動的なデータ集合に対応した対話的外れ値分析手法

坂詰　知完†　　北川　博之††　　天笠　俊之††

† 筑波大学システム情報工学研究科　〒 305-8573 茨城県つくば市天王台 1-1-1
†† 筑波大学計算科学研究センター　〒 305-8577 茨城県つくば市天王台 1-1-1
E-mail: †sakazume@kde.cs.tsukuba.ac.jp, ††{kitagawa,amagasa}@cs.tsukuba.ac.jp

あらまし　データマイニングにおいてデータ集合の中で他と大きく異なる値をもつ外れ値を検出することは重要なタスクとなっている．代表的な外れ値検出手法の一つとして距離に基づく手法があるが，ユーザが求める外れ値を検出するための適切なパラメタの選択が容易でないことが知られている．この問題に対し，ユーザが求める外れ値の検出に適切なパラメタ選択を支援する対話的な外れ値分析手法 ONION が提案されている．ONION では対象データ集合を事前に分析し索引構造を構築することで，種々の対話的分析を支援する．しかし，索引構造の更新については考慮されていなく，データの追加や削除，更新が行われる場合は効率的な更新が不可欠である．本研究では ONION の索引構造に加えて，グリッド索引とカウンタを組合せて用いることで，動的なデータ集合に対応した対話的外れ値分析手法を提案し，実験によりその有用性を評価する．

キーワード　外れ値分析，ONION，動的データ，索引構造

1. はじめに

データマイニングにおいてデータ集合の中で他と大きく異なる特徴や値をもつデータを検出することは重要なタスクとなっている．Hawkins は外れ値を “異なるメカニズムで生成された疑いを喚起するような他の観測値から大きく外れた観測値” としている [1]．外れ値検出は多くのアプリケーションで用いられており，例えばクレジットカードの不正使用検出や科学データに対する測定誤差の識別，監視ビデオでの異常行動検出などがあげられる．

現在まで様々な外れ値検出手法が提案されているが，代表的な手法として距離に基づく手法 [2] [3] [4] や最近傍に基づく手法 [5]，密度に基づく手法 [6]，クラスタリンクに基づく手法 [7] [8] [9]，統計に基づく手法 [10] [11]，角度に基づく手法 [12] [13] がある．これらの外れ値検出手法の中でも古典的で代表的な手法は距離に基づく手法 [2] である．しかし，適切な外れ値の検出をするためには適切なパラメタ設定が必要になる．この問題を解決するために，L.Cao らは ONION [14] を提案した．ONION は対話的な外れ値分析を提案しており，ユーザが求める外れ値を検出するための適切なパラメタを示す．しかし，ONION は静的なデータ集合を対象としており，動的なデータ集合には対応していない．近年の IoT やセンサ，ソーシャルメディアの発達に伴い，動的なデータ集合が増加している．データ集合が変化することによって外れ値も変化するため，外れ値検出にもリアルタイム性が求められている．

そこで本研究では，動的なデータ集合に対応した対話的外れ値分析手法を提案する．データ集合の更新が行われた場合，ONION の索引構造を更新するための計算を素朴な方法で行うと多くの時間がかかってしまう．そのため，ONION を動的なデータ集合に適用させるためには ONION の索引構造を効率的に更新する必要がある．実験では提案手法に対して，素朴な方法含めた 2 種類の手法と比較し，提案手法の有用性を評価する．

本論文の構成は以下の通りである．2. 章で関連研究について説明し，3. 章では距離に基づく外れ値検出手法とその問題点について述べる．4. 章では ONION の概要を説明し，5. 章で動的なデータ集合での外れ値検出の問題点とそれを ONION に適用したときの問題点について述べる．6. 章では提案手法の詳細を説明し，7. 章で評価実験について述べる．最後に 8. 章で本論文をまとめる．

2. 関連研究

現在，様々な外れ値検出方法が提案されているが，最も古典的な外れ値検出手法の 1 つとして距離に基づく手法 [2] がある．距離に基づく外れ値検出 [2] は，対象のオブジェクトを中心に半径 r の円内にあるオブジェクト数が k 個未満のとき，対象のオブジェクトを外れ値とする手法である．最近傍に基づく外れ値検出 [5] では，対象のオブジェクトと k 近傍のオブジェクト間の距離を計算し，その距離に基づいてオブジェクトがランク付けされる．ランキング上位 n 個のオブジェクトが外れ値となる．密度に基づく外れ値検出 [6] では，外れ値度を示す局所外れ値因子 (LOF) を用いる．LOF 値は各オブジェクトの k 近傍オブジェクトとの密度差によって決まり，LOF 値が高いオブジェクトを外れ値とする．

また，近年の動的データやストリームデータの増加に伴い，それらに対する外れ値検出手法が研究されている．Angiulli らは距離に基づく外れ値検出手法をストリームデータに拡張した手法 [15] を提案した．この手法では単位時間に 1 つの値が到着するストリームデータを想定し，一定期間内に到着した値の集合に対して外れ値検出を行う．石田らは時系列データストリーム上の連続した外れ値検出手法 [16] を提案した．この手法では，現在のデータ集合は直前の時刻の集合が変化したものであり，その変化があまり大きくない場合が多いことを利用した差分計

算法を用いる．差分計算法では，直前時刻の集合における外れ値検出結果を利用して差分計算を行い，距離に基づく外れ値検出手法により現在の集合における外れ値を検出する．

本研究では，動的なデータ集合に対して ONION の索引構造を用いた対話的な外れ値検出を行う．

3. 距離に基づく外れ値検出手法

本章では，本研究の基本となる外れ値検出手法での外れ値の定義とその問題点について説明する．

3.1 定　　義

本研究では，外れ値の定義として Knorr らが提案した距離に基づく外れ値検出 [2] によるものを用いる．Knorr らは外れ値の定義を “対象のオブジェクトを中心に半径 r の円内にあるオブジェクト数が k 個未満のとき，対象のオブジェクトを外れ値とする” としている．図 1 は距離に基づく外れ値検出の例を示しており，左右の図は同じデータ集合である．k=3 ではオブジェクト A は正常値になり，オブジェクト B は外れ値になる．

