3F1-03

ウェブアクセスログのためのデータ圧縮に基づく逐次型部分空間クラスタリング

Detecting Hostile Accesses through Incremental Subspace Clustering for Web Access Log

奈良橋 正樹[*1] 鈴木 英之進[*2]
Masaki NARAHASHI Einoshin SUZUKI

[*1]横浜国立大学大学院工学府
Faculty of Engineering, Graduate School, Yokohama National University

[*2]横浜国立大学大学院工学研究院
Faculty of Engineering, Graduate School, Yokohama National University

In this paper, we propose an incremental subspace clustering method for flexibly detecting hostile accesses to a Web site. Typical log data for Web accesses are huge, contain irrelevant information, and exhibit dynamic characteristics. We overcome these difficulties through data squashing, subspace clustering, and an incremental algorithm. We have improved, by modifying its data squashing functionality, our subspace clustering method SUBCCOM so that it can exploit previous results. Experimental evaluation confirms superiority of our I-SUBCCOM in terms of precision, recall, and computation time.

1. 序論

クラスタリングは，与えられた例集合を類似グループに分割する学習問題である [3]．われわれは，距離に基づく圧縮可能性でクラスタを定義し，関連する属性を選択でき，データを圧縮して処理するクラスタリング手法 SUBCCOM を提案した [4]．SUBCCOM は，不要な属性が含まれていても，大量のデータを高速にクラスタリングできるという特長がある．

近年，われわれのインターネットへの依存度は高まる一方であり，ウェブサイトへの敵対的アクセスは深刻な問題を引き起こす場合がある．ウェブサイトへの敵対的アクセスをクラスタリングを用いて特定するためには 3 つの条件，大量データの高速処理 (ウェブログは大量)，関連属性の特定 (無関係な属性あり)，インクリメンタル処理 (ウェブログは逐次更新) が必要と考えられる．SUBCCOM は最後の点に対処できないので，本論文でインクリメンタル版の I-SUBCCOM を提案する．

2. SUBCCOM

2.1 部分空間クラスタリング問題

部分空間クラスタリングへの入力は，$\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$ の m 個の例であり，それぞれ n 個の数値属性から構成される．これらの n 個の属性から構成される集合を V とする．ここで $\mathbf{x}_i$ は，n 次元空間上の点を表す．出力は c 個のクラスタ $\gamma_1, \gamma_2, \cdots, \gamma_c$ と，関連する属性集合 v ($\subseteq V$) である．v に含まれる属性はクラスタが存在する部分空間を表し，$\gamma_1, \gamma_2, \cdots, \gamma_c$ は $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$ の互いに排反な分割である．ユーザは v に含まれる属性数 l を指定する．

2.2 データ圧縮のための CF 木

CF(Clustering Feature) 木は，高速なクラスタリングアルゴリズム BIRCH[5] で提案された B+木 [2] に類似する高さ平衡木であり，データ圧縮のためのデータ構造である．CF 木のノードは，子ノードへのエントリーとして CF ベクトルの集合を持ち，各要素は例集合の抽象表現に相当する．例集合 $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N$ を圧縮すると，CF ベクトル $\mathbf{CF}$ は例数 N，例の線形和 $\sum_{i=1}^{N} \mathbf{x}_i$，および例の属性値の二乗和 $\sum_{i=1}^{N} \|\mathbf{x}_i\|^2$ を要素として持つ．CF ベクトルは加法性を満たすので，動的に更新可能であり，よって BIRCH では訓練例を一回スキャンするだけで済む．2 個の CF ベクトルから様々なクラスタ間距離を計算できるため，元のデータ集合を保存する必要がない．

CF 木は B+木に類似する手順で構築される．新しい例が入力されると，根から葉へ降り，通過したノードは更新される．適切なノードの選択はユーザが指定する距離基準に基づく．葉では，エントリー内の例との距離が与えられた閾値 L 以内であれば，例は最も近いエントリーに割り当てられる．葉のエントリー，すなわち CF ベクトルは CF 木の最小単位であり，圧縮された例の集合を表す．

2.3 評価基準としての圧縮可能性

CF 木は用いられる属性集合 u に大きく依存する．前節で述べたように，CF 木の葉のエントリーは互いに近い例の集合から構成される．したがって，葉のエントリーの数 $\mu(u)$ は例空間における密な領域の数に相当し，圧縮可能性の指標を定義するのに用いることができる．われわれは部分空間の評価基準として $-\mu(u)$ を提案した [4].

