example, a dataset might contain several clusters which have different densities. Figure 10 gives an example which contains two clusters. The density of cluster A is much higher than cluster B. There are two outliers: $\mathbf{o_1}$ and $\mathbf{o_2}$. The figure shows that: for any data point $\mathbf{p} \in B$, the distance between $\mathbf{o_1}$ to its nearest neighbor is smaller than the distance between $\mathbf{p}$ and its nearest neighbor in B. When we apply wavelet transform once on this dataset, either $\mathbf{o_1}$ and cluster B are classified as outliers, or $\mathbf{o_1}$ is classified as a cluster point because it is close to cluster A.

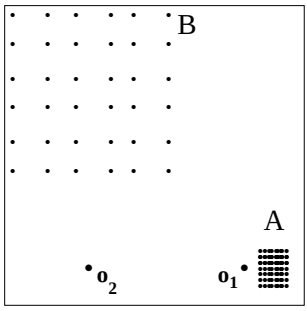


Fig. 10. A dataset contains clusters with different densities.

One step of wavelet transform with single resolution is not enough for detecting outliers. Wavelet has a multi-resolution property, which can be extended to detect the clusters at different density levels. Clusters with high density (dense clusters) are detected at higher resolution, and clusters with low density (sparse clusters) could not be detected because they are affected by the existing dense clusters. Once we remove the dense clusters from the dataset, clusters with relatively low density can be detected at lower resolution. Therefore, we design a sifting algorithm for detecting this type of outlier based on multi-resolution. The basic idea is: apply wavelet transform on the quantized data space, remove all of the cluster points, and repeat the sifting process on the remaining part of the data space T times. We choose $T = 3$ in our experiments. Algorithm 2 is a modified version of Algorithm 1 to detect outliers in a dataset with different densities. In Algorithm 2, we add step 3.5 to compress the data space $\mathcal{Q}_l$ to reduce the time complexity. For a given space $\mathcal{Q}$, the compressed space $\mathcal{Q}'$ is defined as follows: a cell $\mathcal{Q}'(i, j)$ is created from four neighboring cells $\mathcal{Q}(2i, 2j)$, $\mathcal{Q}(2i, 2j+1)$, $\mathcal{Q}(2i+1, 2j)$ and $\mathcal{Q}(2i + 1, 2j + 1)$ in space $\mathcal{Q}$, and $\mathcal{Q}'(i, j) = \sum_{\mu=0}^{1} \sum_{v=0}^{1} \mathcal{Q}(2i + \mu, 2j + v)$. We introduce the compressing process because the data distribution is sparse after we remove the dense clusters. Without losing any accuracy of representing the remaining data points, we shrink the size of the quantized space into a quarter of the original size. As a result, we reduce the memory and time requirements.

In Fig. 10, after applying the sifting algorithm with $T = 1$, cluster A is removed. Repeat the same step again, and cluster B is removed. Also, the outliers $\mathbf{o_1}$ and $\mathbf{o_2}$ are detected.

Algorithm 2. Enhanced FindOut

Input: Multidimensional data points
Output: Outliers

1. Quantize the data space. /* $\mathcal{Q}_{orig}$ */
2. $\mathcal{Q}_l \leftarrow \mathcal{Q}_{orig}$, where $l = 1$.
3. For $l = 1$ to T do /* Sifting process */
   3.1. Apply wavelet transform on $\mathcal{Q}_l$. /* $\mathcal{Q}_{l,t} = \{\mathcal{Q}_{l,A}, \mathcal{Q}_{l,D}\}$ */
   3.2. Find the connected components (clusters) in $\mathcal{Q}_{l,A}$.
   3.3. Create a sub-band $\mathcal{Q}_{l,A'}$ for all clusters.
   3.4. $\mathcal{Q}_{l+1} \leftarrow \mathcal{Q}_l \ominus \mathcal{Q}_{l,A'} \ominus \mathcal{Q}_{l,D}$.
   3.5. Compress $\mathcal{Q}_{l+1}$
4. $\mathcal{Q}_{l,out} \leftarrow \mathcal{Q}_{T+1}$.

