機械学習によるマルウェア検出　リローデッド

笹生　憲 †　　村上純一 ‡　　松木隆宏 ‡　　森 達哉 †

† 早稲田大学　基幹理工学部
169-8555　東京都新宿区大久保 3-4-1
{saso,mori}@nsl.cs.waseda.ac.jp

‡ 株式会社 FFRI
150-0013　東京都渋谷区恵比寿 1-18-18
{murakami,matsuki}@ffri.jp

あらまし 本研究は，実行ファイルの PE ヘッダから静的に得られるありとあらゆる情報を機械学習を適用することにより，どこまで検知率を高めることができるかという **Research Question** を課し，実データを用いた実験的な検証を行う．また，数多くの特徴から識別に貢献しない特徴を削除した後，特に検出に貢献した度合いが高いと考えられる特徴の分析を行う．本研究の貢献は PE ヘッダから得られる数値情報やカテゴリカルな情報等の様々なデータ抽出を自動化したこと，およびこのような高次元の特徴に対してスパース学習アルゴリズムを適用することで，高精度なマルウェア検知および特徴選択の自動化が可能であることを示したことにある．さらに本研究ではマルウェア識別手法を評価するデータセットの選択が性能に与える影響をデータソースの違いやパッカー有無の観点で検証した結果を報告する．

Detecting Malware with Machine Learning Reloaded

Akira Saso†　　Junichi Murakami‡　　Takahiro Matsuki‡　　Tatsuya Mori†

†Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, JAPAN
{saso,mori}@nsl.cs.waseda.ac.jp

‡FFRI Inc.
1-18-18 Ebisu, Shibuya-ku,Tokyo 150-0013, JAPAN
{murakami,matsuki}@ffri.jp

Abstract We propose a novel approach to detecting malware samples accurately with the static information collected from malware samples. We leverage vast amount of information available on PE headers by automating the process of extracting various kinds of fields that include numerical ones and categorical ones. For each malware sample, we succesfully extracted 20K of distinct features. By applying the sparse machine learning algorithm to the high dimensional features, we demonstrate that our approach establishes the best performance among the other approaches that fully make use of the knowledge of experts. We also demonstrate that we extracted primary features that mostly contributed to classify the binary executable files. The extracted features include both the existing popular ones such as timestamp and the new ones that have not been discovered by experts. We also investigate how the selection of data affects the results by analyzing samples obtained through different sources and packed/non-packed samples.

1　はじめに

「アンチウィルスソフトウェアは死んだ」
– ブライアン・ダイ，シマンテック上席副社長 [1]

上記はウォール・ストリート・ジャーナルに掲載された記事の抜粋であるが，世界的な大手セキュリティ企業の執行役員によるセンセーショナルな発言はセキュリティ業界にとどまらず広範にわたって人々の関心を引いたことは記憶に新しい．同記事ではマルウェア攻撃の実に 55% がアンチウィルスソフトの検知をかいくぐることを述べ，従来型の「守る」ことから「検知と対応」へのシフトが必要であると締めている．

本研究は従来型技術の中核であるマルウェア検知は本当に「死んだ」技術なのか，検知技術を向上させる余地はもう残っていないのかという **Research Question** に対して実証的な方法でアプローチする．具体的には静的情報を用いたマルウェア検知を対象とし，検体から収集可能な**あらゆる情報**を組み合わせることにより，検知率がどこまで向上するかという問題に取り組む．ここで検体から抽出する情報はマルウェア解析のエクスパートが発見したマルウェア検知に有用な経験則に加え，PE ヘッダに記載された情報およびそれらの情報から計算可能な数値等を指し，機械学習によって適切に特徴選択を行うことで精度の向上を図る．

本研究で抽出した検体あたりの特徴数は約 2 万 5000 に上り，高次元である．識別に最適な特徴の選択をナイーブな方法で行うと組み合わせ爆発の問題がある．本研究ではスパース制約付きの学習アルゴリズムを適用することにより，識別に有用な特徴の抽出と識別関数の最適化を同時に行うアプローチを採用する．このような識別に有用な特徴抽出の自動化により，従来エクスパートによって発見された既知の特徴に加え，これまでに知られていない新たに特徴を発見することを狙いとする．提案手法の評価においては検知の対象となるマルウェア検体の偏りや，パッカーの存在有無が検知率に与える影響を考慮し，既存アプローチとの比較を行う．