2.4 関連する属性集合のヒューリスティック探索

属性の部分集合は束構造を構成し，その上界は V で下界は空集合である．前節で述べた評価基準に基づいて，V の適切な部分集合を束構造において探索することで，関連する属性の集合 v を求める．前方探索と後方探索はそれぞれ下界と上界からの幅優先探索を意味する．

われわれが [4] で提案したジャンプ探索は，正確さを損なわずに，時間的効率がよいことが期待されている．この手法は最初に各属性 a_i について閾値 $L = T_1$ で CF 木を構築し，評価基準値 $-\mu(a_i)$ がその平均値以上の属性集合 u を得る．ただし，$\mu(a_i) = 1$ となる属性は，属性値がほぼ同じ値であり，クラスタリングに貢献しないと考えられるので除外する．$|u| < l$ となる場合，$|u|$ が l となるように残りの属性から評価基準値の大きい順に，さらに除外した属性を u に追加する．最後に，u から $L = T_2$ で後方探索を実行する．単一の属性は高い圧縮可

連絡先: 奈良橋正樹，横浜国立大学大学院工学府物理情報工学専攻，〒 240-8501 横浜市保土ヶ谷区常盤台 79-5, Tel: 045-339-4135, Fax: 045-339-4148, E-mail: narahashi@slab.dnj.ynu.ac.jp

能性を示すため，T_1 は概して T_2 より小さな値に設定される．

SUBCCOM は部分空間クラスタリングシステムであり，最初に関連する可能性のある属性の集合 u と相当する CF 木の葉のエントリー $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_p$ をジャンプ探索で得てから，$\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_p$ に対して k-means 法 [3] を適用する．SUBCCOM では，手続き *k-means* は圧縮された例 $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_p$ をクラスタリングするように修正されている．

Algorithm: *SUBCCOM*($\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$, l, v, $\gamma_1, \gamma_2, \cdots, \gamma_c$)
Input: 例集合 $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_m$, 属性数 l
Output: 部分空間 v, クラスタの集合 $\gamma_1, \gamma_2, \cdots, \gamma_c$
Parameter: 最終的な CF 木 τ の葉のエントリー $\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_p$

For($i = 1; i \leq n; i$++)
  a_i だけを用いて閾値 $L = T_1$ で CF 木を構築
$u = \phi$
Foreach($-\mu(a_i) \geq \sum_{j=1}^{g} -\mu(a_j)/g$ となる a_i ただし $\mu(a_i) > 1$ で g はその個数)
  $u = u \cup \{a_i\}$
While($|u| < l$)
  $u = u + \{a^*\}$ ただし $a^* = \mathrm{argmax}_{a \notin u} -\mu(\{a\}), \mu(a) > 1$
While($|u| < l$)
  $u = u + \{a^*\}$ ただし $a^* \notin u$
While($|u| > l$)
  Foreach($a_i \in u$)
    $u - \{a_i\}$ だけを用いて閾値 $L = T_2$ で CF 木を構築
  $u = u - \{a^*\}$ ただし $a^* = \mathrm{argmax}_a -\mu(u - \{a\})$
u だけを用いて閾値 $L = T_2$ で最終的な CF 木 τ を構築
$v = u$
$(\gamma_1, \gamma_2, \cdots, \gamma_c) = k\text{-means}(\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_p)$

3. 提案手法

3.1 ウェブアクセスログからの不正アクセスの特定

ユーザがウェブサイトにアクセスするとアクセスログが生成される．次にアクセスログの具体例を示す．

kali.slab.dnj.ynu.ac.jp - ningen [04/Apr/2001:16:48:57 +0900] "GET /algodata HTTP/1.0" 401 476

これは左からリモートホスト名[*1]，リモートホストで ident サービス[*2]を提供しているときのユーザ名，認証を行った際のユーザ名，アクセス日時，リクエストの内容，リクエストに対応するステータスコード，送出した情報量 (バイト) である．リクエストの内容は，HTTP の命令，要求されたファイル名，HTTP のバージョンから構成される．この例は，kali.slab.dnj.ynu.ac.jp からアクセスしてきたユーザが，認証ユーザ名 ningen で/algodata 以下のファイルを取得しようとしたものの，おそらくパスワードを間違ってしまったためにステータスコードが 401，つまり認証拒否となったものである．