2.7. Time Complexity

Let N be the number of data points in the dataset, where N is a very large number. Assume that the data points are d-dimensional, resulting in a d-dimensional data space. FindOut's time complexities on the low or high number of dimensions are different. We will first discuss the low-dimensional case. The time complexity of the first step of the FindOut algorithm is $O(N)$, because it scans all the data points and assigns them to the corresponding cells. Assuming m cells in each dimension of the data space, there would be $K = m^d$ cells. The complexity of applying wavelet transform on the quantized data space (step 3.1) will be $O(ldK) = O(dK)$, where l is a small constant representing the length of filter used in the wavelet transform. Since we assume that the value of d is low, we can consider it as a constant, thus $O(dK) = O(K)$. To find the connected components in the transformed data space (step 3.2), the required time will be $O(cK) = O(K)$, where c is a small constant. The required time for steps 3.3, 3.4, and 3.5 will be $O(K)$. If we apply T levels of sifting, since for each level we shrink the space by two, for $d \geqslant 2$, the required time would be

$$\begin{aligned} O(K + \tfrac{K}{2^d} + \tfrac{K}{(2^d)^2} + \ldots + \tfrac{K}{(2^d)^T}) &= O(K \textstyle\sum_{i=0}^{T} \frac{1}{(2^d)^i}) \\ &= O(K \textstyle\sum_{i=0}^{T} (2^{-d})^i) \\ &= O(K \tfrac{1-(2^{-d})^{T+1}}{1-2^{-d}}) \\ &\leqslant O(\tfrac{4}{3}K) \end{aligned}$$

That means that the cost of applying the process of sifting for multiple levels would be at most $O(\frac{4}{3}K)$. After reading data points, the processing of data is performed in steps 3.1 to 3.5 of the algorithm. Thus the time complexity of processing data (without considering I/O) would be $O(K)$, which is independent of the number of data points (N). Since this algorithm is applied on very large datasets with low number of dimensions, we can assume that $N \geqslant K$. As an example, for a dataset with 1,000,000 data points when the number of dimensions d is less than or equal to 6, and the number of intervals m is 10, this condition holds. Thus based on this assumption the overall time complexity of the algorithm will be $O(N)$.

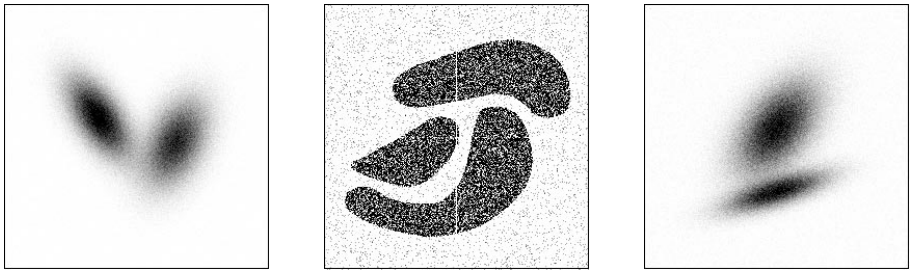
When the dimension d is high, we may have $N < K = m^d$. In Section 2.4, we have provided a solution of using a hash-based data structure for handling the high-dimensional datasets. The total time for quantization is $O(d \times N \times \log_2 N)$. The time complexity of applying wavelet transform on one dimension is $O(N \times \log_2 N)$. The total time taken in applying wavelet transform in all d-dimensions is the summation of the time taken in applying wavelet transform in each dimension. Therefore, it is $O(d \times N \times \log_2 N)$. The complexity of the subtraction step is $O(d \times N \times \log_2 N)$. If we apply T levels of wavelet transforms to detect outliers, the time complexity of FindOut is $O(T \times d \times N \times \log_2 N)$.

3. Experiments

In this section, we evaluate the performance of FindOut and demonstrate its effectiveness and efficiency on different types of distributions of data. Tests are done on both synthetic and real-world datasets. All of the tests are done on a Sun UltraSPARC 168 MHz machine having 1024 MB of main memory.

Table 1. Parameters for generating DS3

Parameters	μ_x	μ_y	σ_x	σ_y	ρ
Cluster1	125.0	55.0	60.0	13.0	0.7
Cluster2	125.0	120.0	50.0	30.0	0.5



(a) (b) (c)

Fig. 11. The datasets used in the experiments. (a) DS1; (b)DS2-10; (c) DS3-10.

