2010 年度修士論文

Boosting を用いたマルウェアトラヒックの検知手法に関する研究

指導： 小松 尚久 教授

2011 年 2 月 4 日

早稲田大学理工学術院 基幹理工学研究科 情報理工学専攻

5109B085-1　　森　悠樹

目次

第 1 章

序論

1.1　本研究の背景

情報通信技術の発展の中，インターネットが我々の生活に浸透し世の中に欠かせないものとなっている．具体的には WorldWideWeb や電子メールなど人々のコミュニケーションツールとして利用され，金融・経済のビジネス面や電力・水道といった生活インフラの基盤の一部も担うようになっている．

ユーザの利便性が向上する一方で，インターネットを介したセキュリティの問題が発生している．第三者からの悪意のある活動により，個人や企業の持つ情報資産に様々な被害が及ぶようになった．例えば，他者の PC に侵入することで情報資産の破壊や改ざん，盗難を行う不正アクセス，ネットワークやサーバを圧迫することで意図した動作をさせない DoS 攻撃 (Denail of Service) などが挙げられる．

同時にこれらの活動をユーザに認識させずに実行させるプログラムが作成された．これらは悪意のあるソフトウェア (Malicious Software) の略称からマルウェアと呼ばれている [1]．マルウェアに感染してしまうと個人情報の流出や PC の乗っ取りなどの被害を受ける可能性があるため，我々の生活を脅かす存在となっている．

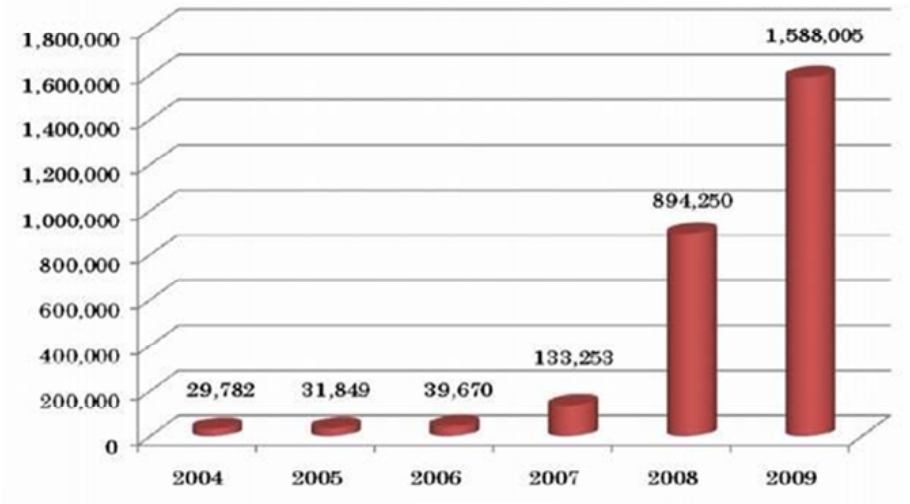

図 1.1　新種マルウェアの発生数

2010 年度上半期の日本国内での被害報告数は約 1 万件にものぼっており [2]，活動が表面化しにくいボットネットによる被害の増加やガンブラーに代表される Web からの感染が増加しているという現状で，早急に対策を講じる必要がある．

また新種のマルウェアは日々増加しているという状況である．年度ごとの新種マルウェアの発生数の動向を図 1.1 に示す [3]．図を見るとわかる通り増加の傾向は一目瞭然である．マルウェアを容易に作り出すことのできるソフトウェアが世の中に存在し，マルウェアを利用したビジネスも横行している．このことより，今後ますます新種のマルウェアが増加すると考えられ，それらへの対応もより一層強めなければならない．

1.2　本研究の目的

本研究の目的は，ルーターやサーバを通過する通信に異常が見られた際の通信の棄却や，マルウェアに感染した PC やサーバのユーザ，管理者にいち早く感染の通知を行うことで，安心なネットワークを実現することである．そこで，マルウェア感染時のトラヒックデータに着目することで，新種マルウェアにも対応した感染検知手法を提案する．

近年のマルウェアによる感染は気付きにくいという問題があるため，感染の拡大を防ぐという意味で感染検知は重要なものだと言える．しかしながら，従来の感染検知では，既知のマルウェアの検知が中心となっている．従来手法として重用されるシグネチャ型の検知手法では，短期間で大量に出現するマルウェアの新種や亜種に対応しきれず十分な検知とは言えない．なぜならば，マルウェアごとにその特徴を示すシグネチャを用意することで検知をするためである．過去のマルウェアの挙動から未知のものを予測で検知する手法も存在するが，誤検知がおきてしまうのが現状である．

そこで本研究では，マルウェア感染時のトラヒックデータの変化に着目することで，新種のマルウェア感染検知を行う．感染時のトラヒックデータの変化に対し，パターン認識の技術を用いることで感染の有無を判定する．特にトラヒックデータの変化について，「感染時の特徴との類似性」「正常時からの逸脱性」という 2 つの視点から感染時のトラヒックデータの特徴を捉え，検知することを想定している．

将来的に本提案手法をルーターやサーバ，ユーザの PC などネットワーク機器群に組み込むことを想定している．感染可能性の注意喚起やマルウェアに関する通信の棄却を行うことで，どんなユーザのセキュリティ意思によらない検知システムを提供し，安心なネットワークを実現することが最終目的である．

そこで本研究ではマルウェアトラヒックを検知可能とする識別器の提案と，その有効性評価の結果を報告する．

1.3　本論文の構成

本論文は以下の構成で書かれている．

■第 1 章「序論」
本研究の背景および目的について述べた．

■第 2 章「マルウェアと対策技術」
マルウェアの基本的な性質や特徴，歴史について述べる．また現在行われている対策技術について述べ，感染検知技術についての関連研究および従来手法について説明する．

■第 3 章「Boosting を用いた感染検知」
本研究の提案手法である，Boosting を用いたマルウェア感染検知手法について述べる．パターン認識の技術および Boosting を用いる背景を説明した後に，Boosting の概要およびその適用方法について述べる．

■第 4 章「マルウェアトラヒック検知実験」
マルウェアトラヒック検知実験について述べる．検知実験の概要を説明した後，今回の実験諸元について述べる．そして最後に CCC Dataset 2010(CCC2010)[4] を用いたマルウェアトラヒック検知実験の結果およびその考察を示す．

■第 5 章「結論」
「Boosting を用いた感染検知手法」について今回行った検討のまとめと，それに関する今後の課題について述べる．

第 2 章

マルウェアと対策技術

本章ではマルウェアの基本的性質および対策技術について述べる．まずマルウェアの特徴やそれによる被害，感染後の動作，歴史等を説明し，マルウェアに対する対策の現状について述べる．

2.1 マルウェアの特徴

2.1.1 基本的性質

前述したようにマルウェアとは Malicious(悪意のある) と Software を組み合わせたものである．ユーザの望まない不正な動作を行うプログラムの総称として用いられている．

マルウェアという言葉が用いられる以前は，このような不正プログラムをウイルスと呼ぶことが多かった．しかしながら PC やインターネットの普及に伴いウイルスの多様化が進み，感染形態や機能，目的などによって数多くの種類が出現した．またウイルスが現れた当初は愉快犯であったり，技術的興味を満たすことが中心であったが，2000 年前後から金銭搾取へと目的が変移していった．

こういった経緯から多様化・高度化・悪質化する不正なプログラムを，統一してマルウェアと呼ぶこととなったのである．

2.1.2 マルウェアの種類 [5]

マルウェアとは悪質なプログラムの総称である．多様化・高度化したマルウェアは非常に多種多様なものとなっている．そこで本項ではいくつかの分類に従って紹介をする．

1. 感染形態に着目した分類
   - ウイルス (Virus)
     ウイルスとは，それ単体では動作せず，自分自身を他のファイルやプログラムに寄生させる感染形態のマルウェアを指す．フロッピーディスクやハードディスクなどのシステム領域を感染対象とするブートセクタ感染型と，実行可能ファイルを主な感染対象とするファイル感染型に大別できる．

- ワーム (Worm)
  ワームとは，単体で動作し自己増殖を行う感染形態のマルウェアを指す，高い感染力を持っており，大規模感染を引き起こす傾向にある．感染手法としては，電子メールやリムーバブルメディアを移動媒体とするもの，Windows のファイル共有やメッセージング機能を利用するもの，OS やアプリケーションの脆弱性に対する攻撃コードを用いるものがある．

- トロイの木馬 (Trojan Horse)
  トロイの木馬とは，有用なプログラムやファイルを装ってユーザ自身によるシステムへの導入・起動を誘い，実際にはユーザの意図しない不正な動作を行うマルウェアを指す．トロイの木馬はユーザの不注意を利用してシステムへの侵入をするため，感染機能を持たないものが多い．

  上記以外にも，フィルタドライバとして実装され，OS のカーネルの深部に潜伏する巧妙な感染形態をもつものもある．マルウェアのファイルやプロセスをアンチウイルスソフトやタスクマネージャに対して隠蔽をするルートキット (Rootkit) や，ユーザのキーボード操作を記録・収集するキーロガー (Keylogger) などは，この感染形態を取ることが多い．

2. 目的に着目した分類
   - スパイウェア (Spyware)
     スパイウェアとは，ユーザの PC 上の個人情報や行動履歴を収集し，特定のサーバなどに送信することを目的としたマルウェアを指す．キーロガーも目的という点ではスパイウェアの一種と考えられる．

   - アドウェア (Adware)
     アドウェアとは，ユーザに企業広告などを提示することを目的としたプログラムである．無害なアドウェアも存在する一方で，ユーザの同意なしに広告を頻繁にポップアップしたり，ユーザの意図しない Web サイトに強制誘導させるものはマルウェアとみなされる．

   - ランサムウェア (Ransomware)
     ランサムウェアとは，ユーザの PC 上のディレクトリやファイルに対して強制的に暗号化やパスワード付き ZIP 圧縮を行うことで，ユーザのデータを「人質」にし，そのデータの復号や解凍の見返りとして，ユーザから身代金 (ransom) を搾取することを目的としたマルウェアである．

- スケアウェア (Scareware)
  スケアウェアとは，ユーザに虚偽の情報を提示し不安 (scare) を煽ることで，無意味なソフトウェアを販売することを目的としたマルウェアである．典型的な例として，偽のマルウェア感染情報をユーザに提示することで Web サイトに誘導し，何も意味の持たないプログラムをアンチウイルスソフトとして販売しようとするものがある．

3. 機能に着目した分類
   - ダウンローダ (Downloader)
     ダウンローダとは，それ自身とは別のマルウェアを特定のサイトからダウンロードし，感染 PC にインストールする機能を持ったマルウェアである．最近のマルウェアの多くは感染後にダウンローダを多段に用いることで解析を困難にしたり，定期的にダウンロードを繰り返したりすることで，新しい機能を持ったマルウェアを容易に拡散させることが可能になっている．

   - ドロッパ (Dropper)
     ドロッパとは，マルウェアを内包した状態で流通し，ユーザの PC 上で実行されると，暗黙のうちにマルウェアをインストール (ドロップ) する機能を持ったマルウェアである．ドロッパの中には Microsoft Word などの文書ファイルになりすまし，実行されると実際の文書を表示すると同時にマルウェアをインストールするという巧妙なものも存在する．