ウェブサイトの管理者はウェブアクセスログから，不正なアクセスをなるべく早く検知し，対策を取る必要がある．しかし，ウェブアクセスログは膨大で，逐次更新され，傾向が変化するため，高速な検知は困難である．そこで，既存の結果を再利用する逐次型部分空間クラスタリング手法を前処理として用い，このタスクを容易にする．具体的には，例を互いに類似するグループに分割することで，特異な例を少数の例から構成されるクラスタに分離できると期待できる．その場合，各クラスタから同数の例を取り出して調べたとき，少数の例から構成されるクラスタから取り出した例集合には高い割合で不正なアクセスが含まれる．

以下，不正アクセス特定のためのウェブアクセスログクラスタリングを定義する．問題への入力は s 個の例集合 $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w_s}$ であり，$\mathbf{w}_i$ は複数個の例から構成され，これを処理単位と呼ぶ．同時にクラスタリングする処理単位を処理対象と呼び，処理対象に含まれる処理単位の個数は h 以下とする．処理対象が h 個の処理単位 $\mathbf{w}_i, \mathbf{w}_{i+1}, \cdots, \mathbf{w}_{i-1+h}$ の場合に，新しい処理単位 $\mathbf{w}_{i+h}$ を追加するときは，最も古い処理単位 $\mathbf{w}_i$ を削除する．処理対象を逐次更新しつつ，更新のたびに部分空間クラスタリングを行い結果を出力していく．

アルゴリズムの性能は，不正アクセスログに関する適合率と再現率で評価する．もっとも，全てのアクセスログを見ることは人間にとって困難であるため，各クラスタから同じ数だけ例を取り出し，取り出した例に含まれる不正ログに関して評価する．適合率 P と再現率 R は，次式で与えられる．

$$P = \frac{\text{(取り出した例に含まれる不正ログの数)}}{\text{(取り出した例数)}}$$

$$R = \frac{\text{(取り出した例に含まれる不正ログの数)}}{\text{(不正ログの数)}}$$

3.2 I-SUBCCOM

前節の問題を 2.4 節の SUBCCOM で解決すると，処理対象が更新される度に計算を一からやり直す必要があるため，非効率的である．この問題に対処するため，SUBCCOM を逐次処理型に改良した I-SUBCCOM を提案する．

I-SUBCCOM は，CF 木の葉に，処理単位ごとの例を別々の CF ベクトルに保存しておくことで，既存の結果を再利用する．CF 木の葉 ρ_i は，処理単位ごとの CF ベクトル $\mathbf{y}_{i1}, \mathbf{y}_{i2}, \cdots, \mathbf{y}_{ij}$ $(1 \leq j \leq h)$ を持ち[*3]，同じ処理単位の例は同一の CF ベクトルに格納される．葉は，各 $\mathbf{y}_{ij}$ の j に関する和である $\mathbf{y}_i$ も持ち，例が葉に属するかを判定するための距離計算には $\mathbf{y}_i$ を用いる．処理単位 $\mathbf{w}_k$ を削除するときは，$\mathbf{y}_{ik}$ を削除してから，$\mathbf{y}_i$ を再計算する．

新しい処理単位を処理対象に加えるとき，直前に構築した CF 木における葉のエントリーと追加した処理単位を新たに処理対象とする．処理単位が h 個の処理対象となっている状態で，新しい処理単位を追加するときは，葉のエントリーから一番古い処理単位で構成される CF ベクトルをあらかじめ削除しておく．葉のエントリーと圧縮されていない例を併せて CF 木を構築する場合は，最初に葉のエントリーを元に CF 木をつくり，次にこの木に圧縮されていない例を流す．

I-SUBCCOM のアルゴリズムは次の通りである．ただし，手続き *build-itree* は集合 Ω 中の 葉のエントリー $\mathbf{y}_i$ から本節で定義した CF 木 τ を構築し，$\Omega = \phi$ なら空木を返す．手続き *SUBCCOM2* は τ を初期 CF 木とし，k-means 法の代わりに k-medoids 法を用いる[*4]．