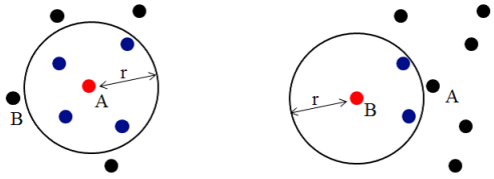

図 1　距離に基づく外れ値検出

3.2 適切なパラメタ設定の難しさ

距離に基づく外れ値検出では各オブジェクトが外れ値であるか，または正常値であるかがパラメタ k と r によって決定する．そのため，適切な外れ値を検出するためには適切なパラメタ選択が重要となる．パラメタによっては正常値とみなされるべきオブジェクトが外れ値として検出されることや，その逆も起こり得る．図 2，3，4，5 はパラメタ k と r の設定によって検出される外れ値がどのように変化するかを示している．図 2 はオブジェクトの分布を示しており，ユーザが外れ値として検出したいオブジェクトをオブジェクト A と B とする．k=3，r=5 の場合，図 3 で示されるようにオブジェクト A と B に加えて，オブジェクト C，D，E，F も外れ値として検出されてしまう．k=2，r=5 の場合，図 4 で示されるようにオブジェクト A と B のどちらも外れ値として検出されない．k=3，r=10 の場合，図 5 に示されるようにオブジェクト A と B に加えて，オブジェクト E が外れ値として検出される．ユーザの意図と一致し，ユーザが検出したいオブジェクトに類似した特徴をもつオブジェクトを検出するためには，適切なパラメタを選ぶことが重要である．

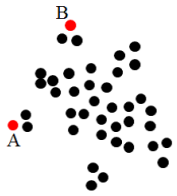

図 2　オブジェクトの分布

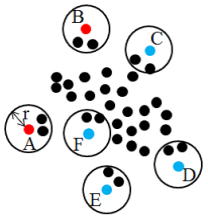

図 3　k=3，r=5 の場合

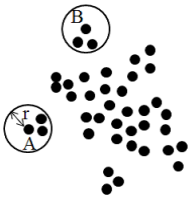

図 4　k=2 と r=5 の場合

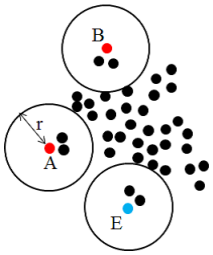

図 5　k=3 と r=10 の場合

4. ONION

本章では，はじめに本研究で用いる ONION [14] の概要について説明する．その後，本研究で対象とする ONION の索引構造についての詳細を説明する．

4.1 概　　要

ONION は L.Cao らが提案したもので，距離に基づく外れ値検出手法において適切なパラメタ選択を行うことが難しいという問題に対して，事前に対象のデータを索引付けすることで，パラメタを変化させたときに検出される外れ値がどのように変化するか，指定したオブジェクト集合を外れ値とするパラメタ範囲はどのような範囲か，等の対話的な外れ値分析を支援する．

ONION での外れ値分析はパラメタ k を $[k_{min}, k_{max}]$，パラメタ r を $[r_{min}, r_{max}]$ の範囲で考え，ユーザが求める外れ値を検出するための適切なパラメタを示す．k_{min} と r_{min} はそれぞれの下限，k_{max} と r_{max} はそれぞれの上限である．各オブジェクトは必ず正常値 (IL)，必ず外れ値 (OL)，外れ値候補 (OC) のいずれかに分類される．IL のオブジェクトは $k \in [k_{min}, k_{max}]$ と $r \in [r_{min}, r_{max}]$ の範囲内で必ず正常値であり，一方で OL のオブジェクトはその範囲内で必ず外れ値である．OC はパラメタによって正常値または外れ値になる．本論文では，オブジェクト O と O の k 番目の近傍オブジェクト間の距離を D_O^k と示す．もし，$D_O^{k_{max}} \leqq r_{min}$ ならばオブジェクト O は IL であり，$D_O^{k_{min}} > r_{max}$ ならばオブジェクト O は OL である．どちらの条件にも一致しないオブジェクトは OC である．

ONION は 4.2 節で説明される索引構造を用いることによって，$k \in [k_{min}, k_{max}]$ と $r \in [r_{min}, r_{max}]$ の各選択 (k, r) に対して OC のどのオブジェクトが正常値または外れ値であるかを効率よく識別する．ONION は OC に対して comparative outlier analytics (CO) と outlier-centric parameter space exploration (PSE)，outlier detection (OD) の 3 種類の外れ値分析を提案している．CO はユーザが外れ値として検出したいオブジェクト集合 O_{in} を入力とし，O_{in} が外れ値であるときに外れ値になる他のオブジェクト集合を識別する．PSE はユーザが外れ値として検出したいオブジェクト集合 O_{in} を入力とし，それに近い外れ値集合とパラメタ集合のペア (k, r) を見つける．OD はパラメタのペア (k, r) を入力とし，そのとき外れ値とな

ID	座標
1	(x_1, y_1)
2	(x_2, y_2)
⋮	⋮
N	(x_N, y_N)