本研究の主要な貢献は下記のとおりである．

- 静的情報を用いたマルウェア検知において，エクスパートが経験的に発見してきた特徴に加え，有用な特徴を自動的に抽出できることを明らかにした．
- スパース学習により，高い精度を維持しつつ特徴数を 5% まで削減できることを示した．
- 既存アプローチと比較して精度向上が可能であることを示した．
- 提案方法はパッカーの有無によらず有効であることを示した．
- 既存方法はデータの偏りにセンシティブであるのに対し，提案方法はランダムなデータに対しても良好な性能が得られることを示した．

本研究は冒頭で掲げた **Research Question** のごく一部に取り組んだにすぎないが，上記の発見はマルウェア検知技術はまだ向上の余地が多いにあることを示唆する証左であると我々は確信している．

本論文の構成は以下の通りである．はじめに 2 章で関連研究を述べる．次に 3 章で特徴抽出方法を述べ，4 章で機械学習の方法を述べる．5 章では提案手法の評価に用いたデータの詳細を示し，6 章で提案手法の評価結果を示す．7 章で本研究の制限や今後の課題を述べた後，8 章にて本論文をまとめる．

2 関連研究

本章では，マルウェアの静的解析に関連するいくつかの研究を述べる．マルウェアの識別では，静的解析と動的解析の主に 2 つのアプローチがなされる．静的解析の中でも，バイナリをそのまま扱う手法，PE Structure を扱うもの，機械語を扱うもので大別される．

Ye らは文献 [2] で HAC という手法を提案し，正常系とマルウェアそれぞれ約 800 万検体の分類を行った．実行ファイルにインポートされる API を特徴とし，階層的アソシエーション分類 (Hierarchical Associative Classifier) を行った．API の情報のみで高精度の識別器が生成できると同時に，非常に高速に識別を行えることを示した．Shafiq らは文献 [3] で，PE の構造をもとに特徴抽出を行う PE-Miner というフレームワークを提案している．PE-Miner では PE file header [4] に基づいてヘッダーに格納されている値とインポートされている DLL を特徴として分類を行う．高い識別精度を持ちデータセットのパッキングの有無に関しても，頑強であることが示されている．SANS　は文献 [5] においてマルウェア 250 万検体と正常系 6500 検体を用いて経験的に導いたマルウェア特有の特徴を報告している．調査において特に重要であった特徴を 28 個挙げており，その抜粋を表 2 に示す．

3 PE フォーマットファイルの特徴抽出

本節でははじめに PE フォーマットの概要と解析方法を述べた後，既存方法における特徴抽出，および本研究で提案する特徴抽出を示す．

3.1 PE フォーマットの概要と解析方法

PE ファイルフォーマットはヘッダとセクションで構成される．ヘッダはローダがプログラムをロードするために必要な情報が格納されており，セクションにはヘッダ，プログラムコード，リソースデータなどが格納されている．ヘッダはさらに DOS スタブ，PE ヘッダ，セクションテーブルなどにより構

表 1: 抽出した特徴のカテゴリと特徴数.

カテゴリ	特徴数
Imported symbols	18,753
Exported symbols	4,037
PE Sections	836
Delay Imported symbols	747
Parsing Warnings	544
Debug information	273
Resource directory	123
Bound imports	111
OPTIONAL_HEADER	59
Directories	32
FILE_HEADER	25
LOAD_CONFIG	25
DOS_HEADER	24
Version Information	16
Flags	15
DllCharacteristics	8
TLS	6
Base relocations	2
NT_HEADERS	1
SANS [5]	28
合計	25,665

成される．本研究では PE ファイルフォーマットのパーサーとして広く使われている pefile [6] を利用する．pefile は PE ファイルフォーマットの構造体の値に加え，section 毎のエントロピーや疑わしい値や誤ったフォーマットのデータに関する警告などの情報を抽出する．例えば，あるセクション領域に書き込みと実行が同時に可能な場合には，パッキングされた実行ファイルである可能性があるという警告が得られる．本研究における特徴の表記は，pefile が採用する表現を利用する．