4. その他の分類
   - ボット (Bot)
     ボットとはロボット (robot) の短縮語であり，指令者からの遠隔操作によって，多岐にわたる活動，目的，機能を実現するマルウェアである．ボットに感染した PC はボットネットと呼ばれるネットワークを形成する．ボットネットは小規模なものでは数百，大規模なものでは数十万もの感染 PC 群によって成り立っている．指令者は，指令サーバ (IRC サーバや HTTP サーバ) 経由でボットネットに制御命令を同報し，その結果，多数のボットが命令に従って一斉動作を行う．スパムメールの大量送信や，DDoS 攻撃，大規模な感染活動など様々なインシデントの原因となっている．ボットネットの概要図を図 2.1 に示す．

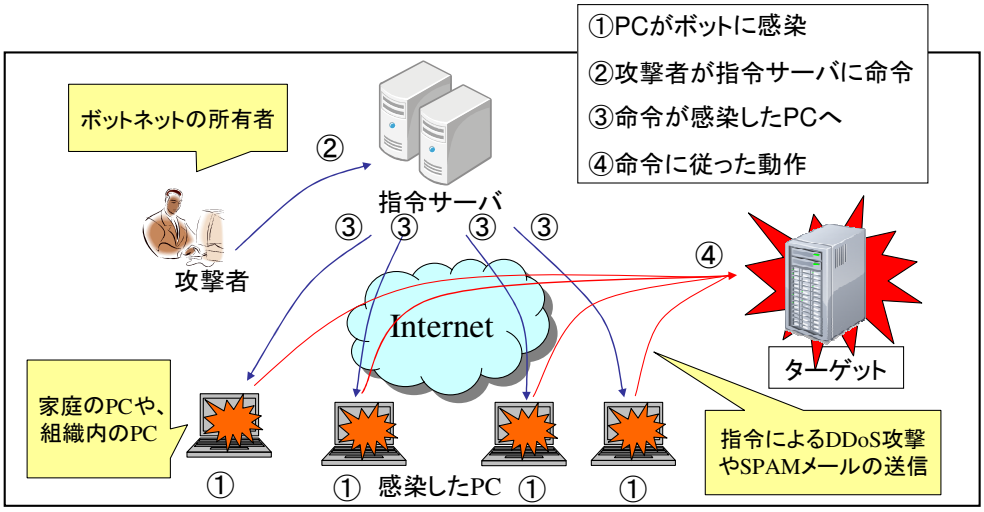

図 2.1　ボットネットの概要

2.2　マルウェアによる被害

マルウェアが登場した当初は，マルウェア作成目的は愉快犯であったり，技術力の誇示が中心であったが，インターネットの普及が進む中で金銭的利益を目的として個人情報の詐取へと切り替わってきた．

そのため感染後の活動も変化してきた．過去のマルウェアではデスクトップ上に画像やメッセージを表示させるなど，被害といえどもさほど大きな問題ではなかった．しかしながら近年では迷惑メールの送信や DoS 攻撃，感染 PC に保存されているクレジットカード番号の収集などが中心となっている．

2000 年 2 月には，Amazon.com や Yahoo!などの大手 Web サイトが大量の計算機からの短時間のアクセスにより，数時間サービスがダウンするという事件が発生した [6].

2003 年 1 月には，インターネット上で急速に拡散するワームが出現し，ワームによる攻撃コードによって韓国のネットワークが数日間利用不能になった [7].

2005 年 10 月にオランダで発見されたボットネットは 10 万ものボットから構成されたという報告があった [8]．後の報告で，このボットネットは 150 万以上のボットから構成されていた事実が判明し，ボットネットによる被害の深刻さを物語っていた．この事件の攻撃者は企業への脅迫や，ハッキング行為を行ったとのことである．ボットネットは PC の管理者，ネットワークの管理者の両者における大きな脅威であるといえる．

2.3　マルウェア感染時の動作[9][10]

マルウェア感染時の動作について説明する．マルウェアに感染する際の活動として，感染，接続，転送，攻撃に大別することが出来る．

1. 感染：脆弱性を突いた攻撃や Web 閲覧などによる感染.

2. 接続：インターネット接続の確認や感染の確認を行う.

3. 転送：検体のダウンロードやアップデートを行う.

4. 攻撃：他の PC への攻撃や迷惑メールの送信を行う.

本節では上記の活動について述べる.

2.3.1　感染

マルウェアへの感染は能動感染型と受動感染型に分けられる．以下に特徴を述べる.

- 能動的感染型
  攻撃者が脆弱性の存在する PC を標的として攻撃を行うものである．対象の PC に対してエクスプロイトコード (Exploit Code) という不正なコードが送り込まれる．ネットワーク環境下において，脆弱性のあるサービスを動作させている PC は感染の標的になる.

- 受動的感染型
  PC の管理者が自らの行動で悪意のプログラムを実行し，感染するパターンである．現在の感染経路としては，Web から感染やメールからの感染が主である．Web 経由では Web ページにブラウザの脆弱性を突くコードが含まれており，閲覧と同時にコードが送り込まれる．またメール経由の感染では，メールにマルウェアもしくはマルウェアをダウンロードするためのソフトウェアを添付し，自発的なクリックを促し感染させるものである.

また上記の攻撃には既知の攻撃，未知の攻撃がある.

既知の攻撃とは，過去のマルウェアに用いられた攻撃など，セキュリティベンダに認知されている攻撃を示す．攻撃の特徴として，流れるパケットの特徴や通信の特徴，ソフトウェアの脆弱性などをシグネチャとして登録しておくことで対処することが出来る．利用者はアンチウイルスソフトのアップデート等の対策を適時行うことで，感染から免れることが出来る.

未知の攻撃とは，ソフトウェアの認知されていない脆弱性を突く攻撃であり，いわゆるゼロデイ攻撃というものである．この攻撃の解析にはある程度の時間を要し，同時にシグネチャの配布

を行う必要がある．シグネチャの配布にかかる時間のため，解析完了後も即座に攻撃を検知することは難しい．

2.3.2　接続

感染したホストはいくつかの接続をすることで，活動までの準備を行っている．以下にその活動を述べる．

- インターネット接続の確認
  感染した PC がインターネット接続されているかを確認する．有名なポータルサイト等に接続を試みることで確認を行う．この確認は新たなマルウェアのダウンロードや C&C サーバへの接続のために行われている．

- C&C サーバに接続して指令を待つ
  ボットに感染した際に見られる活動である．ボットに感染した PC はインターネットの接続が確認された場合，指令が配信される C&C サーバに接続する．

2.3.3　転送

攻撃を受けた PC はマルウェアの転送を開始する．検体のダウンロードや検体のアップデートを行うことで，感染した PC に検体を呼び込む．2.1.2 節で紹介したダウンローダーを実行することで，ダウンロードを開始する．この際の通信方式としてプッシュ型，プル型の通信が存在する．概要を図 2.2 に示す．

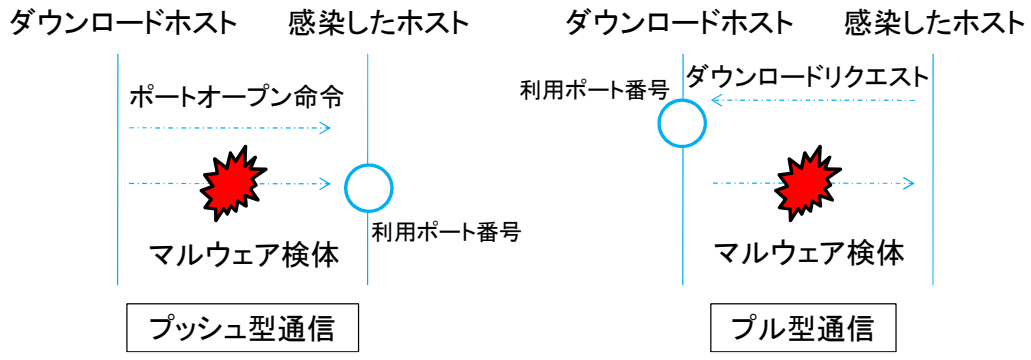

図 2.2　プッシュ型通信とプル型通信

左図のプッシュ型の通信では，感染したホストは攻撃元ホストや C&C サーバなどからの命令によりポートを開き，ダウンロードホストからマルウェア検体が転送されるのを待つ．プッシュ

型ではインターネット上の他のホストから通信が開始されるため，NAPT などの外部からの通信が制限された環境においては通信が開始されない．

右図のプル型の通信では，感染したホストからダウンロードホストにマルウェア検体を要求する形でダウンロードが行われる．感染したホストから通信が開始されるため，外部からの通信が制限された環境においてもダウンロードが実行される．

2.3.4　攻撃

感染後の攻撃としては，感染を拡大させるものや，サーバ機能の停止を目的とするものがある．以下それらについて述べる．

- 感染の拡大
  他の PC に対してポートスキャンを行い，脆弱性の調査を行う．同一のネットワーク，インターネット上のホストを対象にスキャンを行う．また脆弱性を発見した際には，感染活動を行いコードやマルウェアを送り込む．

- DoS 攻撃
  Web サーバやアプリケーションサーバ，ルーターなどに対して過度のアクセスを行うことで，対象への負荷をかける攻撃である．サーバは自らのリソースをもとにサービスを提供するため，過度の要求が発生するとサービスの停止や遅延を引き起こして，経済的な打撃を与える．

- スパムメールの送信
  スパムメールの送信とは，感染した PC を介して無差別にインターネットメールを送信するものである．単純に営利目的のものであったり，悪意のある Web ページへの誘導を行うためのメールが主である．営利目的のものでは，自らの利益に結びつくような Web ページのアドレスを記載したメールを送る．悪意のある Web ページへの誘導をするものは，Web ブラウザや Web コンテンツが利用するアプリケーションの脆弱性をつくようなコードが含まれる Web ページのアドレスを含むメールを送信する．

ボットネットが構成されている際，分散された PC からの攻撃が行われる．そのため DoS 攻撃やスパムメールによる脅威はより一層大きなものとなる．2.2 節で挙げた，Amazon.com や Yahoo! の事件にもボットネットが利用されたという話もあり，脅威のほどが伺える．

2.4　マルウェアの歴史 [5]

本項ではマルウェアの歴史について紹介する．歴史の中でも特にマルウェアが悪用された時代にフォーカスを当てる．

2.4.1　1970 年代〜1980 年代中盤

ネットワーク上で自己増殖するマルウェアがはじめて発見されたのは 1971 年である．BBN 社の BobThomas がインターネットの原型である ARPANET に放った Creeper が世界初のワームといわれている．

また 1980 年にはゼロックス・パロアルト研究所 (PARC) の John F. Shoch と Jon A. Hupp が，ワームを用いた分散コンピューティングの実験を行った．このワームはネットワークを探索し，アイドル中のマシンを見つけると，ネットワークブートさせ，同時に自分自身を送り込み計算力を借りるという機能であった．PARC 内のイーサネット上にテストのために置かれていたワームが暴走し PARC の相当数のマシンがクラッシュする事態が起こった．

1982 年には当時高校生であった Richard Skrenta が，同級生を驚かせる目的で世界初のウイルス Elk Cloner を作成した．1984 年には，Fred Cohen が論文を発表し，その中で初めて (コンピュータ) ウイルスという用語を定義し，将来的な脅威の予見とともに，対策の困難さを言及した．