図 6　オブジェクト表

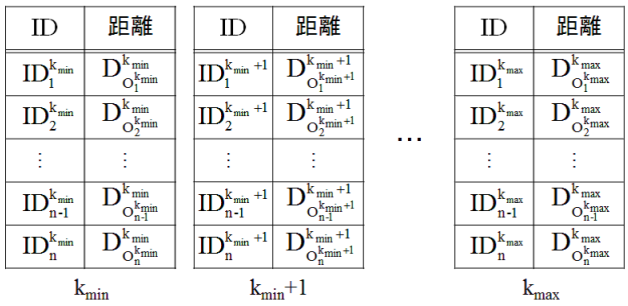

図 7　P-Space

るオブジェクトを検出する．また，OD では NULL をパラメタとして与えることができ，その場合は全ての OL を出力する．

4.2　P-Space

OC に対する上記の対話的分析をサポートするために，ONION は ONION 空間 (O-Space) とパラメタ空間 (P-Space)，データ空間 (D-Space) と呼ばれる 3 種類の索引構造を提案している．各索引構造は OC のオブジェクトを分析することが主な目的であるため，OC のみで構成されている．O-Space は各オブジェクト $O \in OC$ から $k_i(k_{min} \leqq k_i \leqq k_{max})$ 番目に近いオブジェクトまでの距離 $D_O^{k_i}$ を計算し，OC ごとに k_{max}-k_{min}+1 個の距離の値を保持する．P-Space は k_{max}-k_{min}+1 個のテーブルをもつ．i 番目のテーブルは全ての OC からの距離 $D_O^{k_i}$ を昇順で保持する．D-Space はオブジェクト間の支配関係に基づいで構築される．データ集合の中で，外れ値候補 O_i は他のオブジェクト O_j より外れ値度が大きい場合がある．すなわち，任意のパラメタに対して，O_i は O_j が外れ値であるときは常に外れ値になる．この関係を支配関係と呼ぶ．D-Space は全ての OC を複数の支配グループに分割する．同じ支配グループに属するオブジェクトは支配関係をもつ．

OC に対して O-Space，P-Space，D-Space のいずれも上記の対話的分析が可能であるが，[14] で P-Space は O-Space より良い性能を示しており，動的なデータ集合に対する D-Space の更新は難しいということから，本研究では P-Space を用いる．データ集合の全てのオブジェクトはオブジェクト表に登録され，図 6 で示されるように ID が与えられる．(簡略化のために 2 次元のオブジェクトを例に用いているが，高次元であっても同様である．)P-Space は図 7 に示されるように k_{max}-k_{min}+1 個のテーブルから成る．i 番目のテーブルには OC のオブジェクト O_j に対するペア $(ID_j, D_{O_j}^{k_i})$ が保持されており，$D_{O_j}^{k_i}$ の昇順で並んでいる．

4.3　P-Space を用いた対話的外れ値分析

P-Space は上記に示した全ての対話的分析の効率的な処理

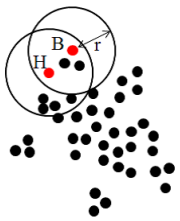

図 8　オブジェクトの追加

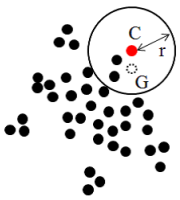

図 9　オブジェクトの削除

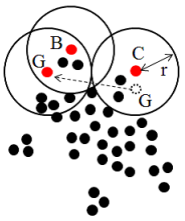

図 10　オブジェクトの更新

をサポートする．ここでは例として outlier-centric parameter space exploration (PSE) を用いた対話的外れ値分析を説明する．上記で示したように，最初にユーザが外れ値として検出したいオブジェクト集合 O_{in} を指定する．ONION は P-Space を用いて O_{in} に近いオブジェクト集合を外れ値として検出するためのパラメータを示す．近さは $\delta(-1 < \delta < 1)$ によって決まる．もし，検出された外れ値の数をユーザが多いと判断したら，$\delta \leqq 0$ で $|O_j| = (1+\delta)|O_{in}|$ となるパラメタ k と r を示す．このとき $O_j \subseteqq O_{in}$ である．もし，外れ値の数を少ないと判断したら，$\delta \geqq 0$ で $|O_j| = (1+\delta)|O_{in}|$ となるパラメタ k と r を示す．このとき $O_j \supseteqq O_{in}$ である．

5.　動的データ集合での外れ値分析

データ集合が動的な場合，オブジェクトの追加や削除，更新が起こり得る．データ集合の変化により，新しく追加または更新されたオブジェクトに加えて，既存オブジェクトの正常値と外れ値の判定が変化する可能性がある．本章では，例を用いてオブジェクトの追加，削除，更新が行われた場合のオブジェクトの正常値と外れ値の判定の変化を示す．なお，ここでは k=3 とした場合を想定する．

5.1　データ集合の変化

図 8 ではオブジェクト H が追加された場合を示している．オブジェクト B を中心とした半径 r の円内にあるオブジェクト数は 2 つであるため，オブジェクト B は外れ値である．しかし，オブジェクトの追加によって，オブジェクト B は正常値になる．オブジェクト H は正常値である．図 9 ではオブジェクト G が削除された場合を示している．オブジェクト G が削除される以前，オブジェクト C は正常値である．しかし，オブジェクト G が削除されることによって，オブジェクト C は外れ値になる．図 10 ではオブジェクト G が更新される場合を示している．もし，オブジェクト G が更新によって移動した場合，オブジェクト B は外れ値から正常値に変わり，オブジェクト C は正常値から外れ値に変わる．オブジェクト G は正常値である．

5.2 動的環境における ONION の問題点

ONION は与えられたデータ集合を分析し，初めにオブジェクトを IL，OL，OC のいずれかに分類する．データ集合が動的である場合，正常値と外れ値の判別は上記のようにデータ集合の更新に応じて変化し，IL や OL，OC へのオブジェクトの分類が変わる．例えば，新しいオブジェクトが 1 つ追加されたときは OL のオブジェクトが OC になり，OC のオブジェクトが IL になる可能性がある．同様に既存のオブジェクトが 1 つ削除されたときは IL のオブジェクトが OC になり，OC のオブジェクトが OL になる可能性がある．オブジェクトの更新はオブジェクトの追加と削除の組合せであるため，上記の全ての変化が起こり得る．また，複数のオブジェクトを同時に追加，削除，更新した場合，OL のオブジェクトが IL に，IL のオブジェクトが OL になる可能性もある．本論文では，このような変化を状態変化と呼ぶ．

