能動学習に基づいたマルウェア階層的クラスタリング

何 天祥[2]　韓 燦洙[1]　高橋 健志[1]　来嶋 秀治[2]　竹内 純一[2]

概要：IoT マルウェア検体の数は，近年急速に増加し，多様化している．大量なマルウェア検体を効率的に分析するために，我々はスケーラブルなクラスタリング手法を研究している．本研究では平均場アニーリング (MFA) を用いてクラスタリングを行い，能動学習によって距離行列の極一部のみを観察する手法を実験で 3,008 検体の IoT マルウェアを用いてアルゴリズムの評価を行った．また，オンライン処理する手法を提案し，評価した．能動学習クラスタリング (Active Clustering, AC) 手法を適用することにより，距離行列全体の 2.6%のみを計算するだけでクラスタリングを行った．その結果，86.9%のファミリ名正解率と 96.5%アーキテクチャ名正解率を達成した．また，我々の先行研究の手法では距離行列の 7.2%を観測する必要があったが，AC では同程度の精度を保ちながら，観測量を 64%削減した．

キーワード：能動学習，クラスタリング，平均場アニーリング，IoT マルウェア

Malware Hierarchical Clustering Applying Active Learning

Tianxiang He[2]　Chansu Han[1]　Takeshi Takahashi[1]　Shuji Kijima[2]　Jun'ichi Takeuchi[2]

Abstract: In recent years, the number of IoT malware specimens has rapidly increased and diversified. In order to effectively analyze a large number of malware specimens, our goal is to cluster from the incomplete distance matrix of the specimens. To this end, we use Mean Field Annealing (MFA) for clustering and use active data selection to determine which distance to observe actively. We also proposed an online processing method for additional collected malware. We used 3,008 IoT malware specimens for experimental evaluation. By applying the active clustering algorithm, clustering is performed by only calculating 2.6% of the entire distance matrix. The family name accuracy was 86.9%, and the architecture name accuracy was 96.5%. In addition, in our previous research, 7.2% of observations of the distance matrix were required, but the active clustering algorithm reached the same level of accuracy, with much lesser observations. The observation reduction rate was 64%.

Keywords: active learning, clustering, mean field annealing, IoT malware

1. はじめに

IoT(Internet of Things) テクノロジーの発展と，IoT デバイスの数の多様化に伴い，IoT マルウェアの数も急速に増加している．これは，*Bashlite* や *Mirai* などの主要な IoT マルウェアのソースコードがインターネット上で漏洩し [1], [4]，そして多くの IoT デバイスはセキュリティ設計が乏しいことが主な原因である [12]．マルウェア解析者は，ハニーポットやマルウェア対策プラットフォーム (VirusTotal [*1]など) などのさまざまなソースから取得した数千または数万の IoT マルウェア検体セットを入手し解析を行う．マルウェアの傾向を追跡するために，これらの大量なマルウェアを効率的に分析する方法が必要である．

本論文では，静的分析とクラスタリングに基づいて IoT マルウェア検体を分析する．大規模なマルウェア検体セットを分析する場合，すべての検体ペア間の類似性を計算する必要があり，そのときの処理時間が問題になり兼ねない．

[1] 国立研究開発法人情報通信研究機構
National Institute of Information and Communications Technology
[2] 九州大学
Kyushu University

[*1] https://www.virustotal.com

我々の先行研究では，マルウェアの系統樹を構築することにより，マルウェアを高速にクラスタリングするスケーラブルなアルゴリズム「FACCP」を提案した [7]．FACCP は静的分析に基づいており，検体間の距離行列のごく一部だけを計算し，クラスタリングを行う．類似性の計算コストを $O(N^2)$ から $O(N \log N)$ まで削減できた．ここで，N は検体セットのサイズである．

我々は正規化圧縮距離 (Normalized Compression Distance, NCD) [10] を用いて，マルウェア検体のバイナリ間の類似性を測る．マルウェアの亜種は，元となるソースコードの一部を改変して作成されることがよくある [1]．したがって，同じコンパイル環境において，これらの検体は高いバイナリ類似性を持つことが期待できる．2 つの文字列 (バイナリ) 間の類似性が高いほど，NCD は小さくなる [5], [6]．したがって，バイナリが類似する検体は同じまたは近いクラスターに分割されることが期待できる．FACCP では，NCD の圧縮時間が実行時間の大部分を占めていた．

FACCP は，距離行列のごく一部のみをランダムに計算することにより，大規模なデータに対応している．この枠組みによる系統樹の作成は回帰モデルの学習すなわち，教師あり学習として捉えられ，距離データはその属性 (共変量) とみなせる．これは能動学習に類似しているため，能動学習を採用し更なる性能を達成したい．実際，能動学習によるクラスタリング (Active clustering, AC) 手法がいくつか提案されている [8], [18]．中間段階である系統樹の作成を飛ばしてクラスタリングのみを目的とするなら，その手法を適用することが可能である．そこで，AC の既存手法 [8], [18] と FACCP をマルウェアクラスタリングにおいて評価した．