3.1. Synthetic Datasets

We present the experiments on two-dimensional datasets so that the results can be easily visualized. Figure 11 shows the synthetic datasets used in the experiments. We used a technique similar to one described in Zhang et al. (1996) to generate the dataset DS3. Two cluster centers are first placed on the two-dimensional plane and then 500,000 data objects are spread following 2-D normal distribution around these points. For the 2-D normal distribution we used the polar method given in Knuth (1998). The dataset is shown in Fig. 11(c). The parameters used for this are shown in Table 1, where μ_x and μ_y specify the mean in each dimension, i.e., the location of the cluster center σ_x and σ_y specify the variance in each dimension, and ρ specifies the correlation coefficient between the variables in each dimension.

DS3-5, DS3-10, DS3-15, DS3-20, DS3-25, and DS3-50 are different versions of DS3 with different percentages of outliers. For example, DS3-5 has 5% outliers. Similarly, we generated DS1. DS1 contains 735,000 data points with 5% outliers, as shown in Fig. 11(a). Dataset DS2 was generated by spreading 65,193 data points in three manually generated regions. Each region has about 21,700 uniformly distributed data points. Each region represents a cluster, as shown in Fig. 11(b). The three clusters have different densities because their regions have nonequal areas. The datasets DS2-5, DS2-10, DS2-15, DS2-20, and DS2-25 refer to the same as DS2 but with 5%, 10%, 15%, 20%, and 25% outliers, respectively.

3.1.1. Finding Outliers in Large Datasets

All the datasets used in the experiments contain more than 50,000 data points. For a fixed quantization, larger datasets do not affect CPU time of the outlier detection. FindOut can successfully handle a large number of data points. Figure 12 shows the outliers detected for DS1.

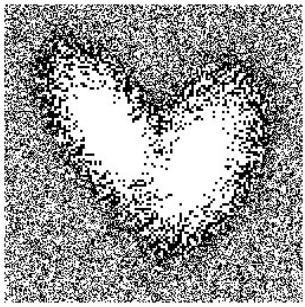


Fig. 12. The outliers of DS1.

3.1.2. Different Percentage of Outliers

We did experiments on five DS2 variants. Figure 13 only shows the outliers that have been detected in two datasets with different percentages of randomly generated outliers. To compare the behaviors of other clustering algorithms, we applied BIRCH on the dataset DS2 and detected three clusters, and then we subtracted the three clusters from the original dataset and got the outliers detected by BIRCH. Figure 13(a) shows the original datasets, Fig. 13(b) shows the outliers detected by FindOut, Fig. 13(c) shows the clusters detected by BIRCH, and Fig. 13(d) shows the outliers detected by BIRCH. From the experimental results, we observe that both approaches can effectively detect different scales of outliers. We know that WaveCluster (Sheikholeslami et al., 1998a) is outlier invariant because the wavelet transform can effectively filter out the outliers. When we subtract wavelet clusters from the original dataset, all of the outliers can be detected. BIRCH shows a similar behavior when it is used to detect clusters. Figure 13(c) shows that when the percentage of outliers over all of the data points increases, the shapes of the clusters detected by BIRCH do not change significantly. The distribution pattern of the dataset can also affect the performance of detection. WaveCluster can successfully cluster any complex patterns consisting of nested or concave clusters. Therefore, when it is applied to outlier detection it can find all of the outliers which locate close to the *cave* parts of the clusters, as shown in Fig. 13(b). BIRCH uses spheres to represent clusters, so that the outliers in the sphere are classified as cluster points and the cluster points outside the sphere are detected as outliers, as shown in Fig. 13(d).

3.1.3. The Important Role of Thresholding in Outlier Detection

The selection of threshold p (defined in Section 2.2) directly affects the results of the outlier detection. Higher thresholds may result in a larger number of outliers. We used a simple uniform threshold value in the above tests. Figure 14 shows different threshold values applied to outlier detection in dataset DS3-10. We can see that the real dense and sparse areas do not change too much when the threshold changes. But the *border* area between them is highly affected by the chosen threshold.