動的なデータ集合で ONION のような対話的分析を可能にするには，P-Space を継続的に更新する必要がある．素朴な方法で P-Space の再計算を行う場合，多くの距離計算とソートが必要になるため，更新に多くの時間がかかってしまう．P-Space の更新をするためには 2 つのステップがある．1 つ目がデータ集合の更新によって状態変化があるオブジェクトを見つけることである．つまり，OC ではなくなるオブジェクトや新しく OC になるオブジェクトを調べる必要がある．2 つ目は現在の OC から成る P-Space の更新である．次の章ではこれらの 2 つのステップを効率的に行う手法を提案する．

6. 提 案 手 法

提案する手法は以下の 2 つのステップから成る．(1) 状態 (IL, OL，OC) が変化するオブジェクトを検出することと，(2) データの更新による P-Space の更新を行うことである．これらを効率的に行うために，本研究ではグリッド索引と拡張したオブジェクト表を用いる．素朴な方法はオブジェクトの状態変化を検出するために全てのオブジェクトのペアで距離の再計算をしなければならない．グリッド索引と拡張したオブジェクト表は P-Space を効率的に更新することを可能にする．

本研究では 1 つのオブジェクトまたは複数のオブジェクトの追加と削除，更新を対象とする．P-Space に影響を与える状態変化は追加の場合，OL から OC と，OC から IL である．削除の場合は IL から OC と，OC から OL である．更新は追加と削除の組合せであるため，上記の全ての状態変化が影響を与える．

6.1 グリッド索引とカウンタをもつ拡張したオブジェクト表

グリッド索引は図 11 で示されるようにオブジェクト空間を分割し，各グリッドは図 12 のようにオブジェクト数とそのオブジェクト ID を保持する．また，オブジェクト表は図 13 で示されるように r_{min} カウンタと r_{max} カウンタ，オブジェクトの状態が加わることによって拡張される．r_{min} カウンタと r_{max} カウンタは，それぞれ r_{min} と r_{max} 内にある近傍オブジェクト数を表している．カウンタを保持することによって，データ集合の更新の際に必要になる計算を減らすことができる．種別の欄では各オブジェクトの状態 (IL，OL，OC) を保持する．

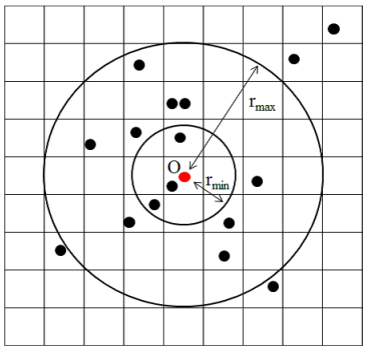

図 11 グリッド索引

								(1,{17})
			(1,{1})				(1,{6})	
				(2,{7,4})				
		(1,{11})	(1,{13})	(1,{8})				
				(2,{2,9})		(1,{12})		
			(2,{3,5})		(1,{16})			
	(1,{15})				(1,{14})			
						(1,{10})		

図 12 グリッド索引の内容

ID	座標	r_{min} カウンタ	r_{max} カウンタ	種別
1	(5, 10)	1	7	OC
2	(6, 6)	3	12	IL
⋮	⋮	⋮	⋮	⋮
17	(11, 12)	1	1	OL

図 13 拡張したオブジェクト表

新しいオブジェクトが追加された場合，そのオブジェクトを中心に半径 r_{min} と r_{max} の円内にあるオブジェクト数を計算する必要がある．グリッド索引を用いることで，円内に含まれるグリッドと円に重なっているグリッドに存在するオブジェクトのみに注目すればよい．その計算に基づいて，新しく追加されたオブジェクトは IL または OL，OC のいずれかに分類される．もし，IL か OL ならば P-Space で保持する距離の値を計算する必要はない．また，オブジェクトが追加されることによって既存オブジェクトの状態変化を調べる必要がある．拡張されたオブジェクト表で保持するカウンタは，状態変化を調べるための距離計算を減らす役割をもつ．状態変化の有無を確認しなければならないオブジェクトは追加されたオブジェクトから距離 r_{max} 内にあるものである．そのオブジェクトの r_{min} カウンタと r_{max} カウンタは更新しなければならない．その後，OL から OC と，OC から IL に状態変化したオブジェクトに注目し，P-Space の更新を行う．

6.2 状態変化を検出するアルゴリズム

状態変化を検出するアルゴリズムをアルゴリズム 1 に示す．状態変化の有無を調べるために追加または削除されたオブジェクトを中心に半径 r_{min} と r_{max} の円内にあるオブジェクトのカウンタを更新しなければならない．オブジェクトの更新の場合は，移動前のオブジェクト位置でオブジェクトが削除され，移

Algorithm 1 状態変化アルゴリズム

for each $G_{r_{min}}$ **do**
  r_{min}，r_{max} カウンタと状態変化を更新する．
end for
for each $G_{or_{min}}$ **do**
  for each $G_{or_{min}}$ にあるオブジェクト **do**
    距離 ($dist_{min}$) を計算する．
    if $dist_{min} \leqq r_{min}$ **then**
      r_{min}，r_{max} カウンタと状態変化を更新する．
    else
      r_{max} カウンタと状態変化を更新する．
    end if
  end for
end for
for each $G_{r_{max}}$ **do**
  r_{max} カウンタと状態変化を更新する．
end for
for each $G_{or_{max}}$ **do**
  for each $G_{or_{max}}$ にあるオブジェクト **do**
    距離 ($dist_{max}$) を計算する．
    if $dist_{max} \leqq r_{max}$ **then**
      r_{max} カウンタと状態変化を更新する．
    end if
  end for
end for