本稿では，能動学習を適用して，検体間類似度の計算コスト (NCD 計算回数) を削減した．能動学習は，距離行列のどの部分を計算するかを能動的に賢く選択することにより，FACCP よりもさらに優れた計算量削減を達成できた．能動学習の重要なアイデアは，学習アルゴリズムによりスマートに観察するデータを選択することである．能動学習は，通常の機械学習アルゴリズムよりもはるかに少ないラベル付きデータを必要とするため，ラベル付きデータの取得が時間・費用のコスト面で困難な場合に適している [16]．

また，我々は新しい検体を逐次的にクラスタリングに加えるためのオンライン処理手法を提案した．3,008 検体の IoT マルウェアを用いて実験を行い，AC 及びオンライン処理手法を評価した．

本論文の貢献は以下になる：

- 実 IoT マルウェア検体 (3,008 検体) を用いて AC の評価を行った．その結果 86.9%のファミリー名正解率と 96.5%のアーキテクチャ名正解率を達成した．
- AC のスケーラビリティを評価した．AC は 97.4%の計算量削減を達成し，先行研究 FACCP の 92.8%より大きく改良できた．結果として，AC では同程度の精度を保ちながら，観測量を 64%削減できた．
- オンライン処理のアルゴリズムを提案し，実験でその有効性を示した．

2. 事前知識

このセクションでは本論文の背景知識を紹介する．

2.1 正規化圧縮距離

正規化圧縮距離 (略称 NCD) は，情報理論に基づき 2 つのオブジェクト間の類似性を測る手法である [10]．NCD は，オブジェクトの圧縮率に基づいて計算される．本論文では NCD を用いて IoT マルウェアバイナリ間の類似性 (距離) を計算した．2 つの文字列 x と y の間の NCD は次のように定義される：

$$\mathrm{NCD}(x,y) = \frac{C(xy) - \min\{C(x), C(y)\}}{\max\{C(x), C(y)\}}.$$

ここで，$C(x)$ は，圧縮アルゴリズム C によって圧縮された x の文字列長である．xy は，文字列 x と y を連結したものである．xy を圧縮するとき，圧縮プログラムは最初に x を圧縮し，次に x の情報を用いて y を圧縮する．したがって，x と y の同じパータンが多いほど，xy の圧縮率は高くなる．

2.2 平均場アニーリング

MFA は，平均場理論とシ焼きなまし法 (Simulated Annealing, SA) を組み合わせたものである．平均場理論は，高次元の相互作用する多くの分子が含まれている複雑なモデルをより単純化したモデルにする．数多くの相互作用を計算する代わりに，ある分子と他の分子との相互作用を，平均化された相互作用として近似することで複雑なシステムやモデルを単純化する．

SA は，最適化問題を解決するよく知られた手法である．アニーリングは，金属がゆっくり冷却される冶金プロセスであり，分子が内部エネルギーを最小化する場所を見つける過程である．探索空間の各点は物理システムの状態に対応し，最小化すべき関数は物理状態の内部エネルギーに対応する．初期状態からランダムに周辺を探索し，より良い状態 (エネルギー関数の値が小さい) には常に遷移する．重要なアイデアは，より悪い状態も一定の確率で遷移を行う．この遷移確率は，温度が低くなるにつれて小さくなる．これにより，アルゴリズムは極小解から飛び出すことができ，最終的に最小解に収束する．

3. 関連研究

能動学習によるクラスタリング手法とマルウェアクラスタリングに関する関連研究を紹介する．

クラスタリングは教師なし学習の一般的な例だが，能動学習は通常，教師あり学習でラベル付けデータの数を減らすことを目的としている．情報量期待値規準に基づいたACは最初に [8] で提案された．論文 [18] も情報量期待値規準を適用し，階層的クラスタリングアルゴリズムを提案した．両方は，ACがランダムサンプリングよりも優れた性能を示した．

マルウェア検体を分析する一般的な方法は静的解析と動的解析である．静的解析は，検体を実行せずに情報を直接抽出する手法である．動的解析は，仮想マシンで検体を実行し，収集された動作ログデータで検体を分析する手法である．

我々の先行研究では，静的解析に基づいて FACCP を提案した [7]．検体間の類似性 (距離) は NCD を用いて計算した．FACCP では，マルウェアの系統樹を構築し，そして系統樹を適切なサブツリーに切ることでクラスターにする．FACCP は，距離の計算量を大幅に削減したため，高速でスケーラブルなアルゴリズムである．

Bailey らは [2] で動的解析に基づくスケーラブルなマルウェアクラスタリング手法を提案した．また，NCD を用いてすべての検体の動作ログデータペア間の距離を計算した．次に，最短距離法と矛盾係数を用いて階層的クラスタリングを行った．我々の方法の利点は，データ間の NCD 行列を全部計算する必要がないところである．

Bayer らもマルウェアの動的解析に基づくスケーラブルな階層的クラスタリング手法を提案した [3]．局所性鋭敏型ハッシュ (Locality Sensitive Hashing，LSH) を用いて距離行列の約 2%のみを計算してクラスタリングを行った．ただし，Bailey らと Bayer らの動的解析手法では，各マルウェア検体を実行して動作ログデータを収集するのは数分がかかる．一方，我々の手法ではマルウェア検体を実行する必要はない．