1970 年代から 1980 年代中盤にかけては，ウイルスやワームが計算機やネットワーク上で実在し得ることや，それらの脅威が認識され始めた「発見の時代」であったといえる．

2.4.2　1980 年代中盤〜1990 年代中盤

1980 年代中盤から 1990 年代中盤には様々な機能が開発され，実世界で試される「発見の時代」となる．

1986 年，パキスタンの Farooq Alvi 兄弟によって，IBM PC に感染する初のウイルス Brain が作成された．このウイルスは彼らが経営する Brain 社のソフトウェアを違法コピーした PC に感染し，駆除のために Brain 社にコンタクトを求めるメッセージを表示するものであった．同年には初のトロイの木馬 PC-Write も登場している．

1988 年には，sendmail，fingerd，rsh/rexec などの複数の脆弱性とパスワードクラックを悪用した Morris ワームがインターネット上で拡散した．

1990 年以降，Virus Exchange BBS と呼ばれるウイルス情報交換のための掲示板が出現し，MtE(Mutation Engine)，VCL(Virus Creation Labratory)，PS-MPC(Phalcon/Skism Mass Produced Code Generator) に代表されるマルウェア作成ツールが登場した．これによりマルウェア作成の技術的なハードルが下がり，実験的なマルウェアが多く登場した．また同時にアンチウイルス無効化機能など，マルウェアの基礎技術が開発された．

2.4.3　1990 年代中盤〜現在

1990 年代中盤以降，マルウェアは「悪用の時代」に突入した．

1990 年後半からウイルスやワームがメールなどインターネット上の通信手段を用いて感染を広めるようになった．1999 年の Bubbleboy や 2000 年の LoveLetter ウイルスは，電子メールクライアントの脆弱性を利用し電子メールを閲覧するだけで感染するため，高速に感染が拡大した．

1995 年には Concept，1999 年には Melissa という MicrosoftWord のマクロ機能を利用したマルウェアも台頭した．これらは感染した Word ファイルを開くことで感染が行われる．Melissa では感染 PC 上の Outlook のアドレス帳に登録された最大 50 個のメールアドレスに向けて，感染した Word ファイルを添付したメールを送信するため，感染が拡大した．

2000 年代前半には，Windows や Windows サーバの脆弱性を狙う攻撃コードを用いたワームの大規模感染が次々と発生した．2001 年 7 月の CodeRed や 2001 年 9 月の Nimda，2003 年 1 月の SQLSlammer，2003 年 8 月の MSBlaster などの代表的なウイルスは攻撃対象のホストが通信をしているサービスに対して攻撃コードを送信し，対象ホストを攻略していた．

P2P ファイル共有ソフト Winny を介して拡散するマルウェア Antinny は，数々の情報漏えいの引き金となった．

2004 年頃から，マルウェアに感染している PC が同期して活動するよう数が世界中で観測され始めた．2.1.2 節でも説明した「ボットネット」である．2.2 項でも示したように，ボットネットによる不正行為 (金銭目的) は日々発生するようになった．スパムメールの 9 割近くがボットネットから送信されているという調査もあり，ユーザをフィッシングサイトやマルウェア感染サイトへと導き続けている．

ボットネット登場後，マルウェアはさらに高度化・巧妙化した．傾向として検出を避けるよう静かに潜伏を続けるものが多くなり，フィッシングや SQL インジェクションなどの金銭的価値のある個人情報への攻撃も目立つようになった．

そんな中，2008 年 11 月に Conficker と呼ばれるマルウェアが出現する．この Conficker は過去のワームが備えていた様々な機能の集大成のようなマルウェアであった．USB メモリを介した感染や，感染 PC 間での P2P ネットワークの自律的確立など，今までにはなかった機能も備えていた．2009 年 2 月の段階で約 1200 万台もの感染が報告され，大規模感染の余韻は現在も残っている．

さらに，2009 年 5 月頃から Gumblar 攻撃という新種の攻撃手法も流行しだした．改ざんされた Web サイトを閲覧することによるマルウェア感染である．大小様々な Web サイトが改ざんされ瞬く間に感染が広がった．マルウェア感染後に感染 PC 上にある FTP アカウント情報を取得し，FTP アカウントを用いた Web サイトを改ざんするというサイクルを繰り返すもので，今現在もボットネットに並ぶ脅威となっている．

2010 年 3 月には，HTC 製スマートフォン「HTC Magic」にマルウェアが発見されたという事

例がある．最近では Android 端末や iPhone といったスマートフォンの普及が進んでいるため，将来的にスマートフォンを標的としたマルウェアが蔓延することも考えられる．

また同月には Mariposa と呼ばれるボットネットを管理していた首謀者が逮捕された．Mariposa のボットネットには 1200 万もの IP アドレスが接続されていた．被害は 190 カ国以上の企業や政府機関，家庭などの PC におよび，銀行講座情報やクレジットカード番号等の情報が盗み出されていた．

2010 年 7 月には Stuxnet と呼ばれるマルウェアが登場した．制御システムや電力会社を狙った初のマルウェアで，工場の操業計画に支障をきたすなど，社会インフラという新たな面への脅威となった．

以上のように，マルウェアの歴史は非常に長い．マルウェアの歴史は裏を返すとマルウェアとの戦いとも言え，これからも新たな脅威に対する注意を払い対策を講じていく必要がある．

2.5　マルウェア対策技術

本項では，マルウェア対策技術の現状について述べる．

2.5.1　ホスト上での対策

ホスト上におけるマルウェアへの対策手法は，各ホスト上でソフトウェアの実行やネットワークの通信を監視し，ボットへの感染を予防あるいは検知するというものである．メモリやファイルシステムに対する不正なアクセスや，マルウェアによく含まれるシステムコールの特徴などに着目し，感染あるいは感染後の詳細なマルウェアの特徴を用いた検知が可能である．

現在多くのセキュリティベンダから対策ソフトとしてアンチウイルスソフトが提供されている．基本的な機能としてはシグネチャ型の検知とファイアウォール機能がある．シグネチャ型検知は個々のマルウェアの種類毎にバイナリパターンを定義したシグネチャを作成し検知する手法である．ファイアウォール機能では，決められたプロセスからの通信や指定されたポート番号のみ外部からの通信を許可するといった機能になっている．

またプログラムの活動パターンからマルウェアの類似度を推測するヒューリスティック検知，ビヘイビア型検知もある．ファイルアクセス方法や OS に関するデータへのアクセス方法・内容をもとに検知している．

2.5.2　通信制限による対策

通信の一部を制限することでネットワーク上の脅威から守る代表的な手法としてファイアウォールがある．外部ネットワークから内部ネットワークへのアクセスの拒否，外部ネットワー

クから公開サーバへのアクセスの中から不要とされるアクセスを拒否する等の機能がある．ファイアウォールにはネットワーク全体の制御をするネットワーク型ファイアウォールと，各ホスト上で動作するホスト型ファイアウォールの 2 つに分けられる．

ファイアウォールは外部から内部，自ホストへのアクセスを制限するのが目的であり，内部から外部へのアクセスにおいて悪意ある通信のみを遮断するのは難しいという問題がある．

2.5.3　ネットワーク監視による対策

通信を監視することで，異常を発見する手法として，ネットワーク型侵入検知システム (Network-based Intrusion Dtection System, IDS) がある．統計的情報から事象を発見するアノマリ型と指定した通信パターンを発見するシグネチャ型がある．

アノマリ型では通常システム利用ではありえないパケットを検出することで，悪意のパケットを検知する手法である．プロトコルの異常やアプリケーションに対する通信などに着目している．

シグネチャ型ではあらかじめ攻撃コードやマルウェアに関する活動の通信パターンが定義されたシグネチャを読み込み，ネットワークの通信に含まれるパケットを解析する．シグネチャと通信パターンが一致した場合，悪意ある活動を検知したとみなす．

2.5.4　サービスを対象とする対策

サービスを対象とする対策は，Web やメールのサービス上で対策を施す手法である．前述した受動的な感染への対策となっており，サービスの利用者やネットワークに依存することないため効果は大きい．Web サイトへのアクセスを解析し，マルウェア感染の危険があるサイトへのアクセスを未然に防ぐ方法がある．セキュリティの知識がない一般の利用者に対して有効である．

メールサービスにおける感染防止対策としては，メールサーバにおいてメッセージを検査する手法がある．ClamAV[11] や IronPort[12] などが，メールサービスの対策手法として用いられている．

Web 検索サービスの 1 つである google では，検索結果にマルウェア感染の可能性を持つページを表示させない技術を導入している．この技術は GhostBrowser[13] と呼ばれている．利用者の意図しないファイルがダウンロードされる Web コンテンツを悪意の持つ Web コンテンツと定義する．マルウェア感染の可能性があるページの検索結果には警告を発する．

2.6　マルウェア対策研究の現状

現在行われているマルウェアの対策研究としては以下のものが挙げられる [14][15].

- 感染検知
  PC がマルウェアに感染しているかどうかを検知する手法の研究.

- 検体解析
  感染後の挙動の感染やコードの解析を行う研究である．これは「動的解析」と「静的解析」に分けられる．
  動的解析とはマルウェアを実際に動作させて挙動を分析する解析方法であり，静的解析はマルウェアのプログラムコードを分析する解析方法である．

- 広域観測
  おとりシステムであるハニーポットを用いたボットネット等の活動状況の観測を行う研究.

検体解析では高度な技術が必要とされ，また広域観測ではハニーポットの設置やセキュリティの問題から研究の対象として感染検知に着目した．また，本研究室で従来から用いられてきた「パターン認識」の技術を感染検知に適用できると考えたことも理由としてある．

2.7　マルウェア感染検知

本研究で対象とするマルウェア感染検知に関連する研究を紹介をする．

2.7.1　感染検知従来手法

文献 [14][16] を参考に感染検知の従来手法について簡単に整理する．従来手法としては以下のものがある．一部 2.5 節でも触れているものもある．

1. シグネチャベースによる検知
マルウェアごとにシグネチャデータを用意することで，パターンマッチングを行い検知する手法である．新たなマルウェアが現れるごとに分析をする必要があり，未知のものは検知できないといった課題がある．

2. ルールベースによる検知
脆弱性をつく攻撃に関してルールを設定することで，そういったパケットが到着した際に感染を検知する手法である．ゼロデイ攻撃のような，新たな脆弱性をついた攻撃には対応しきれないという課題がある．

3. トラヒックデータを用いた感染検知
ボットネットにおける C&C サーバと感染した PC 間の通信や感染後の DNS クエリの異常に着目することで検知を行う．シグネチャベースによる検知に比べて，過去のマルウェアとの比較

が難しいという課題がある．

4. イベント観測
　マルウェアの感染後の動作として現れる，DDoS 攻撃やスパムメール発信等の発症活動に着目し感染したホストの検知を行う．動作が起きてからの検知になり，大きな挙動での検知になってしまうという課題がある．

5. ヒューリスティック検知
　実行ファイルの挙動などを解析し，ライブラリファイルの書き換えなど，一般的にはあまり見られない特異な挙動を探し未知の検知を行う．しかしながら精度が必ずしも高くなく，誤検知もあるという課題がある．

2.7.2　トラヒックデータに着目する理由

　本研究では特にトラヒックデータに着目した感染検知を行う．なぜならば正常時と感染時のトラヒックデータの特徴に違いが見られるからである [16]．その違いに対して本研究室で用いられてきたパターン認識の技術を適用できる．加えて過去のデータを学習することで新種や亜種のマルウェアに対応できると考えられる．また，トラヒックデータには時系列の変化を考えることができ，識別器に対する連続的な入力が仮定出来る．これにより精度向上が伺えるというメリットもある．
　例えばバイオメトリクスでは，発話時の唇動作個人認証において複数のアルゴリズムが提案されているが，時系列を使用しないアルゴリズムと使用するものを比較した際，後者のアルゴリズムのほうが高い精度での認証が可能であることが示されている [17].