The selection of the threshold is determined by three main factors: the data distribution, the wavelet filter, and the level of the wavelet transform. Donoho Donoho et al. (1995) introduced a thresholding method (called *sure thresholding*) which uses the wavelet coefficients at each level j of the wavelet transform

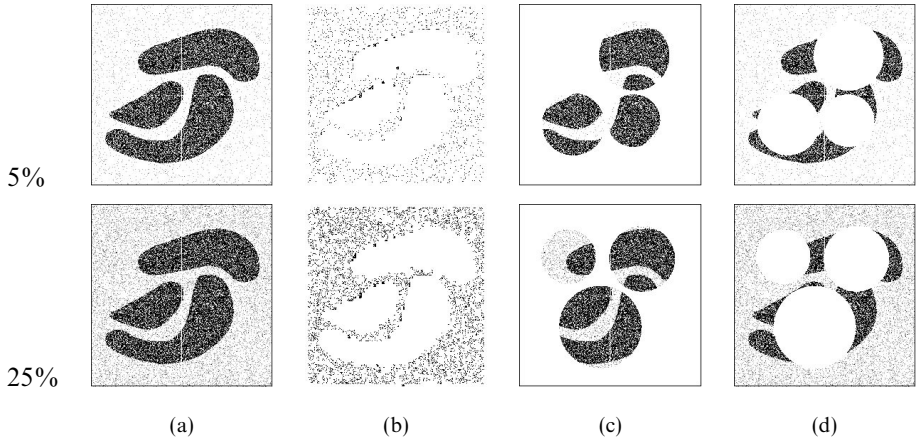


Fig. 13. FindOut on datasets with different levels of outliers. (a) Datasets with outliers; (b) outliers detected by FindOut; (c) clusters detected by BIRCH; (d) outliers detected by BIRCH.

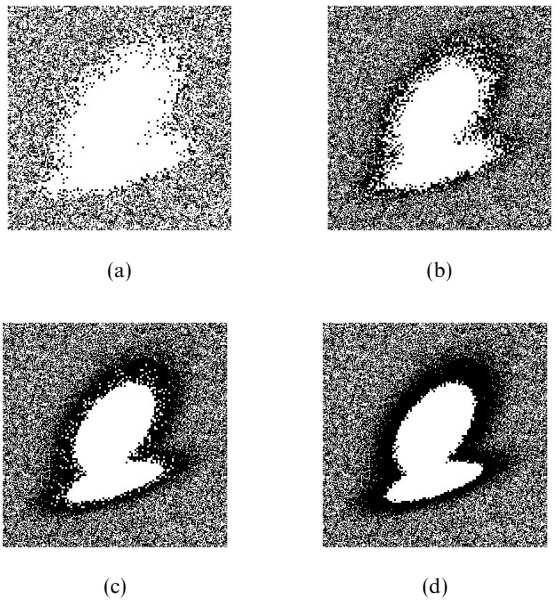


Fig. 14. The outliers of DS3-10 with different threshold values p. (a) $p = 0.2$; (b) $p = 0.05$; (c) $p = 0.10$; (d) $p = 0.50$.

to choose a threshold λ_j. The sure thresholding is a popular data-dependent threshold selection method. We adopt it in our threshold estimation. Let the wavelet coefficients of a dataset without any outliers be true wavelet coefficients and the wavelet coefficients of the dataset with outliers be corrupted wavelet coefficients. Sure thresholding can help uncover the true wavelet coefficients from the corrupted coefficients. The basic idea behind the method is to find a set of estimated wavelet coefficients based on thresholding, and this set of the estimated wavelet coefficients can best approximate the true wavelet coefficients without noise. We call the difference between the estimated coefficients and the true

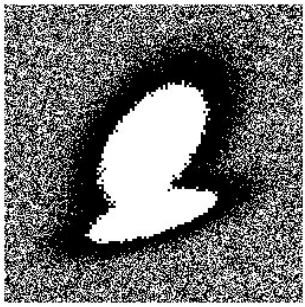


Fig. 15. The outliers of DS3-10 with sure thresholding.

coefficients *risk*. In reality, the true wavelet coefficients without noise may not be available. Therefore, we use a function to find the solution to reduce the risk without knowing the true coefficients:

$$SURE(\eta; W) = d - 2\|\{k : |w_k| \leqslant \eta\}\| + \sum_{k=1}^{d} (\min(|w_k|, \eta))^2$$