また，様々なサイバー攻撃の分析手法を統合したハイブリッドプラットフォームの構築を Takahashi ら [17] によって試みられている．その一環として本研究は，マルウェアの静的解析の課題ついて研究開発を進めており，他にも Kawasoe らが IoT 検体の関数呼び出しシーケンスグラフの作成による検体機能調査手法 [9] を提案した．本課題の目標は，Kawasoe らの手法と本研究のクラスタリング結果を融合し，マルウェア亜種の変遷に伴う機能の変化や進化を捉えることである．

4. アクティブクラスタリング

本節では，アクティブクラスタリング (AC) アルゴリズムを紹介する．このアルゴリズムは，クラスタリングとアクティブデータセレクションの 2 つの部分で構成される．クラスタリングアルゴリズムは，距離行列の現在の観測に基づいてクラスタリングを決定する．アクティブなデータセレクションは，現在のクラスタリング結果に基づいて次の観測箇所を決定する．初期化するため，最初に距離行列の 1%をランダムに観測する．アルゴリズムのダイアグラムを図 1 で示す．

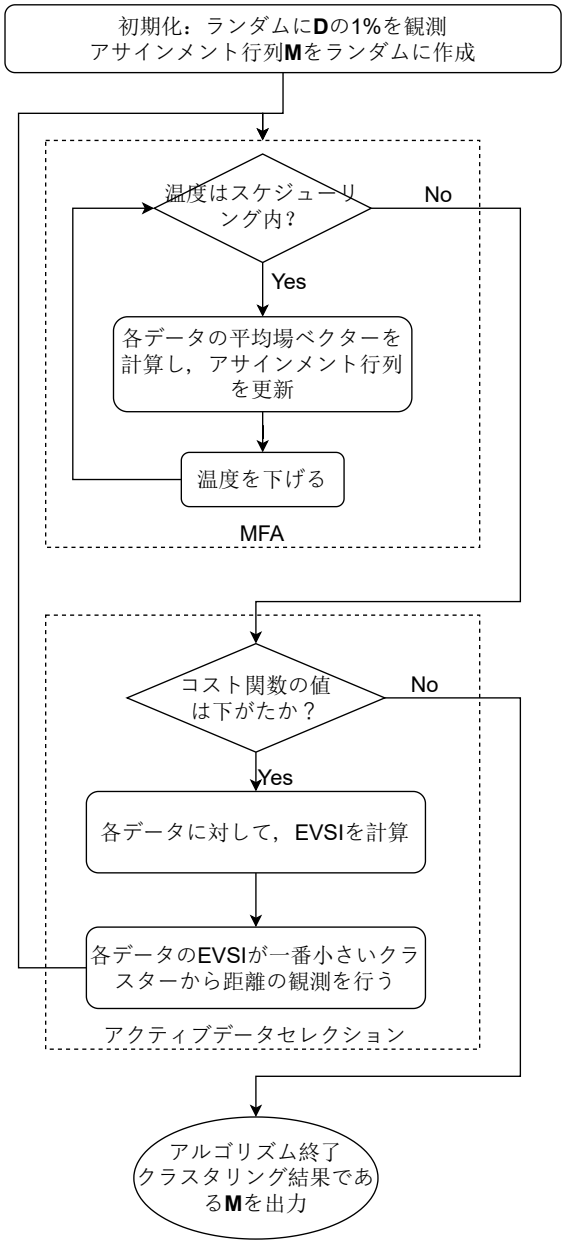


図 1 能動学習によるクラスタリングのダイアグラム

4.1 問題設定

データセットの距離行列は $\mathbf{D} \in \mathbb{R}^{N \times N}$ で表す．序論で述べたように，$\mathbf{D}$ のごく一部のみを観測する．$\mathbf{D}$ のほとんどの部分は不明のままになる．$\mathcal{N}_1$, ..., $\mathcal{N}_N$ を用いて観測された部分を記録する．D_{ij} が観測されたならば $j \in \mathcal{N}_i$. N 個のデータを K 個のクラスターにクラスタリングすることは，割り当て問題と見なすことができる．コスト関数を定義して割り当てを評価すれば，クラスタリングは組み合わせ最適化問題と見なしてコスト関数を最小化する割り

当てを探すことになる．

4.1.1 アサインメント行列

データの割り当ては行列に符号化するができる．論文 [18] の階層的クラスタリングモデルは，二分木に基づいたクラスタリングであり，すべての検体は葉ノードに割り当てられる．葉ノードをクラスターと見なして，N 個のデータを葉クラスターにクラスタリングすることは，割当問題と見なすことができる．K をバイナリツリー T の深さ (ルートノードの深さは 0) とすると，葉ノードの数は 2^K になり，ノードの総数は 2^{K+1} になる．クラスタリングは，アサインメント行列 $\mathbf{M} \in \{0,1\}^{N\times(2^{K+1}-1)}$ で表すことができる．ここで，$M_{ij}=1$ は，i 番目のデータが j 番目のクラスターにあるを意味する．内部ノードもクラスターとして扱われる．内部ノードクラスターに含まれるデータはそのノードの下の葉ノードに属する全てのデータである．$\mathbf{M}$ の 2^k 番目の列から $(2^{k+1}-1)$ 番目の列は，バイナリツリーの k 層目のノードに対応する．$\mathbf{M}$ には次のような性質がある：