動後のオブジェクト位置にオブジェクトが追加されたとみなす．半径 r_{min} の円に完全に含まれるグリッドを $G_{r_{min}}$，重なっているグリッドを $G_{or_{min}}$ とする．$G_{r_{max}}$ と $G_{or_{max}}$ も同様に定義する．オブジェクトを 1 つ追加する場合はカウンタが 1 つ増加する．削除の場合はカウンタが 1 つ減少する．複数のオブジェクトが同じグリッドに追加や削除された場合は，そのオブジェクト数分だけカウンタを更新する．アルゴリズム 1 は以下の 4 つのフェーズに分かれる．

Phase 1: グリッド $G_{r_{min}}$ にあるオブジェクトの r_{min} と r_{max} カウンタを更新する．

Phase 2: グリッド $G_{or_{min}}$ にあるオブジェクトと，追加または削除されたオブジェクト間の距離 $dist_{min}$ を計算する．もし，その距離が r_{min} 以下ならば，r_{min} と r_{max} カウンタを更新する．r_{min} より大きく，r_{max} 以下ならば，r_{max} カウンタのみ更新する．

Phase 3: Phase 2 までにチェックしたグリッドを除いてグリッド $G_{r_{max}}$ にあるオブジェクトの r_{max} カウンタを更新する．

Phase 4: グリッド $G_{or_{max}}$ にあるオブジェクトと，追加または削除されたオブジェクト間の距離 $dist_{max}$ を計算する．もし，その距離が r_{max} 以下ならば，r_{max} カウンタを更新する．

1 つのオブジェクトを追加したことにより，上記の処理で r_{min} カウンタが k_{max} になったオブジェクトは OC から IL に変化する．もし，r_{max} カウンタが k_{min} になった場合，オブジェクトは OL から OC に変化する．1 つのオブジェクトを削除したことにより，r_{min} カウンタが k_{max}-1 になったオブジェクトは IL から OC に変化する．もし，r_{max} カウンタが k_{min}-1 になった場合，オブジェクトは OC から OL に変化する．複数のオブジェクトを同時に追加や削除，更新した場合は，OL から IL または IL から OL になる可能性もある．

Algorithm 2 P-Space の更新アルゴリズム

for each OC ではなくなったオブジェクト O_n **do**
  for $i = 1$ to $k_{max} - k_{min} + 1$ **do**
    P-Space から O_n に関する情報を削除する．
  end for
end for
if 新しいオブジェクトが追加される **then**
  for each OC のままだったオブジェクト O **do**
    距離 ($dist_{new}$) を計算する．
    if $dist_{ins_{min}} \leqq dist_{new} \leqq dist_{ins_{max}}$ **then**
      P-Space の O の情報を更新する．
    else if $dist_{new} < dist_{ins_{min}}$ **then**
      P-Space の O の情報を更新する．
    end if
  end for
end if
if 既存オブジェクトが削除される **then**
  for each OC のままだったオブジェクト O **do**
    距離 ($dist_{del}$) を計算する．
    if $dist_{del} \leqq dist_{del_{max}}$ **then**
      P-Space の O の情報を更新する．
    end if
  end for
end if
for each 新しく OC になったオブジェクト **do**
  i 番目の近傍距離を計算する．
  for $i = 1$ to $k_{max} - k_{min} + 1$ **do**
    P-Space に距離を加える．
  end for
end for

6.3 P-Space の更新アルゴリズム

P-Space を更新するアルゴリズムをアルゴリズム 2 に示す．OC ではなくなったオブジェクトに関する情報は P-Space から削除し，OC になったオブジェクトに関する情報は P-Space に加える必要がある．さらに，OC のままだったオブジェクトの k 近傍が変化していないかを調べなければならない．アルゴリズム 2 は 4 つのフェーズに分かれる．

Phase 1: OC から IL と，OC から OL に状態変化したオブジェクトに関する情報を P-Space から削除する．

Phase 2: このフェーズは新しいオブジェクト O_{new} が追加されたときに実行され，OC のままだったオブジェクト O の情報を更新する．O と O_{new} 間の距離 $dist_{new}$ は P-Space で保持している O と O の i 番目 (i $\in [k_{min}, k_{max}]$) の近傍オブジェクト間の距離に影響があるかもしれない．もし，$dist_{new}$ が O と O の k_{min} 番目の近傍オブジェクト間の距離 $dist_{ins_{min}}$ 以上かつ O と O の k_{max} 番目の近傍オブジェクト間の距離 $dist_{ins_{max}}$ 以下ならば，$dist_{new}$ は O の k 番目の近傍距離として P-Space に格納される．O_{new} が追加される前の O の i ($k \leqq i \leqq k_{max}$) 番目の近傍距離は O の新しい (i+1) 番目の近傍距離になる．も

し，$dist_{new}$ が $dist_{ins_{min}}$ より小さいならば，O の新しい k_{min} 番目の近傍オブジェクトを調べる必要がある．この場合，グリッド索引を用いて O を含むグリッドから徐々に検索範囲を拡大させることで，k_{min} 番目の近傍オブジェクトを探す．

Phase 3: このフェーズはオブジェクト O_{del} が削除されたときに実行され，OC のままだったオブジェクト O の情報を更新する．もし，O と O_{del} 間の距離 $dist_{del}$ が O と O の k_{max} 番目の近傍オブジェクト間の距離 $dist_{del_{max}}$ 以下ならば，O の (k_{max}+1) 番目の近傍オブジェクトを調べる必要がある．そのオブジェクトは新しい k_{max} 番目の近傍オブジェクトになる．その後，O と O の (k_{max}+1) 番目の近傍オブジェクト間の距離は O の新しい k_{max} 番目の近傍距離となり，O の他の近傍距離も更新される．

Phase 4: 新しく追加されたオブジェクト O_{new} が OC になった場合と，既存オブジェクトの状態が OL から OC または，IL から OC に変化したとき，それらのオブジェクトの i ($k_{min} \leqq$ i $\leqq k_{max}$) 番目の近傍距離を計算し，P-Space に距離を加える必要がある．グリッド索引はこの計算にも活用される．