2.7.3　トラヒックデータを用いた感染検知・異常検知

2.7.3.1　感染検知

　トラヒックデータを用いたマルウェアの感染検知の研究事例を紹介する．マルウェアの感染時の特徴に着目した検知手法となっている．
　S.Kondo らは [16]，ボットに感染した PC と C& C サーバとのセッションと Web 閲覧等とのパケットサイズの違いに着目する手法がある．トラヒックデータよりパケットサイズ，送受信間隔によるヒストグラムを特徴量とし，SupportVectorMachine(SVM) を用いて C&C サーバのセッションを検知し，SVM が有効であることを示している．
　安部ら [18] は，[16] と同様に C&C サーバとの通信の特徴として，パケットサイズと送受信間隔に着目し，C&C セッションにおける送信パケットが通常の IRC 通信と比較し応答時間が短く，

Web 閲覧と比べて送受信パケットサイズの違いが顕著であることを確認している．加えてボットに使用される感染開始時からの通信プロトコルの遷移の仕方を特徴として提案している．

東角ら [19] は，トラヒックデータからハニーポットの特徴的な挙動として攻撃を受けながらも感染しなかったケース，マルウェア本体のダウンロードにいたらなかったケース，攻撃から活動に至るまでの一通りの動作が行われたケース，複数の感染が確認されたケースが存在することを示している．また，DNS クエリに着目し，Windows2000 では IP アドレスを指定した正引きが行われ，WindowsXP ではリゾルバの逸脱を確認し，感染初期の活動の検知にこれらの特徴が利用できる可能性を示している．

2.7.3.2　異常検知

従来よりネットワークの異常検知という分野がある．通常とは異なる振る舞いを検出することで異常を検知している．マルウェアの感染時を「正常ではない状態」と捉えることも出来るため，異常検知手法も参考になると考えられる．

及川ら [20] は，主成分分析を用いた異常検知を提案した．プロトコルごとのパケット間の相関関係に基づき，主成分軸から距離が離れているものを異常として検出する．正常状態がある軸または平面に沿って分布し，異常状態はプロトコルの分布が崩れ軸から離れる，という仮定に基づいている．

宮本ら [21] は，パケットヘッダの情報 (TCP フラグ別パケット数，TCP プロトコル別パケット数，パケットサイズ別パケット数) を集約して SVM によりクラスタリングを行い，異常検知を行う手法を提案している．

第 3 章

Boosting を用いた感染検知

本章では，本研究の提案手法である Boosintg を用いた感染検知手法について述べる．識別器を用いた感染検知手法および Boosting の適用について述べた後に，AdaBoost の概要および構成方法に関して説明する．

3.1 識別器を用いた感染検知

パターン認識とは入力されたパターン (指紋等の画像や) をいくつかのクラスごとに分類することができるとき，あるパターンを複数のクラスに対応させることである [22].

まず入力パターンに対し正規化やノイズ除去といった前処理後，特徴量を数値化して抽出する．抽出できた特徴をまとめて特徴ベクトルとし識別に用いる．いま d 個の特徴を用いるとき，特徴ベクトルは式 (3.1) で表される．

$$\vec{\mathbf{x}} = (x_1, x_2, \ldots, x_d)^t \tag{3.1}$$

この特徴ベクトルによって構成される空間を特徴空間と呼ぶ．ある 1 つのパターンは特徴空間上の 1 点を示すこととなり，同一クラスに属するパターンは互いに類似しているためまとまったかたまりとして観測される．つまり特徴空間上に存在する複数のパターンを，クラスごとに分類できる識別境界の作成が識別器の設計ということになる．

未知の入力パターンを適切なクラスに分類することが目的であり，そのためにクラスごとのサンプルを用いて学習を行い，識別境界面を作成し，分類することとなる．手順を図 3.1 に示す．

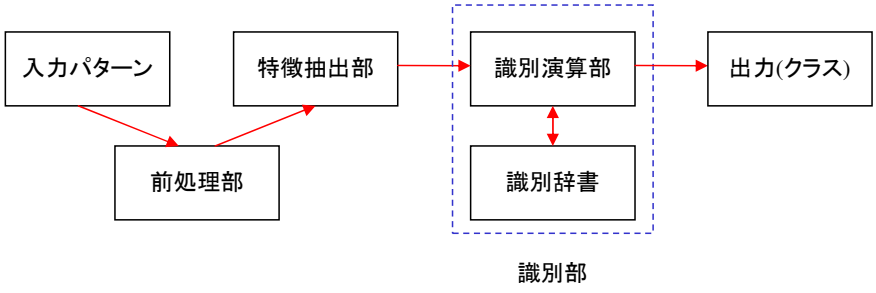

図 3.1　認識系の流れ

本研究ではトラヒックデータを用いた感染検知という観点から，感染の有無を正常なトラヒックデータか否かという 2 つのクラスに対応させて考える．前処理部では，対象となる通信から特定のノードとの通信を切り出して特徴量を抽出しやすい形にする．次に，特徴抽出部でパケットサイズやパケット到着間隔といった特徴量を抽出し，あらかじめ学習をして作成した識別辞書との比較を行うことで，入力されたトラヒックデータが正常か異常かを決定する．

識別器の設計を行うにあたって重要である点は，各クラスの分布が分類できるような特徴量を用いることである．また識別アルゴリズムに関しても，特徴量の分布に適したものを利用する必要があるため，適切な識別器の選択，妥当性の評価をしなければならない．

マルウェアの感染検知に関する従来研究では上記の評価が必ずしも十分なされていないという現状であり，パターン認識という観点からの基礎的検討も必要である．

検討するマルウェア感染検知の流れを図 3.2 に示す．

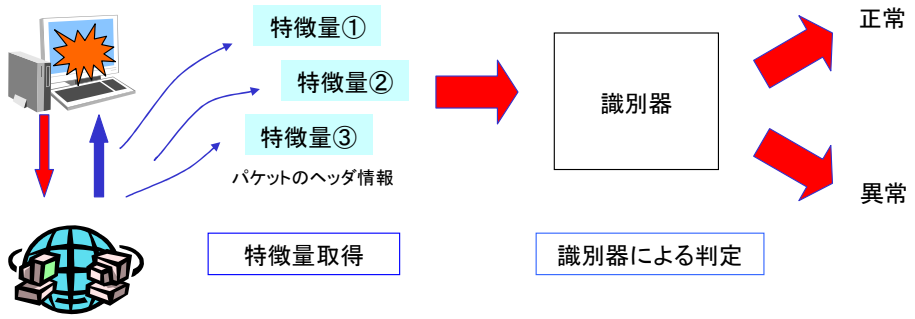

図 3.2　システムの概要

感染の有無を判別したい PC におけるトラヒックデータを取り出し，前処理を行った後にヘッダ情報から複数の特徴量を取得する．これらの特徴量に関して識別器を用いて対象のトラヒックデータが正常か異常かの判定を行う．

識別器を作成する際には，あらかじめ実際の感染時のデータ，正常時のデータを用いて学習を行う．

3.1.1　用いる特徴量について

近年，ネットワーク上でのプライバシーの問題が重要視されている．ネットワーク上のパケットをキャプチャする際に，トラヒックデータのペイロードを見てしまうと，プライバシーを侵害してしまう可能性がある．このような問題からペイロードは暗号化されていることが多い．そのため，トラヒックデータのペイロードを特徴量として扱うことは望ましくない．そこで本研究では，トラヒックデータのヘッダ情報のみから特徴量を抽出し，それを用いたパターン認識を行う．

3.2　正常時・感染時からのアプローチ

近年のマルウェアでは感染時の通信は正常な状態と区別しにくいという傾向にある [10]．例えば，マルウェア検体のダウンロード通信が FTP によるものであったり，ボットに感染した際の指令の通信に IRC を用いているといったもので，一般的なユーザのインターネット利用時の通信と区別がしづらくなっている．つまり感染時の特徴のみに着目するだけで感染を判定することは容易ではないということである．

そこで本研究では「感染時の特徴との類似性」「正常時からの逸脱性」という 2 つのアプローチから感染検知を考える．感染時の特徴から感染時通信と判定し，正常時の特徴から正常通信ではないと判定する．特に正常時に関しては，前述した感染時との区別のしにくさから，少しでも異常時と疑える場合に検知の対象にする．つまり 2 つの特徴から段階的に感染部分を残していくイメージである (図 3.3)．パターン認識の考えに乗っ取ると，弱い識別を複数回繰り返すことで感染の判定の精度を高めていくということである．

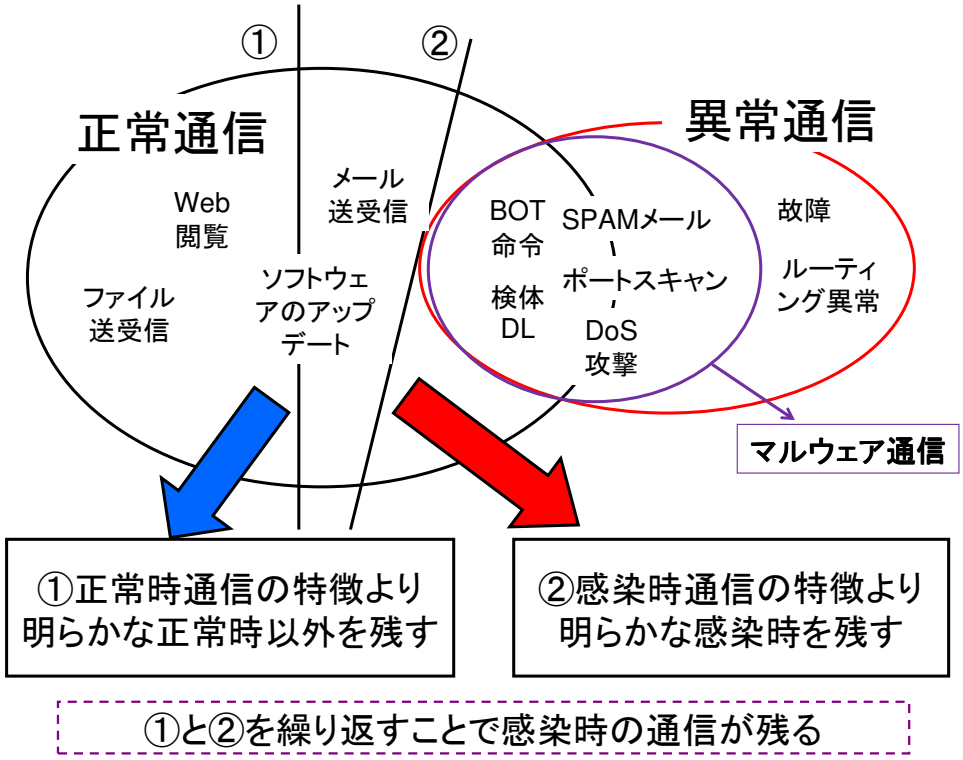

図 3.3　正常時・感染時からのアプローチ

3.3　Boosting

3.3.1　Boosting の概要

前項で示した通り本研究では段階的な識別による検知の精度向上を想定している．そこで，識別の難しいデータを重点的に繰り返し学習していくことで単独の識別器に比べ高い識別性能が期待できるアンサンブル学習に着目する．特にその中でも Boosting[23] を識別器に適用することを考えている．Boosting は逐次的に学習データの重みを変化させながら複数の弱識別器で学習をし，それらを統合した識別器で最終的な識別をする手法である (図 3.4).