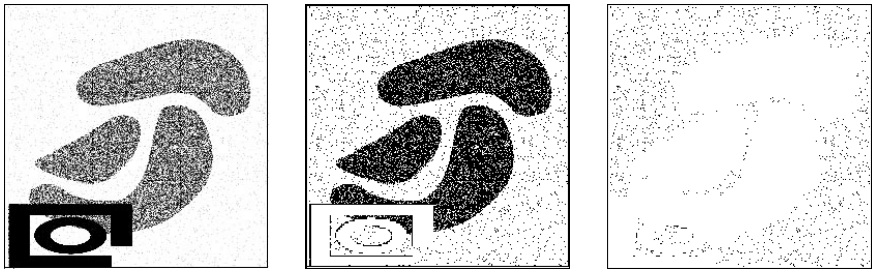
where $\|S\|$ denotes the cardinality of the set, d is the number of wavelet coefficients and $W(w_1, w_2, \ldots, w_d)$ are the corrupted wavelet coefficients before thresholding. The threshold λ is set so as to minimize the difference between the estimated wavelet coefficients and the true coefficients as follows:

$$\lambda = arg \min_{\min\{|w_i|\} \leqslant t \leqslant \max\{|w_i|\}} SURE(t, W)$$

The mathematical details are discussed in Donoho et al. (1995). It was proved that the sure thresholding works effectively in many areas (Donoho et al., 1995). Using this thresholding method, we can reduce the number of misclassified data points. Figure 15 shows the detected outliers based on the sure thresholding. We observe that there is even more clear border between the normal data points and outliers with this thresholding than with a pre-specified threshold value.

3.1.4. Outlier Detection in the Dataset with Different Scales of Density

We created a dataset with five clusters which have different densities. There are two clusters which have high density, and they are located in the lower left corner in Fig. 16(a). There are three clusters which have low density, and they are located in the center of Fig. 16(a). Here, the color intensity represents the density of the five clusters. The average density of the black clusters is 20 times more than the density of the gray clusters. When we apply the wavelet transform once, the two clusters with higher density are removed from the dataset. The outliers and other three clusters are left, as shown in Fig. 16(b). When we continue to apply wavelet transform on the remaining dataset, the outliers are detected, as shown in Fig. 16(c). This experiment demonstrates that the multi-resolution property of wavelet transform can be used to detect the clusters which have different densities; therefore, the performance of FindOut is not affected by complex data distribution.



(a) (b) (c)

Fig. 16. The outliers of DS4 with different scales of density.

Table 2. Execution time (in seconds) for different number ofdimensions and quantizations *m*

Number of dimensions	*m*				
	32	64	96	128	256
5	118	132	216	256	340
10	1,042	1647	2,166	2,680	3,280
15	2,374	3958	7,332	7,156	8,080
20	3,742	6153	10,983	10,660	13,400

3.1.5. Timing on Different Quantizations and Wavelet Transform

Quantization *m* affects running time. When *m* is low (very coarse quantization), the total number of cells is small, and thus it takes less time. Running time is also affected by wavelet filters of different width. The width of the wavelet filter determines the range of the neighborhood. The larger the neighborhood of a data point that is considered, the longer it takes in detecting outliers. Figure 17 shows CPU time when wavelets Haar, Cohen–Daubechies–Feauveau (2,2), Cohen–Daubechies–Feauveau (4,2) (Uytterhoeven et al., 1997), and Daubechies (Strang and Nguyen, 1996) are used on dataset DS3-10 under different quantizations.

Knorr and Ng (1998) and Breunig et al. (2000) showed that their cell-based algorithm is the most efficient when the number of dimensions is less than or equal to 4. However, for higher number of dimensions, its efficiency decreases (Knorr and Ng, 1998). Using the hash-based strategy, FindOut can work with high dimensions with arbitrary quantization. Table 2 shows the timing requirements of FindOut with different numbers of dimensions and quantizations. The size of the testing dataset is 100,000.

3.2. Real-World Datasets

3.2.1. Soccer Data

In Breunig et al. (2000), the authors conducted experiments on soccer-player information from the 'Fußball 1. Bundesliga' (the German national soccer league) for the season 1998/99. The dataset consists of 375 players. A detailed explanation of the attributes used can be found in Breunig et al. (2000). We repeated their

