Outlier detection and evaluation by network flow

Ying Liu* and Alan P. Sprague

Department of Computer and Information Sciences, University of Alabama at Birmingham, Birmingham, AL 35294, USA
E-mail: liuyi@cis.uab.edu
E-mail: sprague@cis.uab.edu
*Corresponding author

Abstract: This paper introduces a novel method to separate abnormal points from normal data, based on network flow. This approach uses the Maximum Flow Minimum Cut theorem from graph theory to find the outliers and strong outlier groups, and evaluate the outliers by outlier degrees. Similar outliers are discovered together and delivered to the user together; in an application where outliers are the points of the greatest interest, this will allow similar outliers to be analysed together. Effectiveness of the method is demonstrated in comparison with three other outlier detection algorithms. Further experimental application testifies this algorithm can improve the query accuracy on a content-based image data set. This algorithm is effective on higher dimensional data as well as low dimension.

Keywords: outlier; clustering; outlier detection; network flow; Maximum-Flow/Minimum-Cut; graph theory.


Biographical notes: Ying Liu is a PhD candidate at Department of Computer and Information Sciences at the University of Alabama at Birmingham. Her research direction is Data Mining, especially outlier detection; the design and analysis of outlier detection built from network model; application of outlier detection; outlier detection algorithms comparison. The research in this paper was rewarded the 1st place of Graduate Student Research Day (2005) – Mathematics and Computer Sciences group. Her research interests also include clustering, classification. She received her Master degree in Computer Sciences at UAB (2004). She graduated in the Department of Computer Sciences from Wuhan University (1998).

Alan P. Sprague is an Associate Professor at Department of Computer and Information Sciences at the University of Alabama at Birmingham. His research interests include (1) Hospital infection control surveillance (controlling the increasing resistance by bacteria to antibiotics). (2) Algorithms for finding frequent sets. (3) Algorithms for special classes of graphs. The emphasis here is on efficiency. Often, this involves exploring the ‘border’ between polynomial time problems and NP-complete problems. (4) Disjoint paths in graphs. He teaches Data Mining, Automata, Formal Language Theory, Computational Geometry, Design and Analysis of Algorithms, etc. He received his PhD in Computer Sciences (1988) and in Mathematics (1972) at Ohio State University.

1 Introduction

Outliers are those points which are different from or inconsistent with the rest of the data (Han and Kamber, 2000). Novelty detection (Markou and Singh, 2003), chance discovery (McBurney and Ohsawa, 2003) and anomaly detection are different names for outlier detection. Unusual usage of credit cards or telecommunication services may be an indication of fraud. Similarly, in a computer system, an unusual sequence of system calls may indicate that the user is an intruder. Outlier detection is useful in medical analysis for finding unusual responses to various medical treatments.

In application such as these, analysts will likely have interest in investigating the outliers. It would be helpful to the analyst for outliers to be grouped, so that similar outliers can be analysed together. The algorithm presented here, based on network flow (Ford and Fulkerson, 1962), provides such a grouping.

Three properties that are very desirable in an outlier detection algorithm are the following.

1 It should group similar outliers together.

For an application in which an analyst may want to examine outliers, it is very useful for outliers to be so grouped. This concern for organising the information to be presented to the analyst, although
commonplace with regard to association rules (Toivonen et al., 1995), has not been commented on with outlier detection.

2. It should assign an outlier degree to each outlier (or outlier group). This allows a user to choose what percentage of the data that should be called outlying, without running the algorithm again.

3. It should be not very sensitive to change in values of parameters. Which points are the strongest outliers should be fairly stable as values of parameters change.

The following is the organisation of this paper. Section 2 gives a short review of outlier detection algorithms. Section 3 introduces network flow. Sections 4 and 5 describe the method. Section 6 presents experiments. Section 7 compares this algorithm with several other outlier detection algorithms mentioned above and Section 8 concludes this paper.

2 Outlier detection algorithms review

Outlier detection, both univariate and multivariate, has long been a concern of statistics. Mahalanobis distance is a classical statistical method of outlier detection; several more robust methods are described in Rousseau and Zomeren (1990) and Rocke and Woodruff (1996). PAELLA (Limas et al., 2004) designs a mixture model which combines Expectation Maximisation (EM) algorithm, clustering analysis and Mahalanobis distance and detects for both normal and non-normal multivariate data sets. SmartSifter (Yamanishi et al., 2004) is an outlier detection method that utilises statistical learning theory. SmartSifter is able to handle dynamically changing data and it can deal with both categorical and continuous variables.