3.2 既存方法における特徴抽出

HAC [2] はインポートされた API 名の出現を出現の有無に応じて二値の特徴としている．これらは後述する特徴に包含される．PE-Miner [3] は，特定の DLL のインポート有無，各種ヘッダーの値，リソースに関する情報を特徴としている．これらもいずれも後述する特徴に包含される．SANS [5] は 2 章で示したように経験則に基づいた 28 の特徴を採用している．これらは後述する特徴に包含されないため，本研究では後述の特徴に追加する形で SANS の特徴を利用する．

3.3 提案手法の特徴抽出

本研究では既存の特徴として SANS [5] で採用している特徴に加え，特徴を独自に抽出する．抽出した特徴は HAC [2]，PE-Miner [3]，SANS [5] を包含する．本研究ではヘッダに記載された値をすべて特徴に組み込むと同時に，セクションの情報も情報を加工することで特徴に組み込んだ．例えば，resource section では，リソースの ID 毎に格納されているリソースの数を特徴として抽出する．前述したように PE ファイルのパーサーに pefile を用いる． 以下では pefile の出力から特徴を抽出する方法を述べる．

表 2: SANS [5] で提案されている特徴の抜粋．いずれも条件に対して真偽の二値 $(1, 0)$ となる．

	Detection Rule
FILE HEADER	timestamp Year < 1992 or Year > 2014
FILE HEADER	NumberOfSections > 9
FILE HEADER	PtrToSymTable > 0
FILE HEADER	Characteristics (BYTE_RESERVED_LO=1)
OPTIONAL HEADER	SizeOfUninitializedData / Sample Size >1
SECTIONS	Raw Size = 0
SECTIONS	PtrToLineNumber ! = 0
SECTIONS	max(Section Entropy) > 7.0

pefile で得られた数値データは基本的にそのまま数値として利用する，ただしすべての数値は $[0, 1]$ のスケールに変換する．Import される API 名などカテゴリカルな値はその値がファイル中に出現したか否かの二値で特徴を表現する．FILE_HEADER の Characteristics のような論理積を取ることで意味をもつ値に関しては，それぞれの意味を表現するビットが立ってる否かを二値の特徴とする．

表 1 に本研究で収集したすべてのデータに対して pefile で抽出した特徴のカテゴリと特徴数および SANS [5] で採用している特徴の数を示す.

4 機械学習

本章では既存研究で提案された特徴および本研究で提案する特徴に対して機械学習を適用する手順・方法を示す．既存方法に関してはオリジナルの方法よりも精度があがるように機械学習アルゴリズムを適用する．特に SANS [5] は個々の特徴の有効性を独立に検証しているのみで組み合わせによる識別を考慮していないため，機械学習を適用することによりオリジナルのナイーブな方法と比較して飛躍的に精度が向上している．

4.1 既存方法に対する学習

既存方法の内，PE-Miner と SANS については特徴ベクトルの次元数が低いため，識別能力の高さ

で定評がある非線形 SVM を適用する．紙面の関係で非線形 SVM の詳細は省略するが，識別モデルのパラメタ C, γ は訓練データに対する 10-times 10-fold CV とグリッドサーチにより最適な数値を決定する．SVM のカーネル関数としては RBF を用い，正則化条件は特につけない．SVM の実装として LIBSVM [7] を用いた．HAC に関しては次元が高く，非線形 SVM の計算コストが非常に高いこと，および次元削減による特徴選択の効果を期待して後述するスパース学習を適用する．

4.2 スパース学習

本研究では特徴の大多数が $x \in \{0, 1\}$ の二値変数であること，および特徴数 m がサンプル数 N に対して大きいことから，識別モデルとして L1 正則化付きのロジスティック回帰を採用する．L1 正則化により，識別に貢献しない特徴を大幅に削減することが可能である．すなわち，非常に多数の特徴から有効なものを経験的に決めるのではなく，識別精度を高めるものを自動的に抽出することを狙いとする．以下では簡単に L1 正則化付きのロジスティック回帰の定式化を示す．

ロジスティック回帰は入力 $\mathbf{x}^{(i)} = \{x_1^{(i)}, \ldots, x_m^{(i)}\}$ $(i = 1, \ldots, N)$ に対して二値の出力 $y \in \{+1, -1\}$ を下式にしたがって出力する識別モデルである．

$$y = \mathrm{sgn}\left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})} - 0.5\right)$$