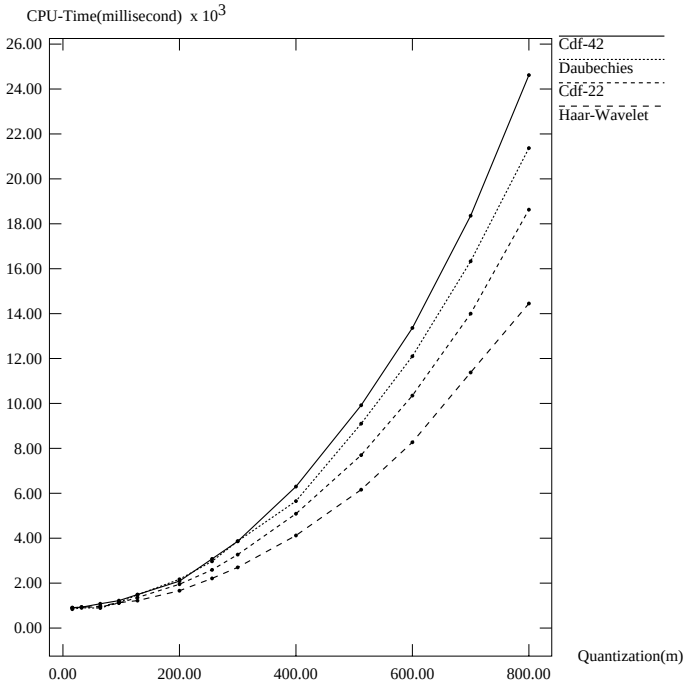
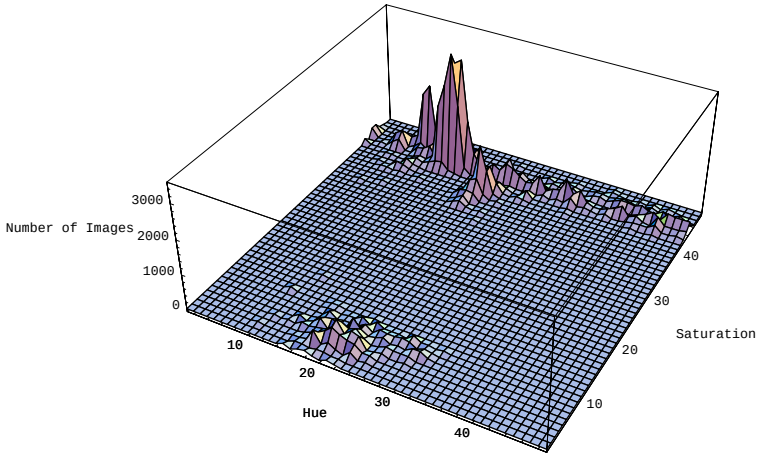


Fig. 17. Run time under different wavelet and quantizations.

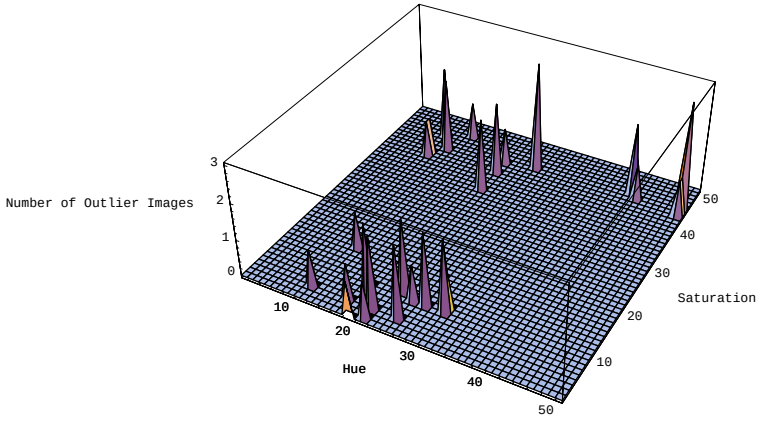
experiment on the dataset provided by the authors. The LOF approach detects five strongest outlier points which represent players Michael Preetz, Michael Schjönberg, Hans-Jörg Butt, Ulf Kirsten, and Giovane Elber. They are listed in decreasing order based on their LOF factors. FindOut found the first four outliers because they are far away from any other player's record. We do not consider Giovane Elber's record as an outlier because his record (games played: 21; goals scored 13) is very close to the records of Bernhard Winkler (games played: 24; goals scored 11) and Carsten Jancker (games played: 26; goals scored: 13). Their LOF values are close to each other. Therefore, it is not an outlier according to our definition. The LOF value indicates the degree of outlier-ness for each data point. The outliers with the highest LOF values, (known as *strongest outliers*), constitute the most important information in these applications. Our approach can identify these outliers which Breunig et al. (2000) find in their experiment.