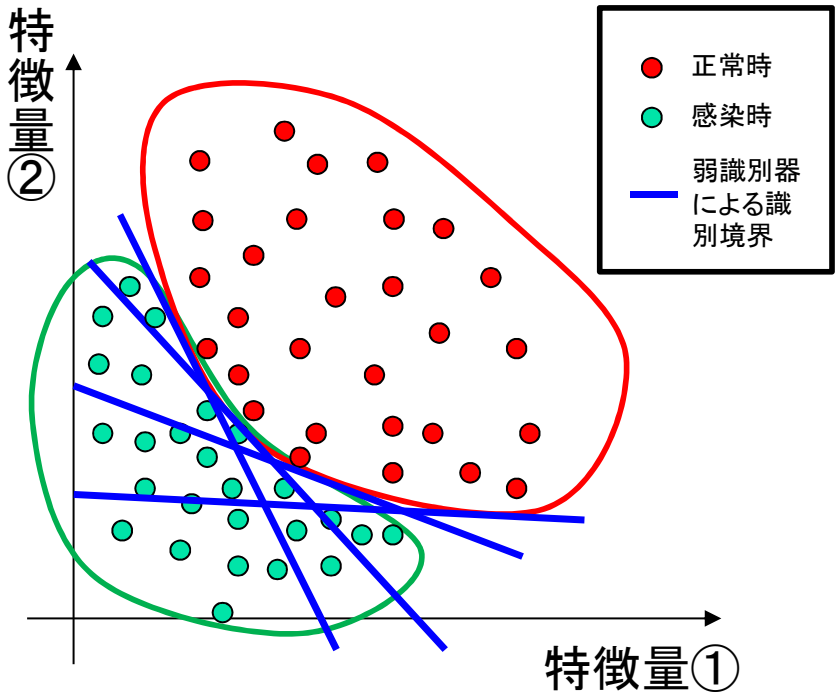

図 3.4　Boosting の概念

弱識別器の学習はそれまでの学習結果に依存して行われる．学習データには最初は一様に重みが付与される．弱識別器の学習のステップごとに分類を誤ったデータの重みを増やし，正しく分類できたデータの重みは減らし，重みの大きなデータを優先的に学習することで，誤分類の多いデータを正しく分類できるように学習を行う．また，データに重みを付加することにより，データの重要度を制御できるので学習アルゴリズムが特定のデータを正しく分類するように仕向けることができる．概要図を図 3.5 に示す.

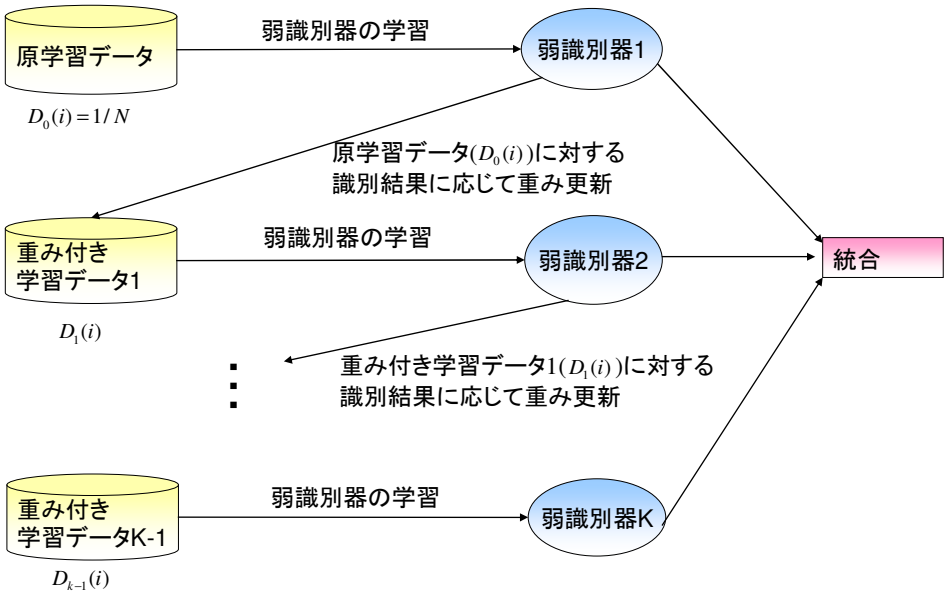

図 3.5　Boosting の概要図

3.3.2　Boosting の特徴

Boosting の特徴として以下の点が挙げられる．

- 弱識別器の精度は 50%以上であれば良い
- 弱識別器ごとにデータの重み付けによる精度向上
- データの重みの更新方法や弱識別器の選び方は多様

また Boosting では誤分類したデータの重みを重くすることで k-1 回目までの学習で誤分類されたデータに関して k 回目の学習で重点的に学習を行う．よって，k 個目の弱識別器では k-1 個目までの弱識別器が苦手とするデータに関してより分類能力が高くなるように学習が行われるため，お互いが相補的な性能を持つようになる．

3.4　トラヒックデータへの Boosting 適用メリット

Boosting をマルウェアトラヒック検知に適用する際のメリットを示す．

1. 段階的な識別

   3.2 説で示した通り，正常と感染を段階的に識別していくことが可能である．そのため感染の可能性が高い通信を識別できる．またデータの重みおよびそれらの更新があるため，サンプルの分布が複雑で識別が難しい場合でもより精度の高い識別が行える．

2. 用いる特徴量を変化させながらの識別
   正常時と感染時のそれぞれの特徴を表現する特徴量は多様に存在すると考えられる．加えてトラヒックデータは，様々な挙動の通信の組み合わせによって構成されているため，複数の特徴量の組み合わせを用いた識別が有効だと考えられる．

3. 処理の軽い識別器
   弱識別器の精度はある程度の性能があれば良いため，単純な識別器を用いることが出来る．単純である分処理が軽くなるので実運用に向いており，リアルタイム性が必要とされるマルウェアトラヒックの検知に適する．

4. 識別器の構成の多様さ
   識別をするまでの工程は複数あり，それらの条件は識別の方針によって調節することが出来る．例えば間違えたサンプルをより重点的に識別したいとき，間違えたサンプルの重みの更新を通常より大きくするといった方法がある．

以上の点から，Boosting を適用することが，柔軟な構成の識別器を作成可能にし，同時に多くのメリットを与えてくれることがわかる．よって感染検知の手法として Boosting を用いた識別器を提案する．

Boosting には様々な手法が提案されているが，本研究では AdaBoost[24]，[25] を用いる．

3.5　AdaBoost の概要

AdaBoost では毎回の繰り返しごとにデータに付加された重みに従い，学習データのリサンプリングを行う．リサンプリングした学習データに対して識別器の学習を行い，その識別器の精度から求めた重みを識別器に与える．その識別器の結果を用いて学習データの重みを更新し，学習データのリサンプリングから繰り返す．最終的な識別の結果は，識別器による判定結果および識別器の重みを用いた多数決を行う．

学習データとして N 個のサンプル (x_1, y_1),・・・,(x_N, y_N) が与えられたとする．x_i はサンプル，$y_i \in \{+1, -1\}$ は x_i が属するクラスのラベルである．

(1) 各サンプルの重みを $D_1(i) = 1/N$ で初期化する．

(2)$t = 1, \cdots, T$ で以下 (A)〜(D) を実行する．
(A) サンプル分布 D_t において，弱識別器 $h_t(x)$ を学習する．そこで学習サンプルに対する誤り率

$$\epsilon_t = \sum_{i:y_i \neq h_t(x_i)} D_t(i) \tag{3.2}$$

が最小となる $h_t(x)$ を選ぶ．$h_t(x)$ は+1 または-1 の値をとる関数である．
　ここで誤り率はサンプルの個数ではなく重みに基づいて算出する．

(B) 誤り率から信頼度 α_t を計算する．

$$\alpha_t = \frac{1}{2}\log\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{3.3}$$

この信頼度 α_t は誤り率が小さいほど大きくなり，弱識別器の重要度を示す尺度となる．

(C) サンプルの重みを更新する．

$$D_{t+1}(i) = D_t(i)\exp(-\alpha_t y_i h_t(x_i)) \tag{3.4}$$

弱識別器で正しく識別できたサンプルでは $y_i h_t(x_i) = 1$ であるから，

$$D_{t+1}(i) = D_t(i)\exp(-\alpha_t) \tag{3.5}$$

となる．つまり，重みは小さくなる．
　一方間違えたサンプルでは $y_i h_t(x_i) = -1$ であるから．

$$D_{t+1}(i) = D_t(i)\exp(\alpha_t) \tag{3.6}$$

となる．つまり，重みは大きくなる．
　$\epsilon_t = 0.5$ の場合は $\alpha_t = 0$ となり，重みが更新されないので学習を継続できない．

(D) サンプルの重みの和が 1 になるように正規化する．

$$D_{t+1}(i) = \frac{D_{t+1}(i)}{Z_{t+1}} \tag{3.7}$$

ここで，

$$Z_{t+1} = \sum_{i=1}^{N} D_{t+1}(i) \tag{3.8}$$

$$= \sum_{i=1}^{N} D_t(i)\exp(-\alpha_t y_i h_t(x_i)) \tag{3.9}$$

(3) 最終的な識別器は，すべての弱識別器を信頼度で重み付けて多数決を取った

$$H(x) = sign\left(\sum_{t=1}^{T}\alpha_t h_t(x)\right) \tag{3.10}$$

となる．

AdaBoost のフロー図を図 3.6 示す.