1) すべてのデータを 1 つのクラスターに割り当てる必要がある

$$\forall i,\ \sum_{j=2^K}^{2^{K+1}-1} M_{ij} = 1.$$

2) ノードの割り当ては親ノードに継承される

$$\forall i,\ \sum_{j=1}^{2^K-1} M_{ij} = M_{i(2j)} + M_{i(2j+1)}.$$

3) ルートノードは全てのデータを含む

$$\sum_{i=1}^{N} M_{i1} = N.$$

4) 各クラスターに少なくとも一つのデータを含む

$$\forall j,\ \sum_{i=1}^{N} M_{ij} \geq 1.$$

4.1.2 コスト関数

アサインメント行列の評価するために, 我々は論文 [13], [18] のコスト関数を用いた：

$$H(\mathbf{M},\mathbf{D},\mathcal{N},K) = \sum_{k=1}^{K} W(k) \sum_{i=1}^{N} \sum_{v=2^k}^{2^{k+1}-1} M_{i\nu} d_{i\nu}, \quad (1)$$

$$\text{where } d_{i\nu} = \frac{\sum_{j\in\mathcal{N}_i} M_{j\nu} D_{ij}}{\sum_{j\in\mathcal{N}_i} M_{j\nu}}. \quad (2)$$

ここで，$d_{i\nu}$ は，i 番目のデータとクラスター ν の間の距離である．H は，重み W 付きのクラスター内の距離の総和の総和である．H はクラスターの緊密さを評価する指標であり，コスト関数が小さいほど，クラスタリングが優れている．

Algorithm 1: MFA

Input: 温度スケジュールベクター: T
Output: $\mathbf{M} \in \{0,1\}^{N\times 2^{K+1}-1}$

初期化: アサインメント行列をランダムに作成 M
for *t from 1 to length(T)* **do**
  while コスト関数が下がってる do
    i 番目のデータをランダムに選ぶ
    i 番目のデータの平均場ベクターを計算:
    $\phi_{ik} = -\frac{\partial H}{\partial M_{ik}}$ (k は 2^K から $2^{K+1}-1$ まで)
    アサインメント行列を更新: $M_{ik} = \frac{e^{\phi_{ik}/T_t}}{\sum_{n=2^K}^{2^{K+1}-1} e^{\phi_{in}/T_t}}$
  end
  t=t+1
end
return($\mathbf{M}$)

4.2 クラスタリング

クラスタリングアルゴリズムは，組み合わせ最適化問題と見なしてコスト関数 H を最小化するアサインメント行列 $\mathbf{M}$ を探す．最適化アルゴリズムは MFA を用いた．

MFA の擬似コードはアルゴリズム 1 に示されている．M_{ij} は，冷却中に 0 から 1 までの任意の実数値をとることができる．このとき，M_{ij} はクラスター j に i 番目のデータを割り当てる確率を表す．温度が下がり，システムが安定した後，M_{ij} は 0 または 1 に収束する．

4.3 アクティブデータセレクション

MFA が距離行列の現在の観測に基づいて最適なアサインメント行列を決定した後，次にどこを観測するかを決定する必要がある．アクティブデータセレクションは，情報量期待値規準に基づき情報量の最も多いところを観測する．情報量期待値規準はベイジアンアプローチであり，論文 [11] で提案された手法である．これは [8], [18] で適用された手法でもある．仮にすべての距離が観測された場合，データ i とクラスター ν の間の距離は次の式で与えられる：

$$d^*_{i\nu} = \frac{\sum_{j=1}^{N} M_{j\nu} D_{ij}}{\sum_{j=1}^{N} D_{ij}}.$$

ただし，距離行列を全部観測しないので，$d^*_{i\nu}$ は計算することができない．したがって，$d^*_{i\nu}$ を確率変数と見なし，その点推定量 $d_{i\nu}$(式 (2)) を用いる．$d^*_{i\nu}$ の事前分布を無情報事前分布と仮定する， $d_{i\nu}$ は標本平均，$\sigma^2_{i\nu}$ は標本分散，$m_{i\nu}$ は標本数とする．$d^*_{i\nu}$ の周辺事後分布は，スチューデント t 分布である：

$$t = \sqrt{m_{i\nu}}(d^*_{i\nu} - d_{i\nu})/\sigma_{i\nu}.$$

したがって，$d^*_{i\nu}$ は:

$$d^*_{i\nu} = d_{i\nu} - t * \frac{\sigma^2_{i\nu}}{m_{i\nu}}$$

で表せる．

$d^*_{i\nu}$ の確認密度関数は $f_{i\nu}(d^*_{i\nu}|d_{i\nu},\sigma_{i\nu},m_{i\nu})$ で表すことができる．*Expected Value of Perfect Information* (EVPI) [8] は以下のように定義する:

$$\begin{aligned}\mathrm{EVPI} = &\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}\max_{\nu}\{d^*_{i\alpha}-d^*_{i\nu}\}\\ &\times\prod_{\nu=2^K}^{2^{K+1}-1} f_{i\nu}(d^*_{i\nu}|d_{i\nu},\sigma^2_{i\nu},m_{i\nu})d\,d^*_{i\nu}.\end{aligned} \tag{3}$$