ここで $\mathbf{w} = \{w_1, \ldots, w_m\}$ は各特徴の重み係数であり，sgn は引数が正数のときに $+1$ を，負数のときに -1 を出力する符号関数である．ロジスティック回帰モデルの学習は特徴ベクトルとラベルのペア $\left(\mathbf{x}^{(i)}, y^{(i)}\right)$ $(i = 1, \ldots, N)$ に対して誤差が最小となるように重み係数 $\mathbf{w}$ を最適化することである．L1 正則化条件を加える事により，$\mathbf{w}$ の要素を極力ゼロにするようなスパースな解を得る効果がある．最終的な最適化問題は下式で与えられる．

$$\widehat{\mathbf{w}} = \mathrm{argmin}_{\mathbf{w}} \sum_{i=1}^{N} \log(1 + \exp(-y^{(i)}\mathbf{w}^T\mathbf{x}^{(i)})) + C\sum_{j=1}^{m} |w_j|$$

正則化項の係数 C はクロスバリデーション (CV) とグリッドサーチにより最適値を求める．10-Fold CV は乱数シードを変えて 10 回行い，グリッドサーチでは C が取りうる区間を 100 個に分割し，それぞれの数値に対して最良の平均精度が得られる点を選択する．すなわち，パラメタを決定するために異なる訓練データ，評価データおよびパラメタの組み合わせを 10,000 回繰り返した．L1 正則化付きロジスティックの最適解を求める実装として本研究は LIBLINEAR [8] を用いた．

表 3: 収集した検体の統計．

検体集合	mix	packing 有り	packing 無し
B1 (ランダム)	2435	1006	1429
B2 (Windows)	647	-	-
M (ランダム)	2988	991	1997

表 4: 評価するデータの組み合わせ．

	組み合わせパターン	学習	テスト
D1	B1 (mix) + M (mix)	4882	541
D2	B2 (mix) + M (mix)	3272	363
D3	B1 (packing 有) + M (Packing 有)	1797	200
D4	B2 (packing 無) + M (Packing 無)	3083	343

5 評価に用いるデータセット

本章では，本論文で扱う正常系およびマルウェア検体の説明と評価に利用したデータの組み合わせを説明する．表 3 は検体の概要をまとめたものである．以下に各々のカテゴリーの詳細を示す．正常系の検体は比較的最近にネットワーク上から集めた様々な種類の実行ファイル 2,435 検体 (B1) と，Windows 7 に初期にインストールされているうちの実行ファイル 647 検体 (B2) を利用する．マルウェアの検体としては，様々なソースで収集した大規模な検体セットからランダムに 2988 検体を抽出した検体 (M) を利用する．さらに，B1 と M に関しては様々なツールやヒューリスティックを組み合わせ，パッカーの適用の有無を判定した結果を利用する．

評価に利用したデータセットを組み合わせを表 4 に示す．各データセットに対して訓練データとテストデータを 9:1 で分割し評価を行った．

6 結果

本章では，提案方法を様々な条件で評価し，既存アプローチと精度を比較した結果および，スパース学習により選択された特徴を示す．

6.1 識別精度

5 章に示した D1〜D4 それぞれのデータセットに対して，既存方法および提案方法に関して特徴抽出，識別器の訓練，およびテストデータを用いた精度評価を行った．いずれの識別モデルも精度 (正答率) が最も高くなるように識別関数を最適化している．結果を表 5 に示す．

提案手法は，他の手法と比較してど D2 を除き最も高い精度が得られている．提案方法の識別精度は 95.34%から 98.89%に収まっており，どのデータセットに対しても概ね良い分類性能が得られた．HAC および PE-Miner はデータセットによって識別精度にばらつきがあることがみてとれる．

いずれの手法も D2 に対して，最も識別精度が高い．特に PE-Miner の識別精度は 99.45% と非常に高く，本研究の提案手法の精度 98.90% を上回っている．D2 は Windows にプレインストールされている実行ファイルが正常ファイルの典型例とみなしているが，これらのファイルは様々なチャネルで収集した一般的な正常ファイルと比較して特有の静的情報を有することが示唆される．すなわち，Windows のプレインストールデータをリファレンスとした場合，一般のデータでは思ったような精度が得られないケースが存在する．実際にそのような設定で提案手法を評価している過去の研究も散見されるので要注意である．