Hodge and Austin (2004) gave a survey of outlier detection methods, focusing especially on those developed within the Computer Science community. Supervised outlier detection methods require prelabelled data, tagged as normal or abnormal and are suitable for data whose characteristics do not change through time. In semi-supervised recognition methods, the normal class is taught and data points that do not resemble normal data are considered outliers. Unsupervised methods process data with no prior knowledge. Four categories of unsupervised outlier detection algorithms are the following.

1. In a clustering-based method, like ROCK (Guha et al., 2000) and DBSCAN (Ester et al., 1996), outliers are byproducts of the clustering process and will not be in any resulting cluster.

2. The density-based method of Breunig et al. (2000) uses a Local Outlier Factor (LOF) to find outliers. If the object is isolated with respect to the surrounding neighbourhood, the outlier degree would be high and vice versa.

3. The distribution-based method of Knorr and Ng (1997) defines, for instance, outliers to be those points \( p \) such that at most 0.02\% of points are within 0.13 \( \sigma \) of \( p \).

4. Distance-based outliers are those objects that do not have ‘enough’ neighbours (Knorr and Ng, 1998, 1999). The problem of finding outliers can be solved by answering a nearest neighbour or range query centred at each object \( O \).

Several mathematical methods can be applied to outlier detection. Wavelets may be used to transform the original feature space, and then find dense regions in the transformed space. A wavelet transform decomposes a signal into different frequency subbands. By using low-pass filters, a wavelet transform can detect outliers (Bilen and Huzurbazar, 2002; Even 1979). Principal Component Analysis (PCA) may also be used to detect outliers. PCA computes orthonormal vectors that provide a basis for the input data. Then principal components are sorted in order of decreasing ‘significance’ or strength. The size of the data can be reduced by eliminating the weaker components which are with low variance (Chapra and Canale, 2006). Convex hull method finds outliers by peeling off the outer layers of convex hulls (Johnson et al., 1998). Data points on shallow layers are likely to be outliers.

This algorithm is an unsupervised outlier detection method. This paper focuses on this type of algorithm for comparison and analysis. Section 7 gives a comparative study of other algorithms’ performance versus ours.

3 Basic definitions

This section states the network flow problem, and defines the \( k \)-nearest neighbour graph.

3.1 Network flow

Let \( G = (V, E) \) be a directed graph with no self-loops and no parallel edges, and let each edge have a capacity which is a non-negative real number. Let vertices \( s \) and \( t \) be specified; \( s \) is called the source and \( t \) the sink. An edge capacity is represent by \( c(e) \). A flow is a function \( f \) from the edges to the reals satisfying:

1. for every edge \( e \in E, 0 \leq f(e) \leq c(e) \)
2. for every vertex \( v \) except the source and the sink, the flow incoming to \( v \) is equal to the flow outgoing from \( v \).

The maximum flow problem is to find a flow function \( f \) which maximises the total flow, where the total flow is defined as the amount of flow leaving the source, minus the amount entering the source. Let \( X \) be a subset of the vertex set \( V \) and \( \bar{X} \) be the complement of \( X \). If \( s \in X \) and \( t \in \bar{X} \) then \( (X, \bar{X}) \) is called a cut separating \( s \) and \( t \). The capacity of cut \( (X, \bar{X}) \) is the sum of capacities of edges from \( X \) to \( \bar{X} \). The Maximum Flow Minimum Cut theorem states that the maximum amount of flow from \( s \) to \( t \) equals the minimum of the capacities of cuts separating \( s \) and \( t \) (Ford and Fulkerson, 1962).
3.2  k-Nearest neighbour graph

Let $V$ be a set, and a distance function from $V \times V$ to the real numbers be defined. The $k$-nearest neighbour graph on $V$ has as its set of edges those pairs $(v, w)$ of vertices such that $v$ is one of the $k$ vertices (excluding $w$) nearest to $w$ or $w$ is one of the $k$ vertices (excluding $v$) nearest to $v$.