3.2.2. Image Feature Dataset

Dataset R is created based on the image feature vectors from the University of California at Irvine Machine Learning Repository. The original dataset has 700 data points and each element describes hue and saturation values of segmentations of seven types of images: brickface, sky, foliage, cement, window, path, and grass. Our experiment showed that our algorithm can detect all of the exceptional data points which represent the exceptional images. To scale up the test, without loss of generality, we expand R to 392,402 data points by randomly generating 10



(a)



(b)

Fig. 18. Image feature dataset. (a) Dataset R; (b) detected outliers.

to 1000 data points around each basic feature vector (data point) within the hyper-sphere centered at the basic feature vector with a small-radius r. Figure 18 shows the histogram of dataset R and detected outliers. In Fig. 18, the three dimensions (n, h, s), *number of images*, *hue*, and *saturation*, represent that there are n images that have hue value h and saturation value s. Figure 18(a) shows the distribution of the image set. Figure 18(b) shows the detected outliers which

represent the exceptional images in the original dataset. The *needle*-shape peaks in Fig. 18(b) denote the images which have exceptional hue and saturation values, and the heights of the *needle* represent the number of such images. We use a small scale for the axis of *number of outliers* in Fig. 18(b) to better visualize the detected outliers. From the experiment, we discovered that all the detected points have exceptional hue and saturation values.

3.2.3. Chase Bank Dataset

A large customer dataset is collected to detect bankruptcy. This dataset contains 39,244 elements with 18 dimensions. We performed our outlier detection algorithm on the dataset to measure the deviation of abnormal customers. Any data which are detected as outliers will be considered as bankruptcy cases. The results confirm the historical records of bankruptcy obtained from the bank in most cases. In comparison, FindOut performed much better than many algorithms which are based on decision trees.

4. Conclusion

This paper has addressed an interesting problem of how to apply signal-processing techniques to solve important data-mining problems and the effectiveness of such approaches. We presented a novel deviation (or outlier) detection approach, termed FindOut, based on wavelet transform. By combining outlier detection with clustering, FindOut can successfully identify various percentages of outliers from large datasets. Thus our approach is highly cost-effective. Experimental results on very large datasets have demonstrated the efficiency and effectiveness of the proposed approach.

Acknowledgements. This research is partially supported by NSF grants IIS-9733730, IIS-9905603, and EIA-9983430.

References

Arning A, Agrawal R, Raghavan P (1996) A linear method for deviation detection in large databases. In Proceedings of the second international conference on knowledge discovery and data mining, pp 164–169, Portland, Oregon, August, ACM Press.

Breunig MM, Kriegel HP, Ng R, Sander J (2000) LOF: identifying density-based local outliers. In Proceedings of the ACM SIGMOD CONFERENCE on management of data, Dallas, TX, 16–18 May, pp 93–104

Barnett V, Lewis T (1994) Outliers in statistical data (3rd edn). Wiley, New York

Cohen A, Daubechies I, Feauveau JC (1992) Biorthogonal bases of compactly supported wavelets. Communications in Pure and Applied Mathematics 45: 485–560

Donoho DL (1992) De-noising by soft-threshold. Technical report 409, Department of Statistics, Stanford University

Donoho DL, Johnstone IM, Kerkyacharian G, Picard D (1995) Wavelet shrinkage: symptopia? Journal of the Royal Statistical Society, Series B 57 301–369

Greenblatt SA (1995) Wavelets in econometrics: An application to outlier testing. Computational Economic Systems: Models, Methods and Econometrics, August. http://econwpa.wustl.edu:8089/eps/em/papers/9410/9410001.pdf

Hoaglin D, Mosteller F, Tukey J (1983) Understanding robust and exploratory data analysis. Wiley, New York

Jain R, Kasturi R, Schunck BG (1995) Machine vision. MIT PRESS, Cambridge, MA

Johnson R (1992) Applied multivariate Statistical analysis. Prentice-Hall, Englewood Cliffs, NJ

Knorr E, Ng R (1997) A unified notion of outliers: properties and computation. In Proceedings of the international conference on knowledge discovery and data mining, pp 219–222, Newport Beach, CA, USA

Knorr E, Ng R (1998) Algorithms for mining distance-based outliers in large datasets. In Proceedings of the 24th VLDB conference, New York, August, pp 392–403