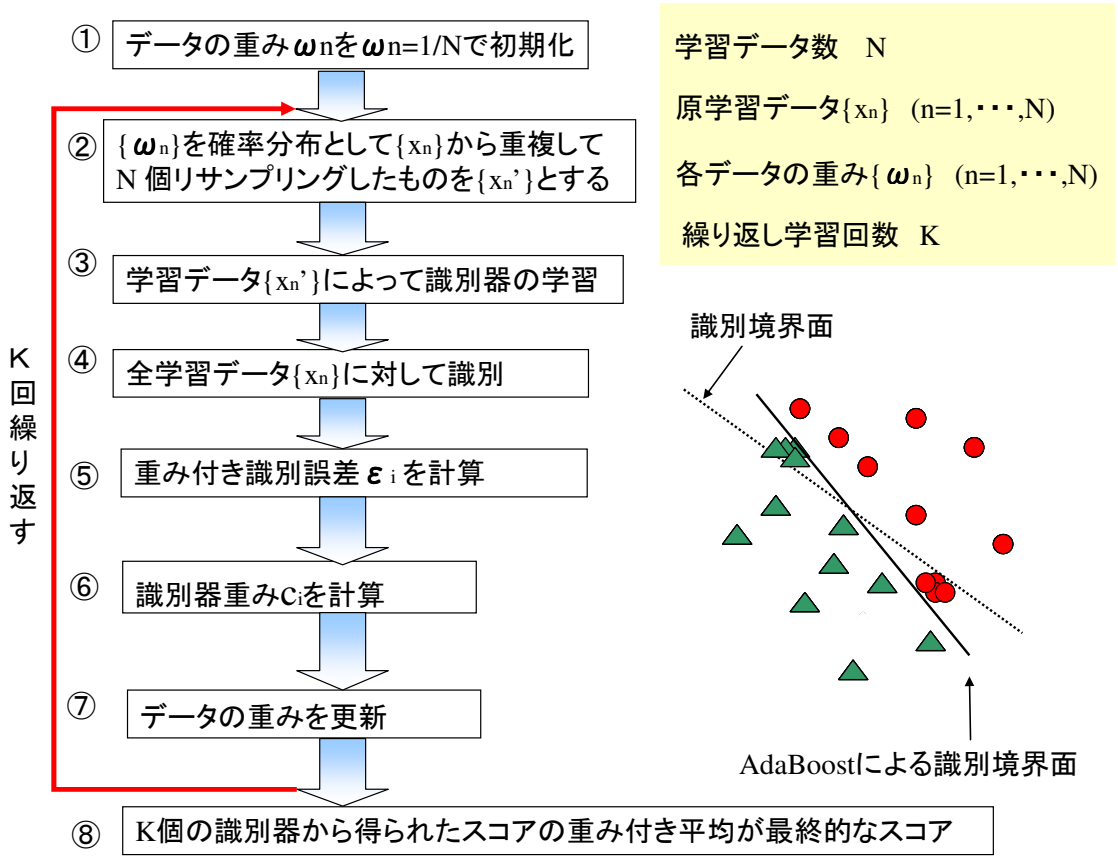

図 3.6　AdaBoost のフロー図

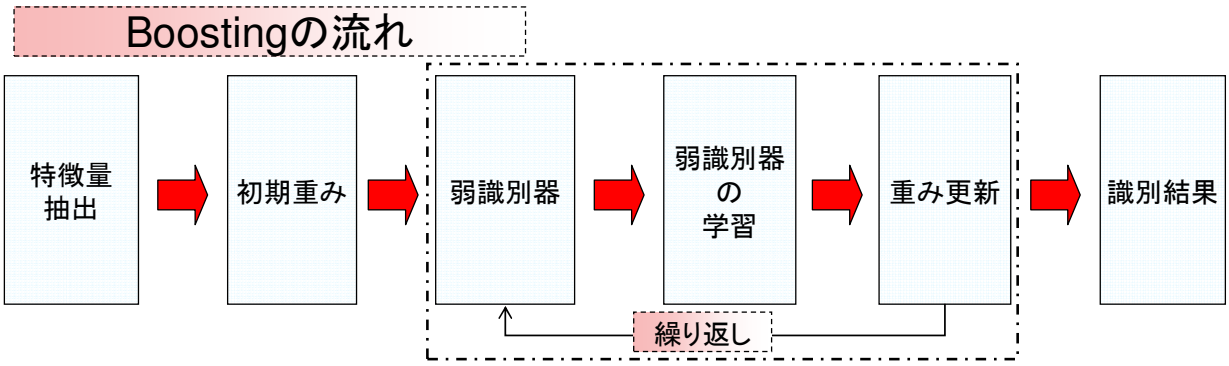

図 3.7　Boosting の工程

3.6　AdaBoost の構成方法

図 3.1 でも示したように，識別を行うまでにはいくつかの工程がある．AdaBoost を識別器に適用する上で，これらの各工程ごとに複数の条件が考えられる．このため識別器の構成の仕方は非常に多様であり，構成方法次第で識別結果に違いが出てくる．そこで本節では AdaBoost の構成方法について述べる．

各工程の流れを図 3.7 に示す．ここで各工程についてどういった条件が考えられるか説明する．

1. 特徴量抽出
   用いるべき特徴量は感染時と正常時の特徴を表現できる特徴量を用いる必要がある．特徴量を構成するベクトル次第で特徴空間の分布が変化するため，複数の特徴量について考慮する必要がある．

2. 初期重み
   初期重みとしては AdaBoost に基づく場合，各サンプルに対して「1/全サンプル数」を重みの値として与える．しかしながら分類しようとする 2 クラスのサンプル数に偏りが見られる場合，弱識別器が増えるにつれてその偏りが識別結果に影響を与える可能性がある．そこで各クラスに対して，「1/各クラスのサンプル数」の重みを与えるという方法もある．

3. 弱識別器
   弱識別器はアルゴリズムの数だけ適用の可能性が考えられる．特徴空間に対して適切なアルゴリズムを用いることが必要である．Boosting の特徴として，弱い識別器を複数回用いることから，比較的処理の軽いアルゴリズムを用いることも重要であるといえる．弱識別器の例としては, LDA や 2 分木, SVM, 特徴量に直接閾値を与える方法などが挙げられる．

4. 弱識別器の学習条件
   Boosting の各学習ステップで最良となる識別境界面を求める条件である．識別境界面を決める際には，トレードオフの関係である FalsePositive(FP) 及び FalseNegative(FN)*を考慮する必要がある．FP と FN になったサンプルの重み合計が最小となるように学習する方法や，FP と FN の差を最小にする方法がある．

5. 重み更新
   AdaBoost においては弱識別器の信頼度 (重要度) に基づいて重みが更新される．特定のクラスについて識別を重視したい場合は信頼度による関数に加えてコストを設定する方法もある

* 本研究においては FalsePositive は感染通信を正常と誤った割合，FalseNegative は正常通信を感染と謝った割合となる．

(CS-AdaBoost[26]).

6. 識別結果決定方法
   識別を行う際の結果の決定方法としては，1 サンプルごとに正常・感染を判定する方法が考えられる．また今回はトラヒックデータに着目していることから，時系列の変化も考慮できるため，連続したサンプルの結果を元にした判定も可能である．

第 4 章

マルウェアトラヒック検知実験

本章では，前章で述べた提案手法を用いたマルウェアトラヒック検知実験の結果を報告する．本実験の概要について説明し，実験諸元を述べる．最後に検知実験の結果と考察を示す．

4.1 実験概要

今回 Boosting の適用効果を確認するため，評価実験を行った．実験の概要を以下に述べる．

本実験ではトラヒックデータのパケットヘッダ情報から特徴量を取得し識別に用いる．特徴量として従来の研究で用いられてきた特徴量 [16, 18, 20, 21] を参考に特徴ベクトルを作成した．正常時と感染時の特徴ベクトルの分布に対して，これらを分類するための識別面を求める．識別面を決める手段として，特徴量の値に対して直接閾値を与える方法と線形判別分析 (Linear Discriminant Analysis) により求める方法の 2 通りを用いる．同時に AdaBoost の規則に従い，弱識別器を求め，最終的な識別器を決める．学習用データから求めた識別器を，テスト用データに適用することで最終的な結果を求める．

実験の概要を図 4.1 に示す．

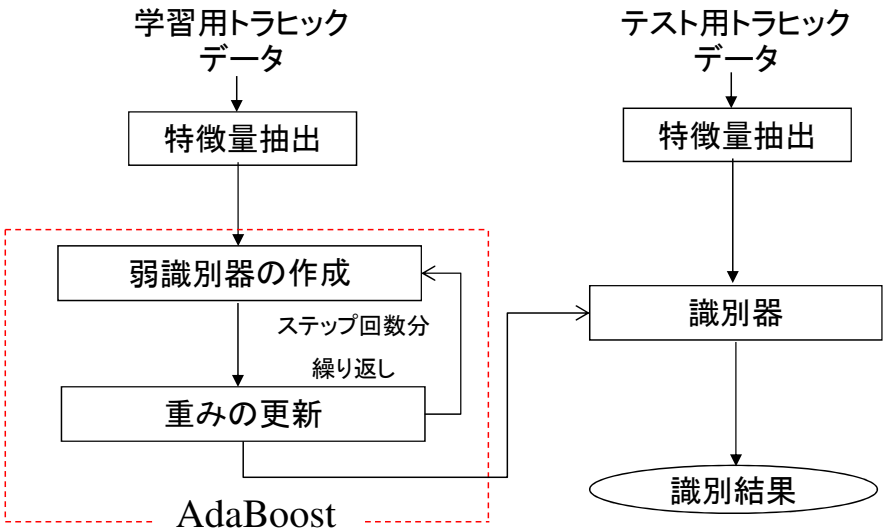

図 4.1　本実験の概要

4.2　実験諸元

4.2.1　実験データ

実験データとして，正常時通信データと感染時通信データを用意する必要がある．そこで，正常時トラヒックデータとしては，イントラネットより取得したデータ，一方感染時トラヒックデータとして CCC2010 を用いた．

CCC2010 のデータ取得期間に合わせて，正常時トラヒックデータと感染時トラヒックデータはともに 2010 年 3 月 5 日から 11 日までのデータを使用した．また，正常時トラヒックデータに関しては，利用頻度が高い 10 時から 17 時までのデータを用い，感染時データに関しては CCC2010 内の情報をもとに明らかにマルウェアに感染していると考えられるデータを切り出し用いた．

また今回の実験では試行回数を増やすためクロスバリデーションを行った．学習データとテストデータの組み合わせを以下の図 4.2 に示す．今回の実験では Session1 から Session4 までの試行結果の合計から結果を求めた．

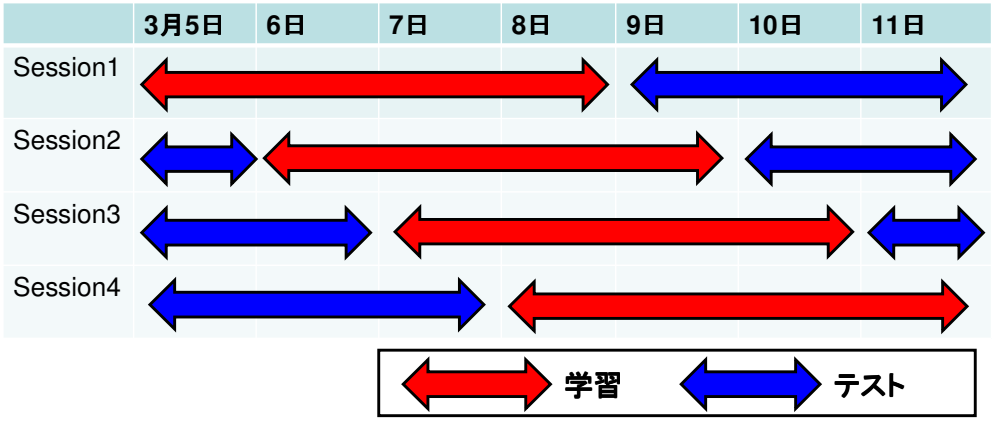

図 4.2　クロスバリデーション

4.2.2　特徴量取得

3.1.1 節でも示した通り，本実験ではトラヒックデータのヘッダ情報のみを用いる．また今回は 60 秒間に流れるパケットから得た各特徴量を取得したものを 1 サンプルとする．なお送受信の方向は区別していない．