---

*1 IP アドレスが記録されていることもある
*2 認証のためにユーザ情報を提供するデーモン
*3 全ての処理単位に属する例が葉に割り当てられるとは限らないため，CF ベクトルの個数は h 個より少ない場合がある
*4 SUBCCOM では例はユークリッド空間上の点を表すと仮定するが，I-SUBCCOM ではウェブアクセスログを表すと仮定する．後者では，クラスタの平均は存在しないウェブアクセスログを表す場合がある

Algorithm: *I-SUBCCOM*($\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_s$, l)
Input: 処理単位 $\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_s$, 属性数 l
Output: クラスタの集合 $\Gamma_1, \Gamma_2, \cdots, \Gamma_s$

```
1   Ω = φ
2   For(i = 1; i ≤ s; i++)
3     τ = build-itree(Ω)
4     w_i に含まれる例を x_1, x_2, ···, x_m とする
5     SUBCCOM2(τ, x_1, x_2, ···, x_m, l, u, γ_i1, γ_i2, ···,
      γ_ic)
6     Γ_i = {γ_i1, γ_i2, ···, γ_ic}
7     Γ_i を出力
8     If(i > h)
9       Foreach(j)
10        τ の葉から y_{j i−h+1} を削除し，y_i を再計算
11    Foreach(j)
12      Ω = Ω + {y_j}
```

4. 実験評価

4.1 ウェブアクセスログの前処理

実験には例数 205,590 であり，99 週間にわたる www.slab.dnj.ynu.ac.jp のウェブアクセスログを用いる．I-SUBCCOM をアクセスログに適用する際に前処理として，距離計算のために属性値はすべて数値に変換する必要がある．ウェブアクセスログを以下の手順で変換し，属性数は合計 6 個とした．

- リモートホスト：リモートホストは研究室内，学科内，学内，国内の他大学，国内その他，それ以外の 6 種に分け，それぞれ 0 から 10 刻みで 50 まで割り当てる．大半を占める国内からのアクセスを細かく区分した．
- ユーザ名：ユーザ名なし，YNU，dnj，member，ningen，student，その他の 7 種に分け，それぞれ 0 から 10 刻みで 60 まで割り当てる[*5]．ユーザ名によってアクセスするディレクトリが異なるため，異なる数値を割り当てた．
- HTTP の命令:GET，HEAD，POST，PUT，DELETE，TRACE，OPTIONS，CONNECT，その他にそれぞれ 0 から 10 刻みで 80 まで割り当てる．
- 要求されたディレクトリ：1000 回以上アクセスのあったディレクトリ，/，/jkiso，/challenge2000，/algodata，/testhomepage，/ismis2002 と，それ以外に分け，それぞれ 0 から 10 刻みで 60 まで割り当てる．要求されたディレクトリのファイル名まで含めた文字数が 50 以上のものは 100 を割り当てる．ワームによるアタックや存在しないファイルを要求している場合など，悪意のあるものは文字数が多くなる傾向があるためである．
- ステータスコード：百の位の 10 倍に置換する．値は 20, 30, 40, 50 の 4 種類となった．
- 情報量：0 バイトを 0，1 から 500 バイトまでを 10，501 から 1000 バイトまでを 20，1001 から 1500 バイトまでを 30，1501 から 10000 バイトまでを 40，それ以上を 50 とする．各区間に数万例ずつ含まれるように区切った．

*5 具体名は，認証ユーザ名である．その他は主に打ち間違いである

4.2 実験

実験ではログの処理単位を 1 週間，処理対象に含まれる処理単位の最大値を $h = 4$，生成するクラスタの個数は $c = 5$ とした．I-SUBCCOM の閾値は，$T_1 = 0.01$，$T_2 = 2.0$ とし，また BIRCH の閾値は $L = 2.0$ とした．

人間にとっては，全てのアクセスログを読むことも，全ての不正アクセスログを調べることも，困難である．5 個のクラスタから無作為に 20 個ずつ例を取り出し，その内容を評価した．ただしクラスタに含まれる例数が 20 に満たない場合はすべて取り出した．Nimda と Code Red は例数が非常に多く，また要求するファイル名が特徴的であるため他のアクセスとの区別が容易である[*6]．実験ではこれらを不正ログとして適合率と再現率を求めた．例を無作為に抽出するため，100 回の平均値を評価の指標とした．