4  Basic ideas underlying this algorithm

A set $C$ of points is given; typically $C$ is the set of points some clustering algorithm saw as the points of one cluster. The basic idea is to separate outliers from the rest of the points. Consider $C$ as the set of vertices of a network: for each pair $(s, s')$ in $C$, consider $ss'$ to be an edge, and to have capacity proportional to the reciprocal of the distance from $s$ to $s'$. Thus if $s$ is near to $s'$, $ss'$ has a high capacity, while if $s$ is far from $s'$, $ss'$ has a low capacity.

The proposed method is based on the following ideas. Suppose that $s$ is an outlier. Let $t$ be the point in $C$ that is farthest from $s$. Suppose further that $s$ is far from all the other points in $C$, then each edge $ss'$ has small capacity. Then the network flow algorithm will tell us that the maximum flow from $s$ to $t$ is small, and quite likely the minimum cut will be $\{(s), C - \{s\}\}$. Alternately, if $t$ is also an outlier, the minimum cut may single out $t$.

Suppose instead that $s$ and another couple of points $s'$ and $s''$ from a group of outlier points and let $t$ be the point in $C$ that is farthest from $s$. Since $s$, $s'$ and $s''$ are very similar to each other, the capacities of edges between them are very high. When the total flow from $s$ to $t$ is smaller than the capacities of $ss'$ and $ss''$, the minimum cut may be $\{(s, s', s''), C - (s, s', s'')\}$ (Liu and Sprague, 2004a,b).

It was the initial intention, when more than one point contained the same coordinates, to capture them simultaneously as potential outliers. The reason for this is twofold. Firstly, it decreases execution time because the procedure will need to run the network flow algorithm fewer times in order to separate out all outliers. Secondly, the user might prefer to analyse similar outliers together.

To avoid having infinity edge capacity (edges between two identical points have length zero), the capacity is defined as $c/(d + 1)$, for some constant $c$ (the precision of the capacity depends on the user). Let us assume $c = 100$. In Figure 1(a), suppose the total flow from source $s$ to its farthest vertex $t$ is 101. Suppose that $t$ is an outlier and the cut is around the sink $t$. Suppose that instead the data file also contains another vertex $t'$ which is identical to sink $t$ as shown in Figure 1(b). The maximum capacity of an edge is 100, so the capacity between $t$ and $t'$ is 100. Those vertices connecting to $t$ also connect to $t'$, including the edge $tt'$. Because the flow from $s$ to $t$ in Figure 1(a) is 101, the flow from $s$ to $t'$ is 101 as well. Edge $tt'$ enables new paths from $s$ to $t$, so, the flow from $s$ to $t'$ (101) will saturate the edge $tt'$ (100) and the cut will be the same as before: the sink side of the cut contains $t$ alone. The total flow increases by 100, becoming 201. Thus, it cannot find the outlier group $t$ and $t'$ together by the minimum cut.

Figure 1 Each edge is bidirectional, drawn as a straight line (see online version for colours)

5  Outlier detection by network flow

5.1  Method overview

In this study, data points are first clustered into a set of clusters. The input to the outlier detection algorithm is the set of data points of one cluster. The algorithm first constructs the $k$-nearest neighbour graph for those data points. Consider the graph to be directed and assign capacities to edges.

Phase 1

1. Select a source $s$ and its farthest vertex as the sink $t$.
2. Find a maximum flow from $s$ to $t$ and a minimum cut separating $s$ and $t$, and use the smaller side as the candidate outlier or outlier group.
Remove the candidate outlier or outlier group from the graph. Repeat Steps 1–3, until the stop criterion is met (see Section 6.3 for the stop condition).

**Phase 2**

Coarsen the original graph and construct the Gomory-Hu Tree (Gomory and Hu, 1961) on the coarse graph.

**Phase 3**

Select outliers from candidate outliers.

**5.2 Choosing the next source**

There are two ways to select the next source. One is to select the next source as the vertex with min (flow_passed + last_wave). Ford and Fulkerson (1962) suggested using augmenting paths to increase the total flow. An augmenting path is a simple path from the source s to the sink t on which flow can be increased. Each vertex could record the flow passing through that vertex in previous augmenting paths and the flow in the last wave which is out of the source but cannot reach the sink. If these two flows are added, each vertex can obtain the flow that can be pushed. High sum value means the vertex strongly connects with other vertices; low sum value means the vertex weakly connects with others. For example, in Figure 2, from s to t, the augmenting path is expanded by breadth first search. From the source vertex s, there are two augmenting paths s → a → c → t with flow 3 and s → a → d → t with flow 3. In the last wave, it finds two paths s → a → d → f → e with flow 4 and s → b → c with flow 10, and there are no more augmenting paths to reach t. Computing the value of flow_passed and last_wave of each vertex, the expression (flow_passed + last_wave) is minimised at f (or e) where its value is 4 + 0 = 4. f or e are the vertices most weakly connected with other vertices on the source side. So, in the next iteration, f (or e) is chosen as the source.