一般に実行ファイルに対してパッキングを行うと静的に得られる情報量は減る方向に働くが，良性・悪性ともにパッキングされたファイル同士の比較である D3 においても分類性能が大幅に下がることはなかった．本研究で検出したパッキングされたファイルには resource section をパッキングするケースと UPX 等の全体をパッキングするケースで区別をしていない．このため，正常系とマルウェアにおける パッカー の差異が特徴により顕著にあらわれ、分類性能に影響を与えられた可能性がある．

最も識別精度が低かったのはパッキングが無い検体の識別である D4 である．この原因は現時点でまだ明確ではないが，少なくともパッキングの有無やその差異がマルウェア検出上で有効な指標であることが示唆される．ただし次章でみるように検出に有用な特徴は必ずしもエントロピー等のパッカーの有無を検出するために有用とされる特徴とは限らず，様々な特徴の組み合わせによって決まるため，特定の特徴が精度を支配しているというわけではない．さらに D4 に関しては PE-Miner の精度が HAC よりも良好である．すなわち PE-Miner で利用している特徴はパッキングに関する情報以外の特徴を捉えていると考えられる．

表 5: 各手法のデータセット別の精度比較.

	D1	D2	D3	D4
提案手法	97.97	98.90	98.00	95.34
HAC	95.38	97.25	96.50	90.09
PE-Miner	92.98	99.45	97.00	94.17
SANS	83.55	93.66	83.00	80.47

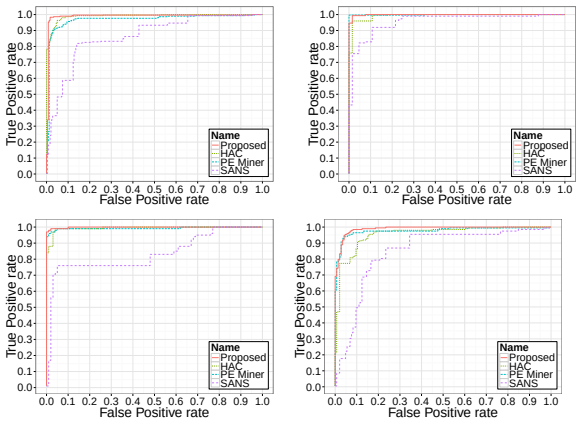

図 1: ROC 曲線: D1 (左上)，D2(右上)，D3(左下)，D4(右下).

各手法に対して，ROC 曲線を描いたものを図 1 に示す．ROC 曲線では曲線が左上に近いほど FP と TP のバランスが良い識別であり，データに偏りがあった D2 を除き，提案手法は他の手法と比較して広い範囲において最良の結果を得ている．以上で示されたように，本研究の提案ではそれぞれランダムに収集した正常系ファイルと悪性ファイルを既存方法と比較して高精度に検出できることを示した．

6.2 特徴の分析

スパース学習後のモデルの変化に関して考察を行う．D1 に対して提案手法と HAC でスパース学習を行い，有効な特徴数を表 6 に示す．提案手法では，特徴数を 25665 から 1167 の 4.54%まで削減することができた．さらに，他手法より良い精度を保つことができた．一方，HAC では特徴数は 19500 から 2384 の 12.23%までしか削減することができなかった．HAC が特徴として用いているインポートされる API 名のみで学習するよりも，様々な特徴を加えて学習を行った方がより少量の特徴まで削減することができると考えられる．

表 6: スパース学習による特徴数の削減効果.

	削減前	削減後
提案手法	25,665	1,167
HAC	19,500	2,384

表 7: スパース学習による各カテゴリの特徴数の削減効果.

カテゴリ	削減前	削減後
Imported symbols	18753	1019
Exported symbols	4037	4
PE Sections	836	15
Delay Imported symbols	747	4
Parsing Warnings	544	12
Debug information	273	3
Resource directory	123	35
Bound imports	111	4
OPTIONAL_HEADER	59	13
Directories	32	4
FILE_HEADER	25	8
LOAD_CONFIG	25	3
DOS_HEADER	24	4
Version Information	16	7
Flags	15	5
DllCharacteristics	8	5
TLS	6	0
Base relocations	2	1
NT_HEADERS	1	0
SANS	28	21

次に提案手法において，分類に影響を与えたカテゴリについて考察を行う．表 7 に特徴がどの程度削減されてたかを示す.