ここで，$\alpha=\arg\min_\nu d_{i\nu}$．EVPI は最適な決定 α^* ではなく，不完全な情報 $d_{i\nu}$ に基づいて決定 α を行う際のロスを測る指標である．

しかし，実際で完全な情報を得ることができないので我々が関心を持っているのは *Expected Value of Sampling Information* (EVSI) である．EVSI は，次のデータを観察した後，どの程度の不確かさが残ってるかを測定する指標である．つまり，EVPI から EVSI を引いた値は，次の観測データから期待される情報量である．観測は ν 番目のクラスターから距離を $m^+_{i\nu}$ 回のサンプリングとする．最も得られる情報量が多いクラスターから観測を行う．クラスター ν から $m^+_{i\nu}$ 回サンプリングするのは，不偏推定量 $d_{i\nu}$ と $\sigma^2_{i\nu}$ に影響しない，サンプルサイズ $m_{i\nu}$ を増やすだけ．したがって，i 番目のデータに対して，クラスター j から観測を行う場合の EVSI は，次の式で与えられる:

$$\begin{aligned}\mathrm{EVSI}_{ij} = &\int_{-\infty}^{\infty}\dots\int_{-\infty}^{\infty}\max_{\nu}\{d^*_{i\alpha}-d^*_{i\nu}\}\\ &\times f_{ij}(d^*_{ij}|d_{ij},\sigma^2_{ij},m_{ij}+m^+_{ij})\\ &\times\prod_{\substack{\nu=2^K\\ \nu\neq j}}^{2^{K+1}-1} f_{i\nu}(d^*_{i\nu}|d_{i\nu},\sigma^2_{i\nu},m_{i\nu})d\,d^*_{i\nu}\,d\,d^*_{ij}.\end{aligned} \tag{4}$$

式 (4) の積分を計算するために，モンテカルロ法を用いてスチューデント t 分布からのサンプリングすることで近似する．データ i に対して，各リーフクラスターに式 (4) を計算し，値が最小のクラスターから距離を観測する．

AC の擬似コードは，アルゴリズム 2 に示す．

4.4 オンライン処理

クラスタリングが完了したあと，新しい検体が収集され，クラスターに加えたい場合，新しい検体をデータセットに加えてクラスタリングをやり直すのは計算量が多い．そこで，オンライン処理のアルゴリズムを提案する．新しい検体の数が元のデータセットの検体数より少ない場合，クラスタリングアルゴリズムをもう一回行うではなく，直接クラスターにクラスタリングする．

オンライン処理は新しい検体とクラスター間の距離を計算し，各検体をそれぞれ一番近いクラスターにクラスタリングする．検体とクラスター間の距離は検体とクラスターの中心検体の間の NCD で計算する，中心検体はクラスター内の距離行列で列要素の和の平均が一番小さい検体で定義する．

Algorithm 2: アクティブクラスタリング

Input: $N\times N$ の未知の距離行列 $\mathbf{D}$, バイナリツリーの深さ K

Output: $\mathbf{M}\in\{0,1\}^{N\times 2^{K+1}-1}$

$\mathcal{N}=(\mathcal{N}_1,\dots,\mathcal{N}_N)$. $j\in\mathcal{N}_i$ は D_{ij} が観測済みを意味する．
初期化: ランダムに $\mathbf{D}$ の 1%を観測 ($\mathcal{N}$ はそれに応じて変わる)
while コスト関数が下がっている **do**
  MFA を用いてコスト関数を最小化する $\mathbf{M}$ を求む．
  アクティブデータセレクション:
  for *i from 1 to N* **do**
    $\nu=\arg\min_\nu \mathrm{EVSI}_{i\nu}$
    data i とクラスター ν に含まれるデータ間の距離を $m^+_{i\nu}$ 回観測する
  end
end

表 1 データセットのマルウェア科名及びアーキテクチャの内訳

アーキテクチャ	Malware Family		Total
	Bashlite	Mirai	
ARM	195	161	356
MIPS	194	165	359
Intel	197	183	380
x86	189	157	346
PowerPC	197	171	368
Renesas	200	199	399
SPARC	200	200	400
Motorola	200	200	400
Total	1572	1436	3008

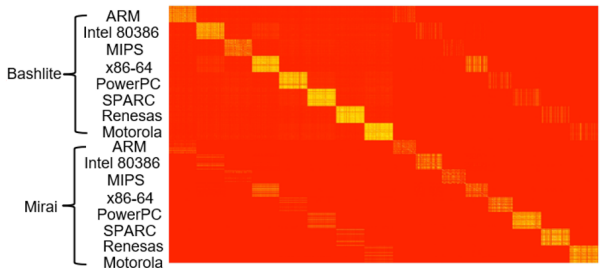
オンライン処理は 10 分割交差検証で評価する，9 割のデータでクラスタリングを行い，残りの 1 割を用いてオンライン処理を行って，ファミリー名とアーキテクチャ名の正解率を評価する．