7. 評価実験

7.1 実験概要

本実験はオブジェクトの追加，削除，更新をした場合の P-Space の更新時間をそれぞれ計測する．実験 1 ～ 3 は人工データを用いる．実験 1 では k_{max}，実験 2 ではグリッドの対角線の長さをそれぞれ変化させて比較する．実験 3 ではバッチ処理の有無による P-Space の更新時間を比較する．実験 4 では実データを用いて k_{max} を変化させて比較する．

バッチ処理により，カウンタ更新と P-Space 更新の効率化が可能になる．同じグリッドでオブジェクトが追加または削除された場合はカウンタ更新をオブジェクト数分だけ 1 度行えばよい．また，複数のオブジェクトの追加または削除により OC ではなくなる場合，毎回 P-Space で保持する値を更新せず，P-Space からそのオブジェクトに関する情報を削除すればよい．

全ての実験で示す更新時間は 3 回試行した平均である．また，実験は Intel Core i7 3.70GHz の CPU と 64GB のメモリをもつ Windows 8 で行い，MATLAB で実装した．

7.2 比較手法

本実験では提案手法と他の 2 つの手法を比較する．1 つ目はグリッド索引のみを用い，オブジェクト表では r_{min} と r_{max} カウンタをもたない手法である．もし，オブジェクトの追加や削除，更新をした場合，再び既存オブジェクトから距離 r_{min} と r_{max} 内にあるオブジェクトの数を調べなければならない．2 つ目はグリッド索引とオブジェクト表の r_{min} と r_{max} カウンタをもたない手法である．この手法では全てのオブジェクトのペアで距離計算を行い，毎回 P-Space を再計算する必要がある．

7.3 実験データ

実験 1 ～ 3 は平均 0，標準偏差 250 のガウス分布と範囲 [-2500, 2500] の一様分布を組合せた 2 次元の人工データを用いる．オブジェクト数は 100000 である．実験 4 では UCI machine learning repository から 3D Road Network (North Jutland, Denmark) Data Set を用いる．これは緯度，経度，標高を表す 3 次元のデータであり，オブジェクト数は 434874 である．

7.4 実験結果

7.4.1 実　験　1

実験 1 の結果を図 14 ～ 18 に示す．各パラメタは k_{min}=15，r_{min}=50，r_{max}=150 とし，k_{max} を 20 から 30 まで 2 ずつ増やす．グリッドの対角線の長さは $k_{min}/2$ とする．データ集合を更新する前の各状態のオブジェクト数は k_{max}=20 で (IL, OC, OL)=(97231, 966, 1803)，k_{max}=30 で (IL, OC, OL)=(96749, 1448, 1803) となっており，k_{max} が大きくなるほど OC の数が増加し，IL の数が減少する．

図 14 は IL，OC，OL に分類される各オブジェクトをそれぞれの手法で追加した場合，図 15 は IL，OC，OL に分類されている各オブジェクトをそれぞれの手法で削除した場合を示す．オブジェクトを追加した場合は，OL と OC のオブジェクトが状態変化するかを確認しなければならない．OL と OC のオブジェクトは少ないため，IL のオブジェクトを追加した場合は提案手法とグリッド索引のみを用いる手法の差は小さい．一方で，オブジェクトを削除した場合は IL と OC のオブジェクトが状態変化するかを確認しなければならない．IL のオブジェクトが多いため，IL に分類されているオブジェクトを削除した場合は提案手法とグリッド索引のみを用いる手法の差は大きくなる．

OC のオブジェクトを追加した場合は IL のオブジェクトを追加した場合と比べて，多くの OC のオブジェクトに対して状態変化するかを確認しなければならない．そのため，グリッド索引のみを用いる手法はより多くの時間がかかる．一方で，OC のオブジェクトを削除した場合は IL に分類されているオブジェクトを削除した場合と比べて，状態変化を確認しなければならない IL のオブジェクト数は減少するが，OC のオブジェクト数が増加する．グリッド索引のみを用いる手法で OC のオブジェクトの状態変化を調べるためには，OC のオブジェクトから半径 r_{max} 内のオブジェクト数を確認する必要がある．IL のオブジェクトの状態変化の確認は半径 r_{min} 内を確認すればよいため，グリッド索引のみを用いる手法は多くの更新時間がかかる．

OL のオブジェクトを追加または削除した場合は，状態変化を確認しなければならないオブジェクト数が少ないため，提案手法とグリッド索引のみを用いる手法の差は小さい．

図 16 は IL に分類されているオブジェクト，図 17 は OC に分類されているオブジェクト，図 18 は OL に分類されているオブジェクトをそれぞれの手法で更新した場合を示す．例えば，IL → OC は IL に分類されているオブジェクトが OC に分類される位置に移動した場合を示している．IL → OC では IL のオブジェクトを削除する処理と OC のオブジェクトを追加する処理の組合せのため，更新時間はそれらの更新時間を合わせたものとほぼ等しい．図 16 と図 17 では，どの更新パターンにおいても提案手法の更新時間が最も短いことが確認できる．図 18 では，状態変化を調べる対象となるオブジェクト数が少ないため，OL → OC の更新を行った場合を除いて提案手法とグリッド索引のみ用いる手法での更新時間の差は小さい．P-Space を再計算する手法では全オブジェクト間のペアで距離計算を行う