**5.3 Stop condition**

For the stop criterion, there are also two methods:

1. users can specify the number of outliers and outlier groups they want
2. the maximum flow is translated into a distance $d'$. In Formula 2, $\#cross_edge$ is the number of edges going from the source side to the sink side. $\max_{flow}/\#cross_edge$ indicates the average capacity per edge and the $n$th root of the average capacity converts the average capacity back to its original scale. This $d'$ serves as an estimate of the average length of edges in the cut. If $d'$ is less than the average edge length of the original $k$-nearest neighbour network, the procedure stops. The second method is called the autostop criterion.

$$d' = \frac{100}{\sqrt[\#cross_edge]{\max_{flow}/\#cross_edge}} - 1$$  \hspace{1cm} (2)

**6 Experiments**

**6.1 Outlier (group) detection**

The process is illustrated by the data set mentioned in the Chameleon paper (Karypis et al., 1999). In the t4.mat data set, there are 8000 points. hMETIS method is used to cluster points into 20 groups (Figure 3). For the data set t4.mat, hMETIS(‘vcluster’) clusters points into 20 clusters (the command and the parameters are: vcluster t4.mat 20 –clmethod=graph –sim=dist –agglomfrom=30) (see online version for colours)

No. 20 cluster, in Figure 4(a), consists of 591 points. The goal is to find the outliers and outlier groups in this cluster. First the algorithm sets up the $k$-nearest neighbour graph $G$, by connecting each vertex with its seven nearest neighbours (Figure 4(b)). All points are connected, and there are a total of 2514 undirected edges in $G$.

The algorithm uses Formula (1) to set up the capacity. For this graph, $n = 4$ is enough to expose the weak edges. The edges with low capacities are called weak edges. Figure 5 shows the results of Phase 1.
Figure 4  No. 20 cluster (a) the 591 points in the cluster and (b) the seven-nearest neighbours graph \( G \) (see online version for colours)

Figure 5  For No. 20 cluster, 16 iterations of network flow for finding outliers and outlier groups are performed (see online version for colours)

Note: Lines 1–3 indicate the outlier groups having Phase 1 outlier degree under 4, between 4 and 5 and between 5 and 6, respectively.

Phase 2 of the algorithm is to adjust the maximum flow. In Phase 1, due to the order of removing candidate outliers and outlier groups, outliers removed later may have artificially low network flow which would be interpreted as being strong outliers. To solve this problem, each candidate outlier group is coarsened into a new vertex. When the stop condition in Phase 1 is satisfied, the remaining data is also coarsened into a new vertex, which is called the body vertex. Suppose the graph of the original cluster is \( G \), the graph of the coarsened cluster is then referred to as \( G_c \) as shown in Figure 6(b). In \( G_c \), the capacity of the edge between candidate outlier groups \( v_i \) and \( v_j \) is the sum of capacities of edges in \( G \) between the vertices in group \( v_i \) and those in group \( v_j \).

Figure 6  The remaining (after Phase 1) points form a body vertex and each candidate outlier group is a vertex of \( G \), (a) \( G \) after Phase 1 and (b) \( G \) before Phase 2, with 17 vertices and 56 directional edges (see online version for colours)

In Phase 2, a Gomory-Hu tree \( G_3 \) is constructed on the vertex set of \( G \) (Gomory and Hu, 1961). The body vertex (No. 17 vertex in Figure 6(a)) of \( G \) is used as the source and each other vertex \( v_i \) as a sink to run the network flow and construct the Gomory-Hu tree as follows:

1  If \( v_i \) is not contained in the sink side by any other vertices, set up an edge between the body vertex and \( v_i \) directly (e.g. the vertices 5, 9 and 10–16 in Figure 6(b)).