5. 実験結果とその評価

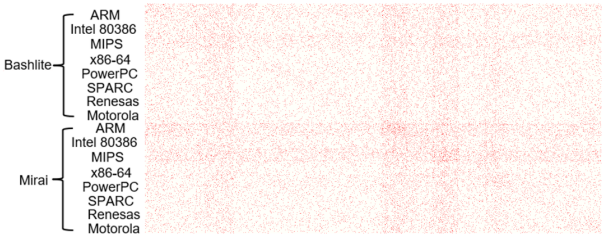
このセクションでは，IoT マルウェアを用いて AC を評価するための実験を紹介する．ベンチマークとして，ランダムサンプリングと MFA を用いた．さらに，距離行列の観測割合とクラスタリングの精度を FACCP [7] と比較した．

5.1 データセット

VirusTotal から収集された 3,008 の Linux マルウェア検体 (主に IoT マルウェア) をデータセットに用いた．収集期間は 2018 年 11 月から 2019 年 2 月までである．データセットは Bashlite と Mirai で構成されていて，内訳は表 1 で示している．マルウェア科名は AVClass [15] によって決定された．AVClass は，VirusTotal の JSON レポートを使用してマルウェア検体の科名を多数決で決める Python



(a) 距離行列のヒートマップ



(b) サンプリングスナップ写真

図 2 距離行列のヒートマップ及び距離行列の観測場所

ツールである．

5.2 実験のセットアップ

プログラミング言語 Julia を用いてアルゴリズムを実装した．Julia は動的型付き言語であり，C 言語に近い性能を持っている．つまり，Julia の書きやすさは R や python に近いが，R や python よりも桁違いに速い場合が多い．Linux の xz コマンド (バージョン 5.1.0α) をマルウェアバイナリの圧縮アルゴリズムに用いた．xz は，マルコフチェーンに基づいた圧縮アルゴリズム (LZMA) [14] を使用している．我々の計算環境の CPU は 2.6GHz Intel-Xenon-Gold-6126 である．アクティブデータセレクションにおいて，モンテカルロ法の計算は 80 スレッドで並列計算した．

5.3 評価指標

我々は 3 つの評価指標を用いて AC を評価した．

(1) **コスト関数**：距離行列の観測割合が増加するにつれて，コスト関数の値がどのように変化するかを比較した．

(2) **ファミリー名のクラスタリング正解率**：クラスタリング正解率は，10 分割交差検証で評価した．具体的には，データセットを 10 分割し，90%の検体を使用してクラスタリングを行う．次に，残りの 10%の検体をそれぞれ最も近いクラスターに割り当てる．クラスターのファミリー名は，そのクラスター内の検体のファミリー名で多数決で決める．ファミリー名の正解率は，正しくクラスタリングされた検体の割合で計算する．

(3) **アーキテクチャ名のクラスタリング精度**：アーキテクチャ名のクラスタリングの精度は，ファミリ名のクラスタリングの精度と同じように計算する．

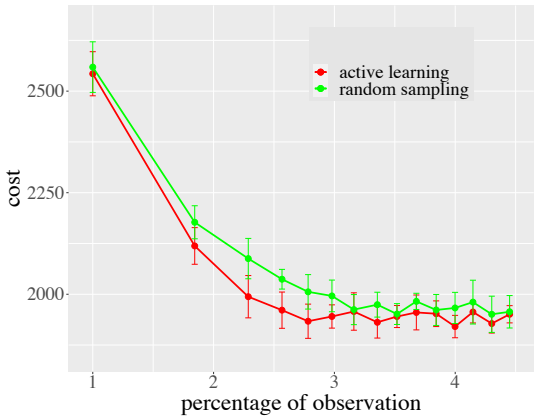


図 3 アクティブデータセレクションとランダムサンプリングのコスト関数

5.4 実験結果

図 2(a) は距離行列を全部計算したヒートマップであり，赤いほど距離が大きい (類似性が低い)．図 2(b) は，アサインメント行列 **M** のヒートマップであり，距離行列の約 3%を観測した後の観測位置を示している (観測済みが赤，未観測が白)．図 2(a) と図 2(b) はどちらも対称行列であり，それらのデータは同じ順序である．アクティブデータ選択アルゴリズムは，主にクラスター間の類似性が低いクラスターから観測を行った．これは，類似性の高いクラスターは早い段階でクラスターを形成し，これ以上距離を観測する必要がないことを意味する．