I-SUBCCOM について，クラスタの属性数が $l = 5, 4$ の場合について実験を行った．比較として，PROCLUS [1] ($l = 5, 4$)，部分空間クラスタリングを行わない BIRCH と k-medoids 法についても同様の実験を行った．公平な比較のため，BIRCH は CF 木の表現と構築方法を I-SUBCCOM と同じとし，逐次型クラスタリングアルゴリズムに変更した．また，クラスタリングをしないで，アクセスログから無作為に 100 例抽出した場合についても適合率と再現率を求めた．

結果を元に適合率，再現率と実行時間のグラフを作成した．ただし適合率と再現率は，Code Red がアクセスログに登場する 29 週目から求めた[*7]．適合率を図 1 に，再現率を図 2 に，またすべての週を処理するのに要した時間を図 3 に示す．図 1 と図 2 では図を見やすくするため，I-SUBCCOM と PROCLUS は結果の良い方だけを示した．

4.3 実験結果の分析

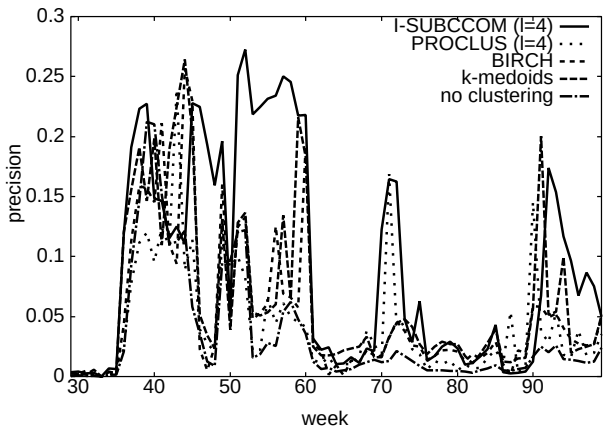

図 1: 適合率に関する結果

図 1，2 より，読者は再現率と適合率が意外に低いと思われるかも知れない．しかし，3.1 節で述べたように，ウェブアクセスログクラスタリングは不正アクセスログを特定するための前処理として使用されるため，この程度の数値でも監視者に警告を発する機能は果せると考えられる．なお，クラスタリング無しはランダムサンプリングに相当するため，図において不正アクセスログの統計値を調べる際に利用できる．

*6 アクセスログに"root.exe"，"cmd.exe"，"default.ida" の文字列が含まれていれば，Nimda,Code Red と判定した
*7 28 週目まで Code Red も Nimda も存在しない

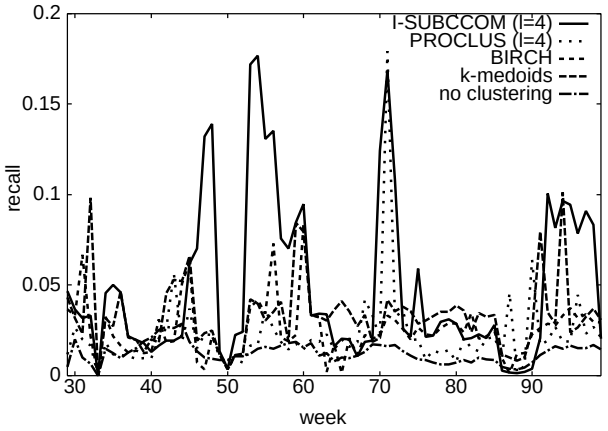

図 2: 再現率に関する結果

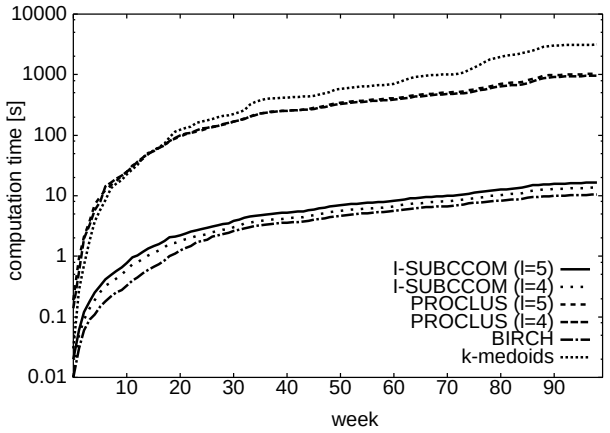

図 3: 実行時間に関する結果

36 週目から 60 週目までは，いずれのクラスタリング手法も適合率が比較的大きいが，このときは不正アクセスログの割合が大きいため，クラスタリングしなくても不正アクセスログを発見することは十分可能である．したがって，54 週から 56 週までは I-SUBCCOM(l=4) の適合率が，クラスタリングしない場合を 10 倍程度上回るものの，この区間では各クラスタリング手法が効果的であるとはいえない．