2  If \( v_i \) is contained in other vertices’ minimum cuts, set up an edge between \( v_i \) and its nearest predecessor. The nearest predecessor of \( v_i \) is a vertex of \( G \) whose sink side contains \( v_i \) and (among such vertices) has minimum sink side. For example, in Figure 6(b), vertex 1 is contained by the sink side of vertices 7 and 9. Because the size of the minimum cut from vertex 7 (size 2) is less than the size of the minimum cut from vertex 9 (size 6), vertex 7 is vertex 1’s nearest predecessor.

3  If \( v_i \)’s minimum cut contains \( v_j \) and \( v_j \)’s minimum cut contains \( v_i \), the label ‘\( v_i - v_j \)’ represents the name of this new node in \( G_3 \). (This is a rare case which only happens when the cut determined by \( v_i \) in Phase 2 and the cut determined by \( v_j \) are the same).

Thus, it obtains a tree \( G_3 \) (Figure 7) with all the nodes of the tree being vertices of graph \( G \) and the root being the body vertex. A descendant’s edge capacity is always less than or equal to its predecessor’s edge capacity. \( G_3 \) represents the maximum flow between all pairs of vertices in the graph. The minimal edge capacity between a pair of vertices in the original graph \( G \) is the minimum edge capacity along the connecting path of this pair in the tree \( G_3 \).

Figure 7  \( G_3 \) in Phase 2 (see online version for colours)

Note: The number by each node is the network flow in Phase 2 from vertex 17 to that node.

The maximum flow increases in a multiplicative fashion (logarithmic scale). Therefore, \( \log_{10}( \text{maximum flow} ) \) is used as its outlier degree. Figure 8 gives the experimental results on a different cluster.

6.2 k-nearest neighbours

This section examines the impact of different choices of \( k \) (for the \( k \)-nearest neighbour graph) on the behaviour of the algorithm. If \( k \) is increased, more edges will be generated.
In computing the flow between a given source and sink, there are more edges that need to be saturated. For example, outlier 3 in Figure 5 (which is named vertex \( w \)) has a maximum flow of 1136 when \( k = 7 \), and when \( k = 15 \), \( w \) (which is labelled 8 in Figure 9(a)) has a maximum flow of 1785. When \( k = 7 \), \( w \) connects with 7 or so other vertices and when \( k = 15 \) \( w \) connects with 15 or so other vertices. The top \( n \) outliers remain consistent when the parameter \( k \) is changed. Although the border of outlier group changes when \( k \) changes, the strongest outliers are almost in the same order (Figure 9(c) and (d)).

**Figure 8** No. 19 cluster has 278 vertices and 1185 undirected edges (see online version for colours)

<table>
<thead>
<tr>
<th>Phase 2 order of removal</th>
<th>Maximum flow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertex 3</td>
<td>2988</td>
</tr>
<tr>
<td>Vertex 1</td>
<td>5724</td>
</tr>
<tr>
<td>Vertex 2 and 1</td>
<td>28,261</td>
</tr>
</tbody>
</table>

**Note:** The algorithm ran seven iterations of Phase 1; the autostop criterion was met after three iterations. The table shows the Phase 2 results. In the row 3 of the table, when removing vertex 2, vertex 1 is also removed. Vertex 1 is a descendant of vertex 2 in \( G_x \).

**Figure 9** In (a) and (b), the experiment lets the algorithm run more iterations after the autostop criterion to observe removing outliers (a) Candidate outliers found in No. 20 cluster with \( k = 15 \). The algorithm automatically stops after nine iterations. (b) With \( k = 10 \), the algorithm automatically stops after 13 iterations. (c) Top 30 outliers, with \( k = 15 \) and (d) Top 30 outliers, \( k = 10 \) (see online version for colours)

The choice of \( k \) is principally determined by the user’s preference that the \( k \)-nearest neighbour graph be connected. Fewer nearest neighbours will cause some small groups of points to become disconnected from the rest of the data, even though they may not be outlier groups. This can happen to a small group (of at least \( k + 1 \) points) which are near to the rest of the points in the cluster, but are extremely near to each other. The algorithm wants all the points to be connected, which puts a lower bound on \( k \).

In Figure 10, by using the same method, the algorithm detects outliers for each of the 20 clusters. If a cluster does not contain many loosely connected points, the algorithm quits after a few iterations, such as clusters 4, 5, 8 and 13. If the cluster contains many outliers, the algorithm runs longer.