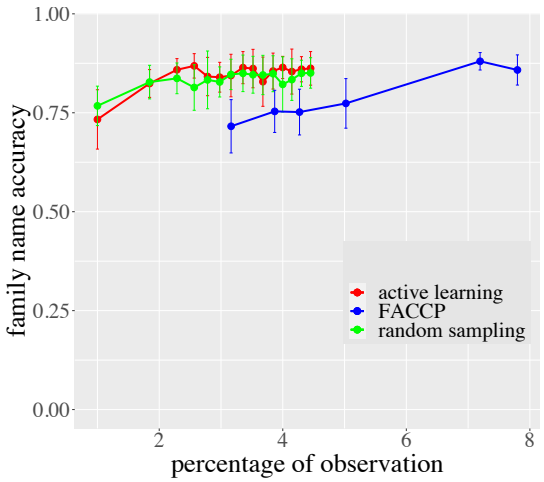
5.4.1 コスト関数の評価

アクティブデータセレクションとランダムサンプリングの観測量によるコスト関数の変化を図 3 に示す．10 分割交差検証を用いて，10 回の結果の平均値を示している，エラーバーで結果の標準偏差を示している．両方のコスト関数は，完全なデータ (100%の観測) で評価されている．アクティブデータセレクションのコスト関数は，ランダムサンプリングよりも約 20%速く収束した．つまり，アクティブデータセレクションアルゴリズムでは，距離行列の観測を 20%削減することができる．

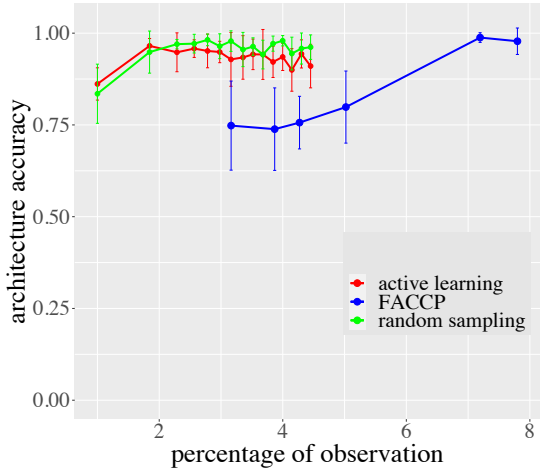
5.4.2 クラスタリング正解率

10 分割交差検証の結果を図 4 に示す．エラーバーで 10 回の結果の標準偏差を示している．赤は AC を示していて，緑は MFA とランダムサンプリングを示している，両者は同じレベルのクラスタリング正解率を示した．FACCP と比較して，AC は 3%台の観測量においてはるかに高い正解率を達成した．言い換えれば，FACCP の半分未満の観測量で同じレベルの正解率を達成できた．

実験結果により，AC がベンチマークであるランダムサンプリングよりも約 20%少ない距離を観察し，同じレベルのクラスタリング正解率を達成したことを確認できた．

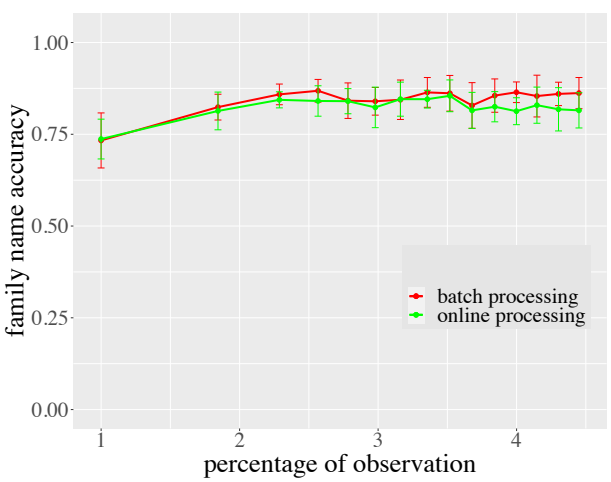


(a) ファミリー名正解率

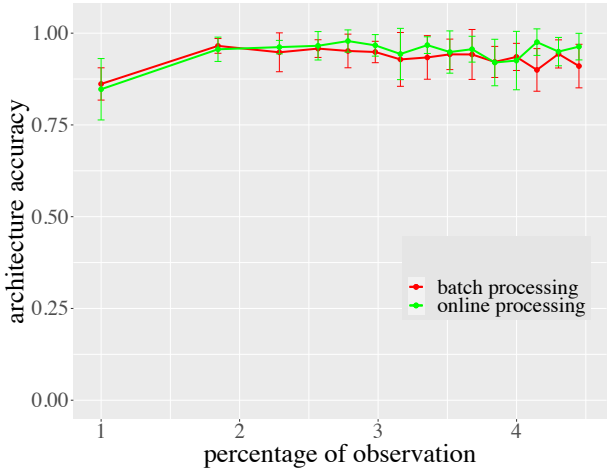


(b) アーキテクチャ正解率

図 4 クラスタリング正解率



(a) ファミリー名正解率



(b) アーキテクチャ正解率

図 5 オンライン処理正解率

FACCP と比較して，AC は同程度の精度を保ちながら，NCD 行列の観察量は 7.2%から 2.6%に削減した．観測削減率は (7.2-2.6)/7.2=64%だった．

5.4.3 オンライン処理の評価

オンライン処理の正解率とバッチ処理の正解率を図 5 で比較した．その結果，両者は同程度の正解率を達成した．計算量について，オンライン処理は一つの検体に対して，距離の計算回数はクラスター数と一致する．今回の実験において，クラスター数は 16，オンライン処理する検体数は 300，この場合距離行列の観測率は 16*300/(2708*300+300*299/2) = 約 0.6%であった．計算時間はもちろんクラスタリングアルゴリズムをやり直す必要がないのでバッチ処理より桁違いに早い．