Imported symbols というインポートされた API の特徴数が 18753 から 1019 と大幅に削減されていおり大半の情報は識別に影響を与えてはいない．ただし，他のカテゴリと比較して相対的に，識別に有効な特徴が多い．カテゴリとしては，Resource directory と SANS のカテゴリがスパース学習後でも多く残った.

表 8 に今回の学習におけるモデルの重みを、それぞれ正に働きかけた特徴と負に働きかけた特徴のうち係数が上位の特徴をいくつか紹介する．正に働きかけた特徴は，マルウェア判定に貢献した特徴であり，負に働きかけた特徴は正常系判定に貢献した特徴である．今回の特徴抽出では，出現したかどうかの 0 か 1 かの値だけではなく size や address などの数値情報も特徴として加え [0,1] にスケールしているため，モデルの係数が一概に分類における重要度をそのまま示すものではない.

正負どちらの特徴も Import symbols が多い．また，負に働きかけたものとしては，リソースセクション における icon や dialog box の数が上位にある．また既存の手法で注目されておらず，分類に有効であった特徴としては，リソースから得られる言語情報やオプショナルヘッダの IMAGE_DIRECTORY_ENTRY_SECURITY がある．IMAGE_DIRECTORY_ENTRY_SECURITY は，実行ファイルにデジタル署名がされているかを表す特徴と考えられる．このように既存手法ではあまり注目されていなかった特徴を自動的に抽出することができた.

7 議論

本章では本研究の制限やそこから示唆される今後の課題を議論する.

7.1 データ

本研究は正常系とマルウェアをあわせても 5,000 検体程度と，決して大規模での評価にはなっていない．これらの検体は B2 を除いて極力様々なソースからランダムに収集したものであるが，昨今のマルウェア検体の増加率を鑑みると必ずしも世の中すべての検体を代表したものとは言い切れない．5,000 検体の規模では漏れてしまう検体種別をターゲットにするためにはさらにデータ数を増やすアプローチが有用であろう.

7.2 時間的な変化

現実の問題では，過去のマルウェアから未来のマルウェアを分類できるかどうかが非常に重要である．一般に多くの特徴を利用して学習を行った方が新たな傾向が生じた際に変化を捉えやすいと考えられる．本研究の貢献の一つであるスパース学習による特徴選択により，時期によるマルウェアのトレンドや傾向が見えてくることが考えられる．そのような時間的変化を解析し，変化が識別精度に与える影響の解明と対策方法の開発は今後の課題としたい.

7.3 静的解析と動的解析

本研究では静的情報として PE フォーマットを利用して特徴抽出を行ったが，他にも利用できる特徴を増やすとことにより，識別精度をさらに向上できることが予想される．例えば新たな特徴として動的解析の利用が考えられる．動的解析の導入により，難読化されているケース等，静的情報だけでは判別

表 8: スパース学習後に係数が高かった上位 10 の特徴 (正負)．N はデータが数値データであること，B はバイナリであることを示す．重みが正の値の特徴はマルウェア判定に貢献した特徴であり，負の値の特長は正常系判定に貢献した特徴である．