**Figure 10** Outlier detection for all the 20 clusters (see online version for colours)

**Note:** The experiment chooses four-nearest neighbours for the clusters 16 and 17 and seven-nearest neighbours for all the other clusters. The autostop criterion is used for all the clusters.

**6.3 Phase 3: select outliers**

A subtle issue in Phase 3 is that different users may disagree on how many outliers are there, depending on different intended uses of the data, application-specific requirements and other affecting factors. This study focuses on detecting those data points loosely connected to the main data and output them with their outlier degrees. The user can either specify what percentage of the data should be considered outliers, or a threshold on outlier degrees can be specified to separate outliers from normal data points.

**6.4 Experiments on high dimensional data**

**6.4.1 Boston housing data**

For high dimensional data, this network flow algorithm considers the distance between points on all attributes. This algorithm is a graph-based method. Coordinates of points are not retained by the graph, only distances are retained (via capacities of edges). Therefore, after setting up the graph, dimensionality of the data is not directly felt.

To compare the experimental results with the work of Aggarwal and Yu (2001), the following experiment uses the Boston housing database (UCI KDD Archive) to detect
Outlier detection and evaluation by network flow

Outliers in high dimensional data. Among the 14 attributes, it picked 13, eliminating the single binary attribute (as Aggarwal and Yu also did). Firstly, 506 records were grouped into five clusters by hMETIS (Karypis et al., 1999). The experiment analysed one cluster with 164 records, using the four-nearest neighbour graph. After applying the network flow algorithm for 17 iterations, it found 10 single candidate outliers, 3 outlier groups with 2, 6 and 4 points and 5 subclusters (small clusters) with 34, 21, 24, 15 and 48 points. For those single candidate outliers, some are outliers at one attribute, such as those records with very low (9.8%) or high (96.1%) proportion of owner-occupied units built prior to 1940. For the abnormal patterns detected by their work, outlier detection by network flow also detects the same outlier with low nitric oxide concentration (0.453), high proportion of pre-1940 houses (93.4%) and high index of accessibility to radial highways (8). The latter two attributes with high values usually correspond to high nitric oxide concentration. For the abnormal point with low crime rate (0.04741), modest number of business acres per town (11.93) and low median home price (11,900), it was inside a subcluster with 21 records. Records of this subcluster have attribute values similar to this above abnormal point. The low-modest values of the first two features are usually indicative of high housing price. In each candidate outlier group, attribute values are very close at each dimension. For example, an outlier group with two points has low crime rate (0.04203 and 0.04294), modest number of business acres per town (15.04 and 15.04), and low median home price (22.9 and 20.6). This algorithm can successfully find all outliers detected by Aggarwal and Yu, and it can further find outlier groups similar with detected outliers.

6.4.2 Image segment data

The experiment is conducted on a Corel image database consisting of 9800 images. The image segmentation method is adopted as proposed in Blobworld (Carson et al., 2002): each image is split into a set of semantic regions (called segments). In total the images are split into 82,556 segments. Each image segment is represented by an eight dimensional feature vector (three texture features, three colour features and two shape features).

The 82,556 segments were partitioned into 100 clusters by a genetic algorithm-based clustering method (Zhang and Chen, 2005). Figure 11 displays a particular cluster containing 1108 segments, which are principally green. Outlier detection was performed on the segments in that cluster, using the four-nearest neighbour graph; 147 outliers were detected. Some of these outliers are displayed in Figure 11(b). By outlier detection on the clusters, the image query accuracy is improved by about 5%, compared to the same system without outlier detection (Liu et al., 2006).

6.5 Feature selection versus all features for high dimensional data

Feature selection is a challenging issue for high dimensional data. Aggarwal and Yu proposed an evolutionary algorithm to mine the appropriate $k$ combinations out of a total of $d$ attributes, where $k$ is an input parameter. It was reported that using all features is not effective in detecting outliers, and that it is difficult to identify a few relevant dimensions where the outliers may be observed (Aggarwal and Yu, 2001; Lazarevic and Kumar, 2005). Experiments in this paper on the housing (13 dimensions) and image (8 dimensions) data use all the features. Clustering performs an initial grouping (or selection) of features and outlier detection may be deemed as feature cluster repair. Outliers are those image segments different from the dominant colour which is dark green in this case (Figure 11).