しかし，61 週から 99 週までの，不正アクセスログの割合が低い区間に目を向けると，クラスタリングする場合の適合率が，しない場合の適合率を大きく上回る．特に，71 週目の I-SUBCCOM(l=4)，PROCLUS (l=4)，72 週目の I-SUBCCOM(l=4)，90 週目の PROCLUS (l=4)，91 週目の k-medoids 法，92 週目の I-SUBCCOM(l=4)，93 週目の I-SUBCCOM (l=4) では，前者が後者を適合率に関して 10 倍程度上回っている．処理対象に含まれる不正アクセスログが少なくても，その内の多くを特定することに成功しており，クラスタリングする方が有用であることがわかる．これは，不正アクセスログを少数の例からなるクラスタに分離することに成功していることを表している．

再現率も適合率と同様に，クラスタリングする場合の方がしない場合よりもほぼすべての場合で高くなっている．61 週から 99 週において適合率が特に高いところは再現率も同様に高い．なお，30 週目と 40 週目の間に再現率が 0 となる期間，処理対象に不正アクセスログは含まれていない．

適合率と再現率に関しては，各クラスタリング手法の間に顕著な違いは見られないものの，実行時間には大きな差がある．すべての処理対象に要した時間は，I-SUBCCOM が k-medoids 法の約 186 倍，PROCLUS の約 61 倍高速だった．I-SUBCCOM は既存の結果を再利用することでこの高速性を実現している．BIRCH は，クラスタリングに適した部分空間を選択しない通常のクラスタリング手法であるため，I-SUBCCOM より約 1.3 から 1.6 倍高速だった．しかしその代償として，BIRCH は適合率，再現率において，他の手法を上回る部分が少ない．よって今回の実験で最も有用な手法は I-SUBCCOM と考えられる．

クラスタリングの副次効果として，珍しいアクセスの発見が容易になったことがあげられる．例えば HTTP の命令はほぼすべて GET であり，それ以外の命令のログは極めて少なく，クラスタリングしない場合は特定できる可能性が低い．しかし，これらのログが少数の例からなるクラスタに分離されたことで，ほぼ確実に特定できるようになった．

5. 結論

ウェブアクセスログは大量で無関係な属性を含み，逐次更新される．本論文は，ウェブサイトへの不正アクセスを特定する有望な前処理として，逐次型部分空間クラスタリングを提案した．われわれは以前提案した SUBCCOM [4] を，データ圧縮用のデータ構造を改良することで逐次型に改良し，上記の必要条件を全て満たした．

実験の結果，本論文で提案する I-SUBCCOM は再現率，適合率，および計算時間に関して有望な結果を示し，比較対象の手法よりも優れていることが示された．今後は，名目属性を数値属性に自動的に変換する手法や，他の種類の不正検知への適用に取り組んで行きたい．

参考文献

[1] C. C. Aggarwal et al. Fast Algorithms for Projected Clustering, *Proc. 1999 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD)*, pp. 61-72, 1999.

[2] D. Comer, The Ubiquitous B-Tree, *ACM Computing Surveys*, 11(2):121-137, June 1979.

[3] L. Kaufman and P. J. Rousseeuw, *Finding Groups in Data*, Wiley, New York, 1990.

[4] M. Narahashi and E. Suzuki, Subspace Clustering Based on Compressibility, *Discovery Science, Lecture Notes in Computer Science 2534 (DS)*, pp. 435-440, Springer-Verlag, 2002.

[5] T. Zhang and R. Ramakrishnan and M. Livny, BIRCH: An Efficient Data Clustering Method for Very Large Databases, *Proc. 1996 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD)*, pp. 103-114, 1996.