特徴量抽出のイメージと取得した特徴量に関して，図 4.3 に示す．

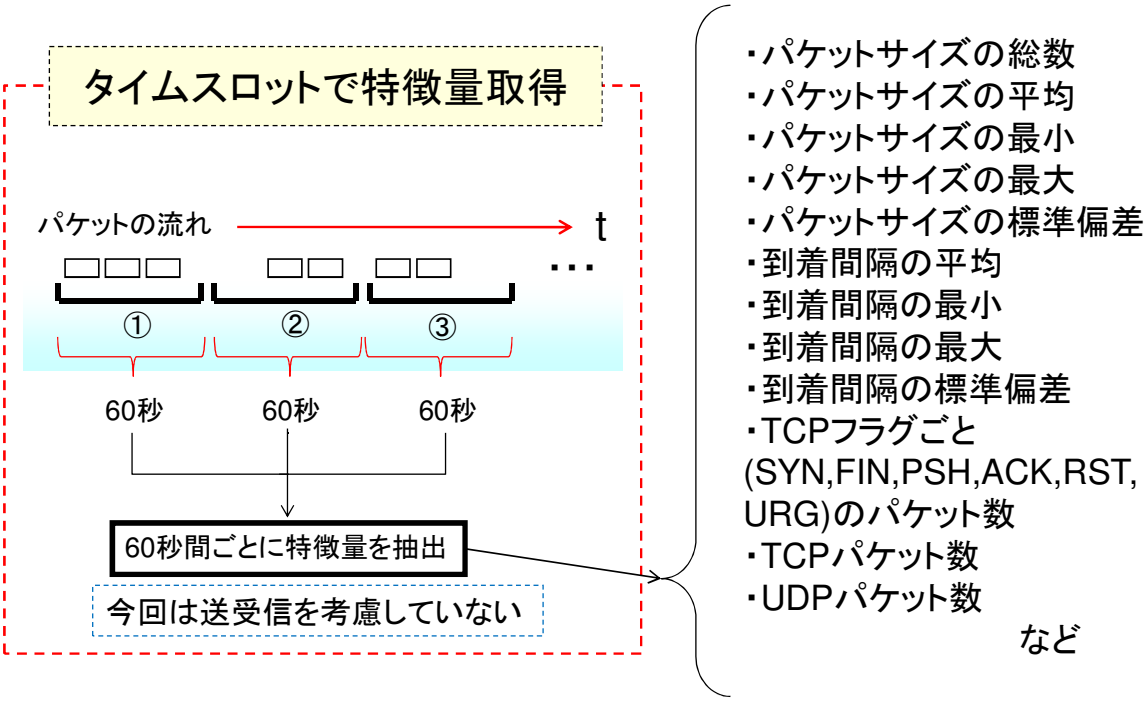

図 4.3　特徴量取得

4.2.3　識別器の構成

4.2.3.1　特徴量

Boosting の効果を確認するという観点から，従来の研究で用いられた特徴量を参考にする．特徴ベクトルとして，2 次元のベクトルを考える．4.2.2 節で示した特徴量群の中から 2 つを組み合わせて特徴量ベクトルを作成した．また複数の特徴量を用いる際との比較のため特徴量を 1 つのみでも識別を行った．

4.2.3.2　初期重み

本実験において正常時通信と感染時通信のサンプル数に大きな差が現れなかった．サンプル数の違いによる識別結果の偏りは見られないと考え，初期重みは従来どおり

$$\omega_0(x) = \frac{1}{\text{全サンプル数}}$$

とした．

4.2.3.3　弱識別器

弱識別器として，閾値を用いた判別器及び線形判別分析 (LDA) を用いる．

■閾値を用いた判別器　トラヒックデータから得た 2 つの特徴量の組み合わせに対して，閾値を用いて正常・感染を判定する弱識別器を用いる．AdaBoost の各ステップで与えられた学習データから学習して求まった閾値を a，特徴量の値を s，弱識別器を

$$h_t(x) = \begin{cases} +1, & (s \geqq a) \\ -1, & (s < a) \end{cases} \tag{4.1}$$

とする．正常と感染の判定を逆にしたものも含めて，各特徴量に関して学習を行い，4 つの中から最も誤り率の低いものをそのステップの弱識別器とする．概要を図 4.4 に示す．

■線形判別分析　その他の弱識別器として線形判別分析 (LDA) を考える [25]．LDA を用いることで正常，感染の分布状態から特徴を反映した境界面を引くことが期待できる．

LDA は特徴空間上の 2 クラスのサンプル分布からこの 2 クラスを識別するのに最適な 1 次元軸を求めるためのアルゴリズムである．具体的にはクラス内変動・クラス間変動比を最大にする 1 次元軸をもとめることで識別境界面を得る．概要を図 4.5 に示す．

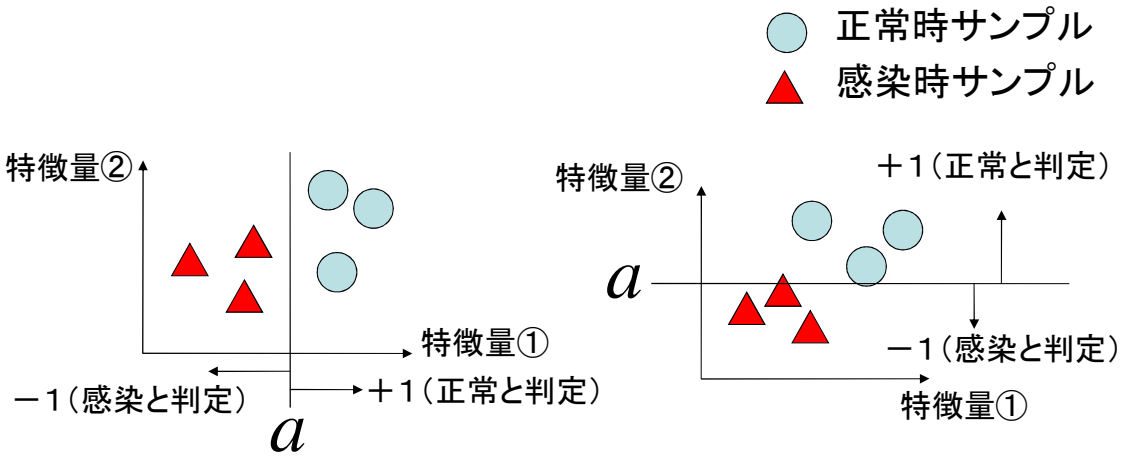

図 4.4　閾値を用いた判別器の概要

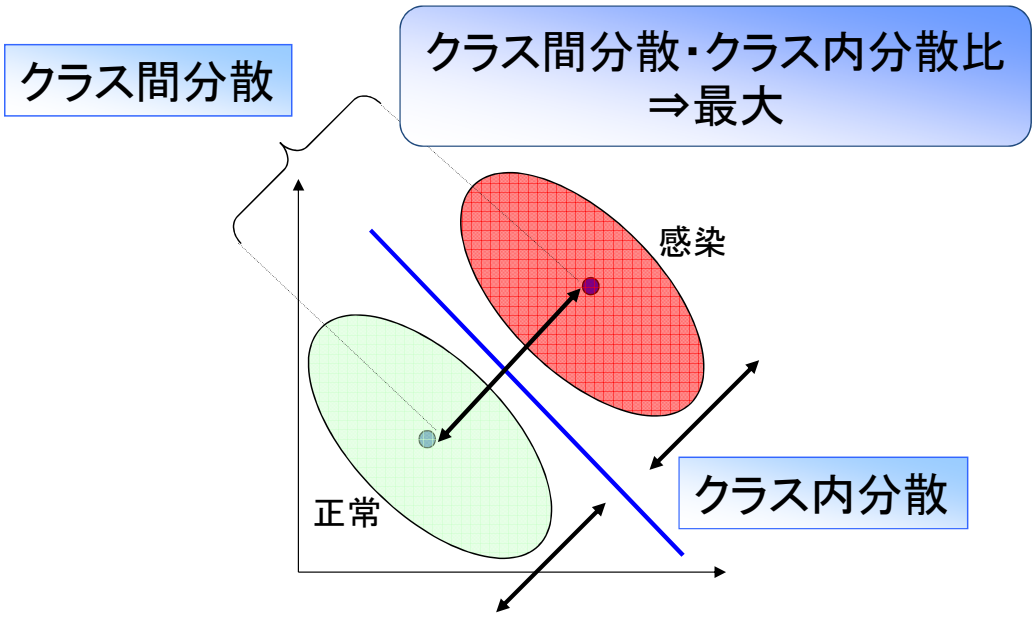

図 4.5　LDA の概要

4.2.3.4　弱識別器の学習条件

誤識別した重みの和 (FP+FN) が最小となる値で識別境界面を求めた.

$$\text{誤識別した重みの和} = \frac{\text{正常時と誤ったサンプルの重み合計}}{\text{感染時サンプルの重み合計}} + \frac{\text{感染時と誤ったサンプル重み合計}}{\text{正常時サンプルの重み合計}}$$

4.2.3.5　重み更新方法

AdaBoost の基本的な更新方法を用いた．弱識別器の信頼度 α_t に基づく関数

$$D_{t+1}(i) = \begin{cases} D_t(i)\exp(-\alpha_t), & \text{(正しく識別できたサンプル)} \\ D_t(i)\exp(\alpha_t), & \text{(間違えたサンプル)} \end{cases} \tag{4.2}$$

4.2.3.6　識別結果決定方法

今回の実験では，トラヒックデータの時系列に関する検討がまだ十分といえないため，1 サンプルごとに感染の有無を判定した.