6.6 Using network flow for clustering

With some minor changes, the proposed outlier detection algorithm can be turned into a clustering and outlier detection algorithm. For a given data set, the algorithm

![Figure 11](image1.png)

(a) Normal segments with dark-green colours and (b) outlier segments with red, purple or black colour.
first computes its $k$-nearest neighbour graph and run the network flow algorithm on each connected component (if $k$ cannot connect all the points). If a minimum cut removes more than $x\%$ of the whole data set, apply outlier detection to the removed part as well as to the remaining data. The value of $x$ is determined by the minimum size of a cluster which is specified by the user.

To coarsen the graph, the shortest edges in it are contracted. The process keeps contracting the shortest edges until $r\%$ of the original edges have been eliminated. Here $r$ is a parameter specified by the user. When two vertices are merged, for a common neighbour $v$ (if any) of them, the new edge connecting the merged vertex with $v$ will have a length that is the average of the original edge lengths. Several other coarsening methods are described in Karypis and Kumar (1995). Figure 12 shows an example in which the vertex B is merged to A.

To produce the results shown in Figure 13, the experiment started with the four-nearest neighbour graph, which is a connected graph. (If $k$-nearest neighbours cannot connect all points, outlier detection will work on each connected component). In this experiment, the compression rate $r\%$ is 70%. The labels in Figure 13 are the cluster IDs. Cluster 1 is the remaining main body after outlier detection for the input data set. Clusters 1.4, 1.6, 1.21 and 1.24 are split off in the 4th, 6th, 21st and 24th iterations of the outlier detection, respectively, because they are bigger than 5% ($x\% = 5\%$) of the whole data set. Cluster 1.4.7 is split off in the 7th iteration of the outlier detection on cluster 1.4. The other points in Figure 13, which are not included in any clusters, are outliers and outlier groups. The autostop criterion is further used on each partition of the graph where its size exceeds $x\%$.

Figure 14 shows the top 20 LOF values, that is, the 20 strongest outliers, when MinPts is chosen as 10 and 20. If the value of MinPts is increased further, the top outliers’ positions will move to the bottom-right corner of the data set. The algorithm proposed in this paper is robust with the input parameter $k$-nearest neighbours: results are consistent when $k$ changes (see Section 7.2 Figure 9(c) and (d)).

6.7 Performance

The complexity of network flow is based on the number of vertices ($|V|$) and edges ($|E|$) of the network. The increase of each augmenting path uses the breadth first search method (Edmonds-Karp version of the Ford-Fulkerson algorithm) to find each augmenting path and the complexity is $O(|V||E|)$ (Edmonds and Karp, 1972). If it is a big data set, the running time will be long. To solve this problem, the network could be coarsened before outlier detection to improve the efficiency as discussed in Section 6.6.

7 Unsupervised outlier detection algorithm comparison

This section focuses on comparing the proposed method with three other outlier detection methods.

7.1 Density-based algorithm

Breunig et al. (2000) proposed an outlier detection algorithm based on each point’s neighbourhood density, that is, estimating the density at point $p$ by analysing its $k$-nearest neighbours. By measuring the difference in density between a point and its neighbouring points, this algorithm assigns every point a degree of being an outlier called LOF. If the point is isolated with respect to the surrounding neighbourhood, the LOF value would be high and vice versa. The LOF algorithm is sensitive to the values of the parameter $MinPts$ that the user must supply. Figure 14 displays the top 20 LOF values, that is, the 20 strongest outliers, when $MinPts$ is chosen as 10 and 20. If the value of $MinPts$ is increased further, the top outliers’ positions will move to the bottom-right corner of the data set. The algorithm proposed in this paper is robust with the input parameter $k$-nearest neighbours: results are consistent when $k$ changes (see Section 7.2 Figure 9(c) and (d)).
In contrast to LOF, which smoothes the distance within an object’s \( k \)-nearest neighbourhood, the outlier detection by network flow algorithm strengthens the relationship between data points by increasing the edge capacities a great many times. The advantages are to reinforce the data properties, enforce the data characteristics inside clusters and expose the weak edges between clusters.

### 7.2 Statistical method

A well established statistical technique to detect outliers makes use of Mahalanobis distance (Rocke and Woodruff, 1996). Let \( D \) be a data file, in which each data point is a point in a Cartesian \( p \)-dimensional space. The Mahalanobis distance of a point \( x \) equals \( \sqrt{(x-\mu)^T C^{-1} (x-\mu)} \) where \( \mu \) is the centroid of the data points and \( C \) is the sample covariance matrix. Points whose Mahalanobis distance exceeds some preselected value are considered outliers. The Mahalanobis distance of a point \( x \) is analogous to the number of standard deviations \( x \) is from the mean.