Knorr E, Ng R (1999) Finding intensional knowledge of distance-based outliers. In Proceedings of the 25th VLDB conference, Edinburgh, September, pp 211–222

Knuth DE (1998) The art of computer programming (3rd edn). Addison-Wesley, Reading, MA

Mallat S (1989) A theory for multiresolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 11: 674–693

Ramaswamy S, Rastogi R, Shim K (2000) Efficient algorithms for mining outliers from large data sets. In Proceedings of the ACM SIGMOD CONFERENCE on management of data, Dallas, TX, 16–18 May, pp 427–438

Ruts I, Rousseeuw PJ (1996) Computing depth contours of bivariate point clouds. Computational Statistics and Data Analysis 23: 153–168

Sarawagi S, Agrawal R, Megiddo N (1998) Discovery-driven exploration of OLAP data cubes. In Proceedings of the sixth international conference on extending database technology (EDBT), Valencia, Spain, March

Sheikholeslami G, Chang W, Zhang A (1998a) Semantic clustering and querying on heterogeneous features for visual data. In Proceedings of the 6th ACM international multimedia conference (ACM Multimedia '98), Bristol, UK, September, pp 3–12

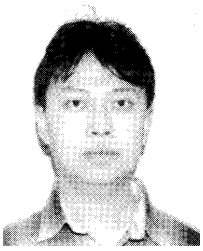
Sheikholeslami G, Chatterjee S, Zhang A (1998) WaveCluster: a multi-resolution clustering approach for very large spatial databases. In Proceedings of the 24th VLDB conference, August, pp 428–439

Strang G, Nguyen T (1996) Wavelets and filter banks. Wellesley-Cambridge Press, Wellesley, MA, USA

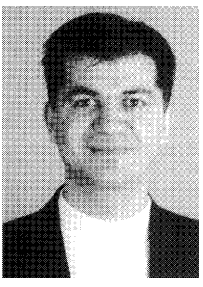
Uytterhoeven G, Roose D, Bultheel A (1997) Wavelet transforms using lifting scheme. Technical report ITA-Wavelets Report WP 1.1, Department of Computer Science, Katholieke Universiteit Leuven, Belgium, April

Zhang T, Ramakrishnan R, Livny M (1996) BIRCH: An efficient data clustering method for very large databases. In Proceedings of the 1996 ACM SIGMOD international conference on management of data, Montreal, Canada, pp 103–114

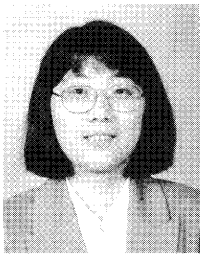
Author Biographies



Dantong Yu is currently working as a computer analyst with the US ATLAS group at Brookhaven National laboratory. He received his Ph.D. in computer science from the State University of New York at Buffalo in December, 2000. He earned his B.S. in computer science from Beijing University, China, in 1995. His research interests include grid computing, distributed data management, data mining, and contented-based image retrieval.



Gholamhosein Sheikholeslami is currently working in the Internet Knowledge Framework team under the Internet Technology Group at Cisco Systems, Inc. He received his Ph.D. in computer science from the State University of New York at Buffalo in 1999. He also earned an M.S. in computer science and an M.S. in electrical and computer engineering from the State University of New York at Buffalo in 1995 and 1996, respectively. He attended Tehran University, Tehran, Iran, where he received his B.S. in computer science in 1989. His research interests include multimedia database systems, visual database clustering, data mining, metadata and knowledge management.



Aidong Zhang received her Ph.D. degree in computer science from Purdue University, West Lafayette, Indiana, in 1994. She was an Assistant Professor in the Department of Computer Science and Engineering at State University of New York at Buffalo, New York, from 1994 to 1999, and an Associate Professor since 1999. Her research interests include content-based image retrieval, geographical information systems, distributed database systems, multimedia database systems, digital libraries, and data mining. She serves on the editorial boards of the *International Journal of Multimedia Tools and Applications*, *International Journal of Distributed and Parallel Databases*, and ACM SIGMOD DiSC (Digital Symposium Collection). She has also served on various conference program committees. Dr Zhang is a recipient of the National Science Foundation CAREER award.

Correspondence and offprint requests to: A. Zhang, Department of Computer Science and Engineering, State University of New York at Buffalo, Buffalo, NY 14260, USA. Email: azhang@cse.buffalo.edu