符号	データ	型	カテゴリ	大分類	小分類	重み
正	D1	N	Base relocations	IMAGE_BASE_RELOCATION	VirtualAddress	3.05
		B	Imported symbols	COMDLG32.DLL	GetSaveFileNameA	2.59
		B	Imported symbols	Cabinet.dll		2.51
		B	Resource directory	LANG_HEBREW		2.33
		B	Imported symbols	KERNEL32.dll	Module32First	2.28
		B	Parsing Warnings	Suspicious flags set for section 3		2.17
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	SizeOfCode	2.12
		B	Parsing Warnings	Suspicious flags set for section 2		2.00
		B	Imported symbols	KERNEL32.dll	SetProcessShutdownParameters	1.99
		B	Resource directory	SUBLANG_CHINESE_SIMPLIFIED		1.91
	D2	N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	SizeOfHeapCommit	7.37
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	SizeOfStackReserve	6.00
		B	Imported symbols	Cabinet.dll		2.69
		B	Resource directory	LANG_POLISH		2.52
		B	Imported symbols	EXPSRV.dll	rtcMidCharVar	2.39
		B	SANS	NumberOfRvaAndSizes != 16		2.33
		N	PE Sections	IMAGE_SECTION_HEADER	.text/Misc_PhysicalAddress	2.30
		B	SANS	LinkerVersion		1.85
		B	Imported symbols	user32.dll	WindowFromPoint	1.83
		B	Imported symbols	ole32.dll	CreateObjrefMoniker	1.70
	D3	B	Imported symbols	user32.dll	SetRectEmpty	3.03
		B	Resource directory	SUBLANG_ENGLISH_SOUTH_AFRICA		2.54
		B	Imported symbols	MsVfW32.dll	DrawDibEnd	2.38
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	DllCharacteristics	2.34
		B	Imported symbols	SHELL32.dll	SHGetDesktopFolder	2.31
		B	Imported symbols	COMDLG32.DLL	GetSaveFileNameA	1.95
		B	Parsing Warnings	Suspicious flags set for section 3		1.83
		B	SANS	SizeOfInitializedData / Sample Size >3		1.82
		B	Imported symbols	SHLWAPI.dll	PathFileExistsA	1.65
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	MinorOperatingSystemVersion	1.63
	D4	N	Base relocations	IMAGE_BASE_RELOCATION	VirtualAddress	5.39
		B	Imported symbols	KERNEL32.dll	Module32First	2.25
		B	Resource directory	LANG_RUSSIAN		1.89
		B	Imported symbols	KERNEL32.dll	SetProcessShutdownParameters	1.88
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	CheckSum	1.81
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	AddressOfEntryPoint	1.76
		B	Imported symbols	KERNEL32.dll	SetPriorityClass	1.66
		B	Resource directory	LANG_HEBREW		1.59
		B	Resource directory	SUBLANG_CHINESE_SIMPLIFIED		1.55
		B	Imported symbols	KERNEL32.dll	Module32Next	1.55
負	D1	N	Resource directory	icon	count	-7.55
		N	Directories	IMAGE_DIRECTORY_ENTRY_SECURITY	VirtualAddress	-6.59
		N	Resource directory	dialog box	count	-5.13
		B	Bound imports	VB40032.DLL	OffsetModuleName	-4.35
		B	Resource directory	SUBLANG_FRENCH_SWISS		-4.07
		N	Debug information	IMAGE_DEBUG_TYPE_CODEVIEW	AddressOfRawData	-4.05
		B	Resource directory	LANG_JAPANESE		-3.46
		B	Imported symbols	shell32.dll	SHGetFileInfoA	-3.09
		B	Imported symbols	MSVCR71.dll	_adjust_fdiv	-2.66
		B	Parsing Warnings	Overlapping offsets relocation data		-2.48
	D2	B	Parsing Warnings	Corrupt header		-3.18
		N	Resource directory	dialog box	count	-3.15
		N	Version Information	VS_FIXEDFILEINFO	ProductVersionLS	-3.00
		B	Bound imports	SHLWAPI.dll	OffsetModuleName	-2.92
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	SizeOfHeaders	-2.82
		B	Imported symbols	ntdll.dll	towupper	-2.75
		N	Debug information	IMAGE_DEBUG_TYPE_CODEVIEW	PointerToRawData	-2.70
		N	Version Information	VS_FIXEDFILEINFO	FileVersionLS	-2.34
		N	Resource directory	icon	count	-2.13
		B	Parsing Warnings	RVA invalid address		-2.04
	D3	N	Resource directory	dialog box	count	-5.25
		B	Imported symbols	shlwapi.dll	StrStrIA	-3.21
		B	Imported symbols	shell32.dll	SHGetFileInfoA	-2.73
		B	Imported symbols	winmm.dll	mciSendCommandA	-2.19
		B	Imported symbols	MSVCRT.dll	exit	-2.12
		B	Imported symbols	msvcrt.dll	strrchr	-2.06
		B	Resource directory	LANG_JAPANESE		-2.03
		B	Imported symbols	shell32.dll	SHGetPathFromIDListA	-1.86
		N	OPTIONAL_HEADER	IMAGE_OPTIONAL_HEADER	CheckSum	-1.68
		B	Imported symbols	ddraw.dll	DirectDrawCreate	-1.59
	D4	N	Resource directory	icon	count	-5.03
		B	Resource directory	LANG_JAPANESE		-4.17
		B	Resource directory	SUBLANG_FRENCH_SWISS		-3.65
		B	Bound imports	VB40032.DLL	OffsetModuleName	-3.07
		N	Debug information	IMAGE_DEBUG_TYPE_CODEVIEW	AddressOfRawData	-2.81
		B	Imported symbols	USER32.dll	MsgWaitForMultipleObjects	-2.62
		N	Directories	IMAGE_DIRECTORY_ENTRY_SECURITY	VirtualAddress	-2.37
		N	Debug information	IMAGE_DEBUG_TYPE_CODEVIEW	SizeOfData	-2.32
		B	Imported symbols	shell32.dll	SHGetMalloc	-2.10
		N	Version Information	VS_VERSIONINFO	Length	-1.71