6. 考察

アクティブデータセレクションアルゴリズムは，FACCP より NCD の計算量を 64%を削減したが，MFA の実行時間は非常に長いため，AC は FACCP よりもはるかに遅くなっている．現在，AC の実行時間は約 7 時間だが，FACCP の実行時間は約 30 分である．アクティブデータ選択アルゴリズムは優れた性能を示したが，最適化アルゴリズム MFA がアルゴリズム全体の弱みになっている．したがって，今後の予定では，より高速で並列化可能な最適化アルゴリズムを適用することで，この不足を補う．

7. まとめ

本論文では，AC を 3,008 検体の IoT マルウェアを用いて評価実験を行った．我々の実験では，AC はに NCD 計算回数を 97.4%削減し，ファミリ名正解率が 86.9%，アーキテクチャ名正解率が 96.5%を達成した．AC は，FACCP よりも 64%の NCD 計算量を削減した．大規模なマルウェア検体セットにも対応できるために，今後の予定では，MFA の代わりに遺伝的アルゴリズムなどの高速で並列化可能な最適化アルゴリズムを適用し，より高速なアルゴリズムを実現することを目指してしている．

謝辞 本研究は総務省の「電波資源拡大のための研究開発 (JPJ000254)」における委託研究「電波の有効利用のための IoT マルウェア無害化／無機能化技術等に関する研究開発」によって実施した成果を含む．

参考文献

[1] Antonakakis, M., April, T., Bailey, M. and et al.: Understanding the Mirai Botnet, *Proceedings of the 26th USENIX Conference on Security Symposium*, pp. 1093–1110 (2017).

[2] Bailey, M., Oberheide, J., Andersen, J. and et al.: Automated Classification and Analysis of Internet Malware, *Recent Advances in Intrusion Detection, 10th International Symposium, RAID*, pp. 178–197 (2007).

[3] Bayer, U., Comparetti, P. M., Hlauschek, C., Krügel, C. and Kirda, E.: Scalable, Behavior-Based Malware Clustering, *Proceedings of the Network and Distributed System Security Symposium, NDSS* (2009).

[4] Black Lotus Labs: Attack of Things!, `https://www.netformation.com/our-pov/attack-of-things-2/`.

[5] Cebrian, M., Alfonseca, M. and Ortega, A.: The Normalized Compression Distance Is Resistant to Noise, *IEEE Transactions on Information Theory*, Vol. 53, No. 5, pp. 1895–1900 (2007).

[6] Cilibrasi, R. and Vitanyi, P. M. B.: Clustering by compression, *IEEE Transactions on Information Theory*, Vol. 51, No. 4, pp. 1523–1545 (2005).

[7] He, T., Han, C., Isawa, R. and et al.: A Fast Algorithm for Constructing Phylogenetic Trees with Application to IoT Malware Clustering, *Neural Information Processing - 26th International Conference, ICONIP*, pp. 766–778 (2019).

[8] Hofmann, T. and Buhmann, J.: Active Data Clustering. (1997).

[9] Kawasoe, R., Han, C., Isawa, R., Takahashi, T. and Takeuchi, J.: Investigating behavioral differences between IoT malware via function call sequence graphs, *Proceedings of the 36th Annual ACM Symposium on Applied Computing* (2021).

[10] Li, M., Chen, X., Li, X., Ma, B. and Vitányi, P. M. B.: The similarity metric, *IEEE Trans. Information Theory*, Vol. 50, No. 12, pp. 3250–3264 (2004).

[11] Minton, P., Raiffa, H. and Schlaifer, R.: Applied Statistical Decision Theory., *American Mathematical Monthly*, Vol. 69, p. 72 (1962).

[12] MaireO' Neill：Insecurity by Design: Today's IoT Device Security Problem, *Engineering*, Vol. 2, pp. 48–49 (2016).

[13] Puzicha, J., Hofmann, T. and Buhmann, J. M.: A theory of proximity based clustering: structure detection by optimization, *Pattern Recognition*, Vol. 33, No. 4, pp. 617–634 (2000).

[14] Salomon, D.: *Data compression - The Complete Reference, 4th Edition*, Springer (2007).

[15] Sebastián, M., Rivera, R., Kotzias, P. and Caballero, J.: AVclass: A Tool for Massive Malware Labeling, *Research in Attacks, Intrusions, and Defenses*, Springer International Publishing, pp. 230–253 (2016).

[16] Settles, B.: Active Learning Literature Survey, Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009).

[17] Takahashi, T., Umemura, Y., Han, C., Ban, T., Furumoto, K., Nakamura, O., Yoshioka, K., Takeuchi, J., Murata, N. and Shiraishi, Y.: Designing Comprehensive Cyber Threat Analysis Platform: Can We Orchestrate Analysis Engines?, *2021 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)* (2021).

[18] Zoller, T. and Buhmann, J.: Active learning for hierarchical pairwise data clustering, *Proceedings 15th International Conference on Pattern Recognition. ICPR*, Vol. 2, pp. 186–189 vol.2 (2000).