Figure 15 displays the result of applying this technique on the No. 20 cluster in Figure 4. The points outside of the ellipse graphed have Mahalanobis distance greater than three and may be considered outliers. Changing the cutoff value changes the size of the ellipse, but its orientation and centroid remain the same. Although the size of the ellipse changes, it will keep some density connected outlier groups inside the ellipse.

**Figure 15** The points outside of the ellipse graphed have Mahalanobis distance greater than three (see online version for colours)

Mahalanobis distance is not a robust method, however, because means and covariances, on which it is based, can be greatly influenced by a few strong outliers. A more robust analogue of Mahalanobis distance, called the minimum volume ellipsoid estimator, is presented in Rousseeuw and Zomeren (1990). Like the Mahalanobis distance estimator, it will consider all points outside some ellipse to be outliers.

### 7.3 Distribution-based algorithm

Section 6.4.1 already compared the results generated on Boston housing data with that of Aggarwal and Yu. For the projected \( k \) dimensional attributes, the algorithm divides each attribute into \( s \) equi-depth ranges, so there will be a total of \( s^k \) bins and the average number of data points in a bin is \( u = n/s^k \), where \( n \) is the total number of data points. Treating the distribution of the \( n \) points to bins, with the probability of a point landing in a particular fixed bin being \( 1/s^k \), the standard deviation of the number of points landing in the bin (according to the binomial distribution) is \( \sigma = \sqrt{Ns^k(1-s^{-k})} \). Bins containing fewer than \( u - 3\sigma \) points are treated as outliers.

The method proposed by Aggarwal and Yu is not suitable for this low dimensional data. In the two-dimensional data, there will be \( s^2 \) bins; each row of bins has the same number of data points and likewise for each column of bins. Figure 16(a) shows the results when \( s = 5 \). The average number of points per bin is \( u = 591/25 = 23.64 \) and the standard deviation \( \sigma \) is 4.76, consequently, bins containing fewer than 9.3 points are considered outlying. In Figure 16(a), the outlying bins are shaded. A total of 35 data points are considered outliers. The results of this method, using five intervals in each attributes, are not very sensitive to the geometry of the data points in this data set.

If the experiment switches to using four intervals (or 3 or 2) per attribute, the results will be even less sensitive to the geometry of the data. Results of using six intervals in each attribute are shown in Figure 16(b). A total of 5 bins are outlying and a total of 8 points are labelled as an outliers. Figure 16(c) shows the results of using seven intervals in each attribute. Bins with less than 1.7 points (\( u - 3\sigma \)) are outlying and a total of four data points are labelled as outlying.

**Figure 16** Equal depth intervals for the No. 20 cluster (a) five intervals, (b) six intervals and (c) seven intervals (see online version for colours)

*Note: Bins with cross lines are outlier bins.*

### 8 Conclusions

This paper has proposed a novel outlier detection and evaluation method which is based on network flow. This network flow outlier detection algorithm is a density-based algorithm and is based on the \( k \)-nearest neighbour graph. The network model represents and magnifies data properties. By running the Maximum-Flow/Minimum-Cut process, this algorithm detects outliers/outlier groups and
gives every candidate outlier (group) an outlier degree. This network model is also effective for high dimensional data. This algorithm detects outliers in groups, which is very useful for an analyst of the outlier points.

Future work includes efficiently setting up the network. It may be useful to compress the data before setting up the nearest neighbour graph, instead of after. Another interesting challenge is extending the method of this paper to categorical data. To set up the network when some of the attributes are categorical, the concept of ‘link’ found in ROCK (Guha et al., 2000) may be useful. However, that may lead to too many edges, so decreasing the number of edges is an interesting problem. The outlier detection by network flow algorithm works for very high dimensional data, but this paper does not discuss data properties in higher dimensions, like hundreds of dimensions. Future work also includes further exploration of network flow, especially casting non-obvious problems to this model.

Acknowledgements

We thank Chengcui Zhang for numerous helpful comments and suggestions on this work and paper. We also thank Xin Chen for image data set testing and making Figure 11.

References