が難しいケースへの対応が可能となる．本研究は悪性・良性の識別問題をターゲットとしたが，悪性の中でもどのクラスに属しているかという多クラス分類問題への拡張も考えられる．静的・動的情報の両方を組み合わせたマルウェア検知・分類技術の開発と高精度化は今後の課題である．

8 まとめ

実行ファイルの PE ヘッダから静的に得られるありとあらゆる情報を機械学習を適用することにより，どこまで検知率を高めることができるかという **Research Question** に取り組んだ．まず PE ヘッダから様々な情報を自動的に抽出する方法を開発した．得られた高次元の特徴データに対してスパース制約付きロジスティク回帰アルゴリズムを適用することにより，既存の方法と比較して高精度な検知精度が得られること，および正常系のソースやパッカーの有無等，様々なデータ条件の組み合わせに対して提案手法が高精度にマルウェア検出が可能であることを示した．さらにスパース学習により，識別に有効な特徴を元の特徴数から 5% まで削減可能であることを示した．

今回開発した特徴抽出において，数値データの取り扱いは観測した最小値と最大値をもとに $[0, 1]$ 区間へのスケーリングを適用するという非常に簡素なものであったが，適切な量子化を適用することで識別に有用な情報をさらに引き出すことができると期待できる．さらに中長期的な時間で学習モデルがどのように変化し，最新のマルウェア検体に対してどの程度の追従性があるか等の評価は今後の課題としたい．本研究の検証に使ったデータは限られた数量ではあるものの様々なソースから収集したランダムな検体であり，高い識別精度を得ることが出来た．また上述したように我々の方式はさらに精度を向上する見込みがある．これらの事実はマルウェア検知技術はまだまだ発展の余地が残されていることを支持する証左であると我々は確信している．

参考文献

[1] Danny Yadron, “Symantec Develops New Attack on Cyberhacking.” `http://online.wsj.com/news/articles/SB10001424052702303417104579542140235850578`.

[2] Y. Ye, T. Li, K. Huang, Q. Jiang, and Y. Chen, “Hierarchical associative classifier (hac) for malware detection from the large and imbalanced gray list,” *Journal of Intelligent Information Systems*, vol. 35, no. 1, pp. 1–20, 2010.

[3] M. Shafiq, S. Tabish, F. Mirza, and M. Farooq, “Pe-miner: Mining structural information to detect malicious executables in realtime,” in *Recent Advances in Intrusion Detection* (E. Kirda, S. Jha, and D. Balzarotti, eds.), vol. 5758 of *Lecture Notes in Computer Science*, pp. 121–141, Springer Berlin Heidelberg, 2009.

[4] “Microsoft PE and COFF Specification.” `http://msdn.microsoft.com/en-us/windows/hardware/gg463119.aspx`.

[5] “Attributes of Malicious Files.” `https://www.sans.org/reading-room/whitepapers/malicious/attributes-malicious-files-33979`.

[6] “pefile `https://code.google.com/p/pefile/`.”

[7] C. C. Chang and C. J. Lin, “LIBSVM : a library for support vector machines.” Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`, 2001.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, “LIBLINEAR: A library for large linear classification,” *Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.