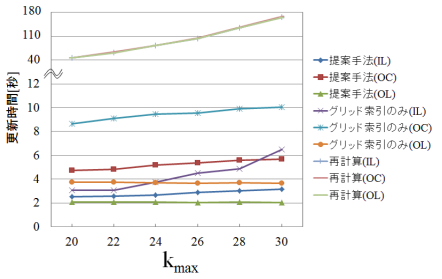

図 14　オブジェクトの追加

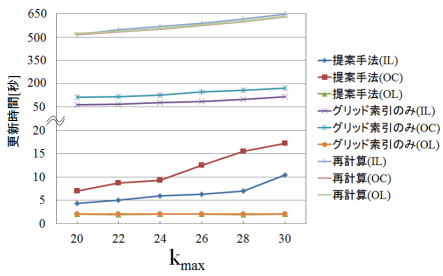

図 15　オブジェクトの削除

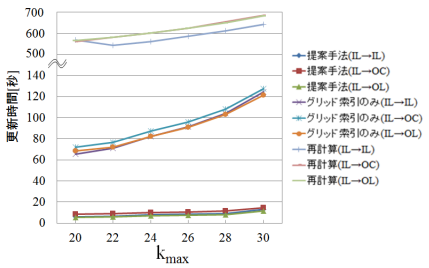

図 16　IL に分類されている
オブジェクトの更新

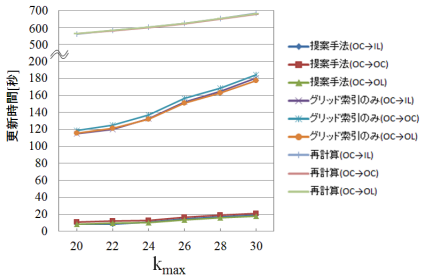

図 17　OC に分類されている
オブジェクトの更新

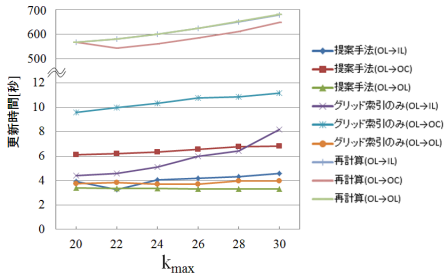

図 18　OL に分類されている
オブジェクトの更新

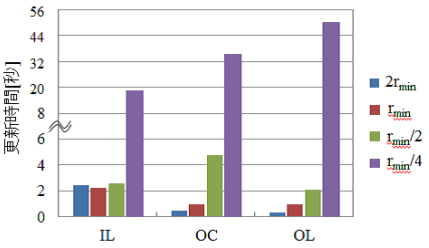

図 19　オブジェクトの追加

ため，各更新で更新時間は同程度である．k_{max} の値が大きくなるほど OC のオブジェクト数と P-Space のテーブル数が増加するため，k_{max} の増加とともに更新時間は増加する．

なお，k_{min} を変化させ，オブジェクトの追加，削除，更新を行った実験においても同じように提案手法の更新時間が短い傾向が見られた．k_{min} の値が大きくなるほど OC のオブジェクト数と P-Space のテーブル数が減少するため，P-Space を再計算する手法では k_{min} の増加とともに更新時間は減少する．

7.4.2　実　験　2

実験 2 の提案手法に対する結果を図 19 ～ 21 に示す．各パラメタは k_{min}=15，k_{max}=20, r_{min}=50，r_{max}=150 とし，グリッドの対角線の長さは $2r_{min}$，r_{min}，$r_{min}/2$，$r_{min}/4$ で比較する．図 19 は IL，OC，OL に分類されるオブジェクトを追加した場合，図 20 は IL，OC，OL に分類されているオブジェクトを削除した場合，図 21 は各更新をした場合を示す．グリッドの対角線の長さを $r_{min}/4$ と設定し，グリッドサイズを小さくしすぎるとオブジェクトの状態変化を確認するために検索しなければならないグリッド数が増えるため，更新に時間がかかる．一方で，グリッドの対角線の長さを $2r_{min}$ と設定し，グリッドサイズを大きくしすぎるとオブジェクト間の距離計算をしなければならない回数が増加するため，更新に時間がかかる．用いるデータ集合に応じて適切なグリッドサイズにすることが重要と言える．

7.4.3　実　験　3

実験 3 の提案手法に対する結果を図 22 と図 23 に示す．各パラメタは k_{min}=15，k_{max}=20, r_{min}=50，r_{max}=150 とし，グリッドの対角線の長さは $k_{min}/2$ とする．追加または削除するオブジェクト数は 2 つ，3 つ，4 つ，5 つのそれぞれ行う．オブジェクト数が 2 つのときは 1 つのグリッドで 2 つのオブジェクト，オブジェクト数が 3 つのときは 2 つのグリッドでそれぞれオブジェクトを 2 つと 1 つ，オブジェクト数が 4 つのときは 2 つのグリッドでそれぞれオブジェクトを 2 つずつ，オブジェクト数が 5 つのときは 3 つのグリッドでそれぞれオブジェクトを 2 つ，2 つ，1 つ追加または削除する．図 22 はオブジェクトを追加した場合，図 23 はオブジェクトを削除した場合を示す．バッチ処理により，7.1 節で述べたようにカウンタの更新と P-Space 更新の効率化が可能になるため，バッチ処理を行わない場合よりも追加と削除ともに更新時間が短くなった．

7.4.4　実　験　4

実験 4 の結果を図 24 と図 25 に示す．各パラメタは k_{min}=15，r_{min}=0.2，r_{max}=0.6 とし，k_{max} を 20 から 28 まで 4 ずつ増やす．グリッドの対角線の長さは $r_{min}/2$ とする．データ集合を更新する前の各状態のオブジェクト数は k_{max}=20 で (IL, OC, OL)=(428483, 6107, 284)，k_{max}=28 で (IL, OC, OL)=(424535, 10055, 284) となっている．

図 24 は IL，OC，OL に分類される各オブジェクトをそれぞれの手法で追加した場合，図 25 は IL，OC，OL に分類されている各オブジェクトをそれぞれの手法で削除した場合を示す．オブジェクトを追加する場合は OL と OC のオブジェクトが状態変化するかを確認する必要があるが，それらのオブジェクト数が少ないため，IL と OC のオブジェクトを追加した場合は提案手法とグリッド索引のみ用いる手法の差は小さい．k_{max}=20 では P-Space を再計算する手法でも更新にあまり大きな時間がかからない．しかし，k_{max} の値が大きくなるほど OC のオブジェクト数と P-Space のテーブル数が増加するため，再計算する手法は時間がかかるようになる．一方で，オブジェクトを削除

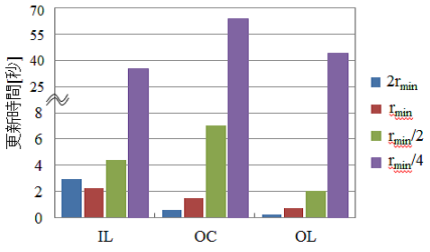

図 20　オブジェクトの削除

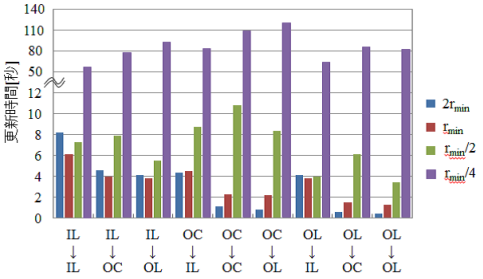

図 21　オブジェクトの更新

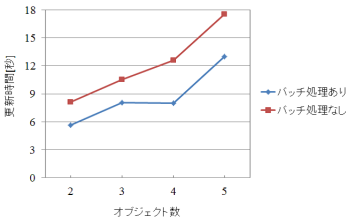

図 22　オブジェクトの追加

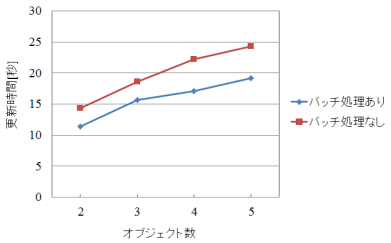

図 23　オブジェクトの削除

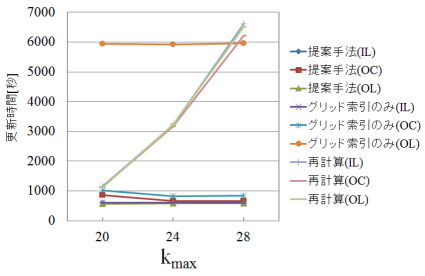

図 24　オブジェクトの追加

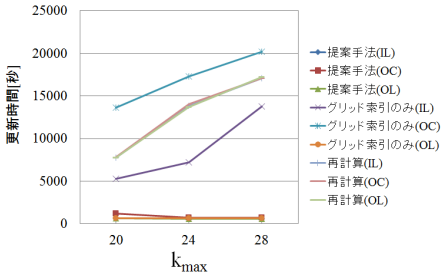

図 25　オブジェクトの削除

する場合は IL と OC のオブジェクトが状態変化するかを確認する必要がある．IL のオブジェクト数が多いため，グリッド索引のみ用いる手法と再計算する手法では多くの時間がかかる．

なお，オブジェクトの更新はオブジェクトの追加と削除の組合せのため，追加や削除と同じように提案手法の更新時間が短い傾向が見られた．

8. ま と め

本研究では，動的なデータ集合での対話的外れ値分析を可能にする手法を提案した．提案手法ではグリッド索引とカウンタを用いて ONION の索引構造を更新し，オブジェクトが追加，削除，更新される環境での適用を可能にした．ONION の索引構造を更新するためには，追加や更新されたオブジェクトの状態判別に加えて，既存オブジェクトの状態変化を調べる必要がある．グリッド索引を用いることで，状態が変化するオブジェクトを効率的に見つけることを可能にした．また，既存オブジェクトの状態変化を調べるためにカウンタを用いることで，距離計算を減らすことを可能にした．実験では，各環境でパラメタの値を変えて性能評価を行い，提案手法の有効性を確認した．性能比較のため，グリッド索引のみを用いる手法と，グリッド索引とカウンタをともにもたない手法を用いた．また，バッチ処理によるカウンタ更新と P-Space 更新の効率化に関して性能比較を行った．

今後の課題としては，より多様な実データを用いた評価や，P-Space で保持するテーブル数を k_{max}-k_{min}+1 個ではなく，多様に変化させた場合の評価等が挙げられる．

謝辞　本研究成果は，平成 28 年度共同研究 (SKY 株式会社)(CPE27116K)「機械学習の適用による SKYSEA Client View のログ及び資産情報からの例外的状況の自動検出」により得られたものです．

文　献

[1] D. Hawkins. Identification of Outliers. Chapman and Hall, London, 1980.
[2] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In Proc. VLDB, 1998.
[3] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. VLDB, 1999.
[4] E. M. Knorr, R. T. Ng and V. Tucakov. Distance-based outliers: algorithms and applications. VLDB, 2000.
[5] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In Proc. SIGMOD, 2000.
[6] M. M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In Proc. SIGMOD, 2000.
[7] R. T. Ng and J.Han. Efficient and effective clustering methods for spatial data mining. VLDB, 1994.
[8] M. Ester, H. P. Kriegel and X. Xu. A database interface for clustering in large spatial databases. KDD, 1995.
[9] T. Zhang, R. Ramakrishnan and M. Livny. Birch: an efficient data clustering method for very large databases. SIGMOD, 1996.
[10] E. Eskin. Anomaly detection over noisy data using learned probability distributions. ICML, 1994.
[11] V. Barret and T. Lewis. Outliers in statistical data, Wiley, Chichester, 2001.
[12] H.-P. Kriegel, M. Schubert, and A. Zimek. Angle-based Outlier Detection in High-dimensional Data. In Proceedings KDD'08, pages 444-452, 2008.
[13] R. P. N. PhamA. A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data. Proc. ACM SIGKDD, pages 877-885, 2012.
[14] Lei Cao, MingruiWei, Di Yang, and Elke A. Rundensteiner. Online Outlier Exploration Over Large Datasets. KDD'15, 2015.
[15] F.Angiulli and F.Fassetti. Detecting distance-based outliers in streams of data. CIKM 2007.
[16] K. Ishida and H. Kitagawa. Detecting current outliers: Continuous outlier detection over time-series data streams. Proc. DEXA, LNCS, 5181: 255-268, 2008.