4.2.3.7　今回の構成のまとめ

今回の実験における AdaBoost を用いた識別器の構成をまとめたものを表 4.1 に示す．表 4.1 において

- 訓練誤差: 学習用データに対して識別を行った際の誤りの割合
- 汎化誤差: テスト用データに対して識別を行った際の誤りの割合

となっている．

表 4.1　識別器の構成

	条件
初期重み	1/全サンプル数
弱識別器	閾値及び線形判別分析 (LDA)
弱識別器の学習条件	FalsePositive 及び FalseNegative の重みの合計が最小となる値
重み更新	弱識別器の信頼度に基づく関数
識別結果判定方法	1 サンプルごとに感染の有無を判定
訓練誤差・汎化誤差の算出方法	ラベルと違うサンプル数/全サンプル数

4.3　実験結果

4.3.1　識別境界面

[16, 18, 20, 21] を参考に，従来の研究で比較的効果が得られていた特徴量の組み合わせで AdaBoost の効果を確かめた．

識別境界面の例を図 4.6,4.7 に示す．図 4.6 は横軸がパケットサイズの標準偏差，縦軸が到着間隔の標準偏差，図 4.7 は横軸がパケットサイズの標準偏差，縦軸が UDP パケット数となっている．それぞれに正常時通信と感染時通信が示されており，閾値及び LDA を用いた識別境界面が示されている．図 4.6 ではグラフの周りの数字が，図 4.7 では凡例の数字が，識別境界面の作成された AdaBoost のステップ回数に対応している．

どちらの図でも複数の識別面が描かれる様子が見え，AdaBoost が適用されていく様子を確認できる．

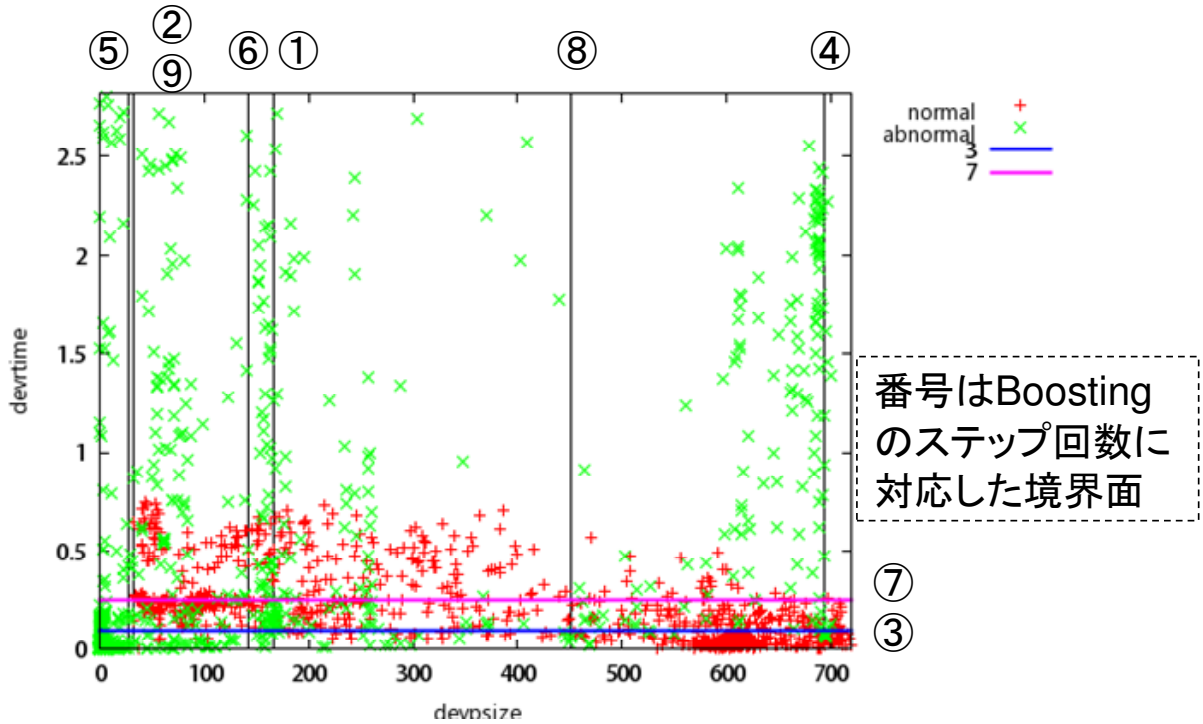

図 4.6　特徴空間および識別境界面 (閾値)

4.3.2　識別率

次に AdaBoost のステップ回数を 100 回とし，特徴量の組み合わせを変化させた際の識別率を表 4.2 に示し，特徴量を 1 つのみ用いた際の識別率を表 4.3 に示す．表 4.3 では表 4.2 で用いた特

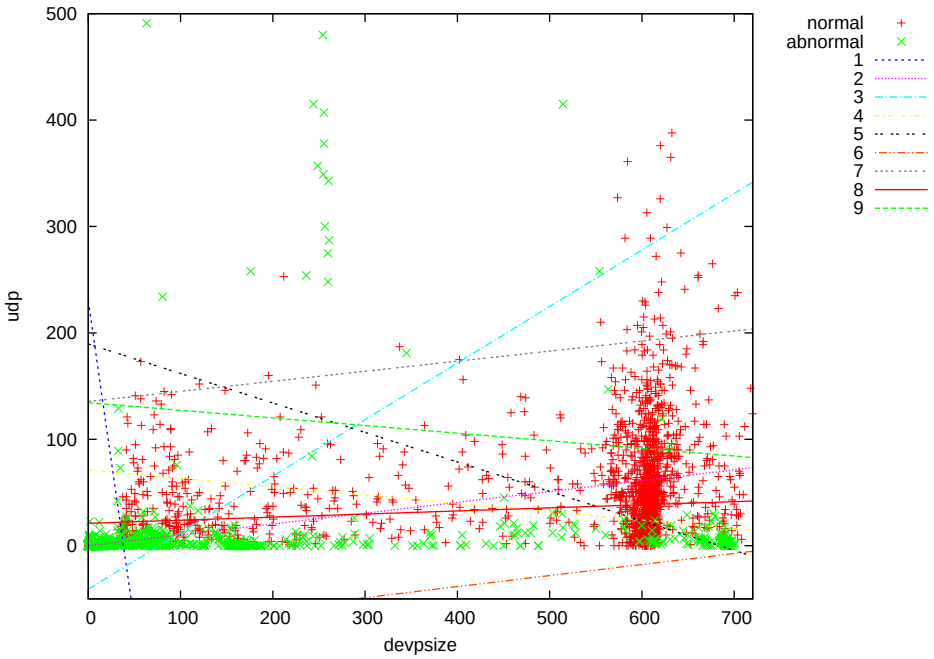

図 4.7　特徴空間および識別境界面 (LDA)

徴量を 1 つずつ用いた場合となっている．

識別率の指標として True Positive Rate，True Negative Rate(以下それぞれ TPR，TNR) も用いる．TPR，TNR とはそれぞれ，感染時通信を感染と識別できた割合，正常時通信を正常と識別できた割合である．表内では TPR/TNR の形で値を示している．

表 4.2 において弱識別器に閾値，LDA を用いたいずれの場合においても一定の精度 (80% 前後) で正常通信，感染通信を特定することが出来ている．番号 1 の特徴量の組み合わせでは弱識別器に閾値を用いた場合に識別率が高く，一方でそれ以外の組み合わせでは LDA を用いた場合のほうが識別率が高くなっている．分布によって適した識別器に違いがあり，特徴量ごとに識別器を変化させることで精度向上できる可能性がある．

TPR，TNR に関しては番号 4 の特徴量の組み合わせを除いて，TNR が TPR を上回っている．感染通信を正常通信と誤ることは望ましくないため，良い傾向にあると言える．弱識別器決定の際の閾値を FN を重視するように設定することで，TNR の精度を向上出来る可能性があるため，今後も検討が必要である．

また表 4.2 と表 4.3 の比較をすると，識別率は特徴量を組み合わせた場合の方が全体として高い値を示している．例えば SYN パケット数と FIN パケット数に着目した場合，単一で用いた場合に比べて組み合わせた場合の方が識別率が上がっている．他の組み合わせに関しても同様の結果が現れている．ピーク値の比較としては，表 4.2 における識別率 90.9%と TNR93.9%は表 4.3 における識別率 90.5%と TPR92.1%を上回っている．以上より，特徴量を複数組み合わせることの有効性が示されたといえる．

表 4.2　特徴量組み合わせ時の識別率

		識別率 [%]		TPR/TNR[%]	
番号	特徴量組み合わせ	閾値	LDA	閾値	LDA
1	パケットサイズ平均-ACK パケット数	84.2	82.6	77.0/92.0	72.0/93.9
2	パケットサイズ標準偏差-到着間隔標準偏差	81.3	86.8	77.7/85.0	81.5/92.4
3	パケットサイズ標準偏差-UDP パケット数	89.6	90.9	86.7/95.3	87.7/91.6
4	SYN パケット数-FIN パケット数	81.1	84.1	86.0/82.9	86.6/81.5
5	SYN パケット数-TCP パケット数	78.4	83.4	76.2/80.5	75.0/92.2

表 4.3　単一の特徴量を用いた時の識別率

特徴量	識別率 [%]	TPR/TNR[%]
パケットサイズ平均	78.0	73.1/83.2
パケットサイズ標準偏差	81.7	80.2/83.4
到着間隔標準偏差	47.1	43.7/50.7
SYN パケット数	71.4	73.7/68.9
FIN パケット数	78.3	89.8/66.0
ACK パケット数	85.0	80.8/89.5
TCP パケット数	75.4	76.6/74.3
UDP パケット数	90.5	92.1/88.9

4.3.3　訓練誤差・汎化誤差

次に訓練誤差および汎化誤差を図 4.8，4.9 に示す．訓練誤差は横軸が AdaBoost のステップ回数，縦軸が学習サンプルを用いた際の誤差率，汎化誤差は横軸が AdaBoost のステップ回数，縦軸が学習に用いたサンプルとは異なるサンプルを用いた際の誤差率を表している．また凡例の番号は表 4.2 の特徴量組み合わせの番号に対応し，「line」は閾値を直接特徴量に与えた場合，「lda」は LDA を用いた場合の結果を示している．

図 4.8，4.9 から弱識別器に閾値を用いた場合，LDA を用いた場合どちらにおいても AdaBoost のステップ回数が増えるにつれて訓練誤差，汎化誤差が減少している．単一の識別器を用いた場合を 1 ステップ目の結果と考えることが出来るため，AdaBoost を適用することで検知精度が向上することを確認できた．

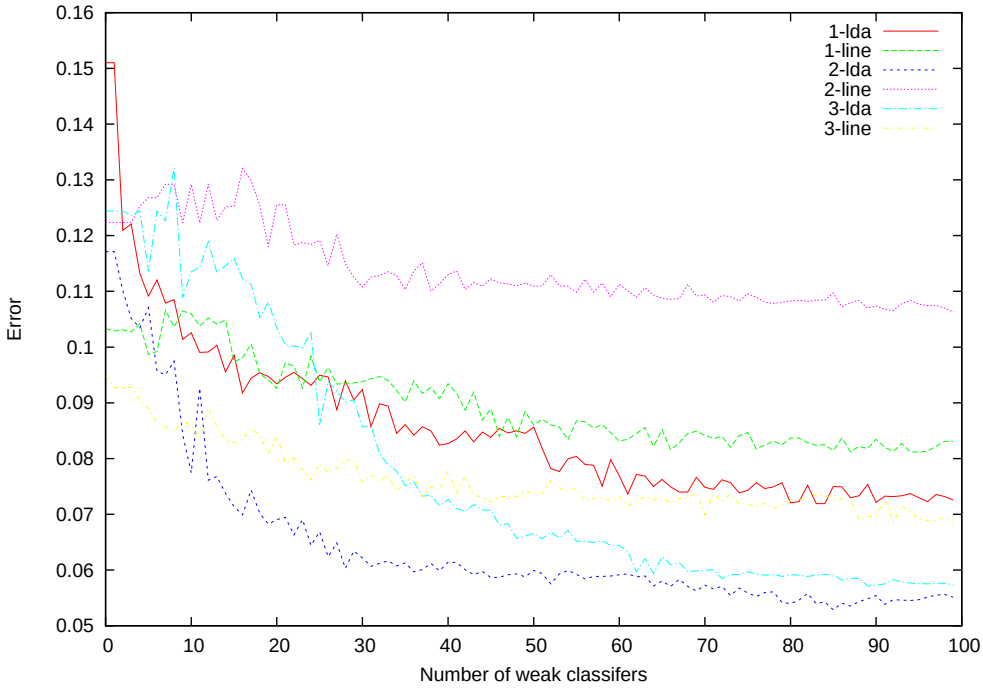

図 4.8　訓練誤差

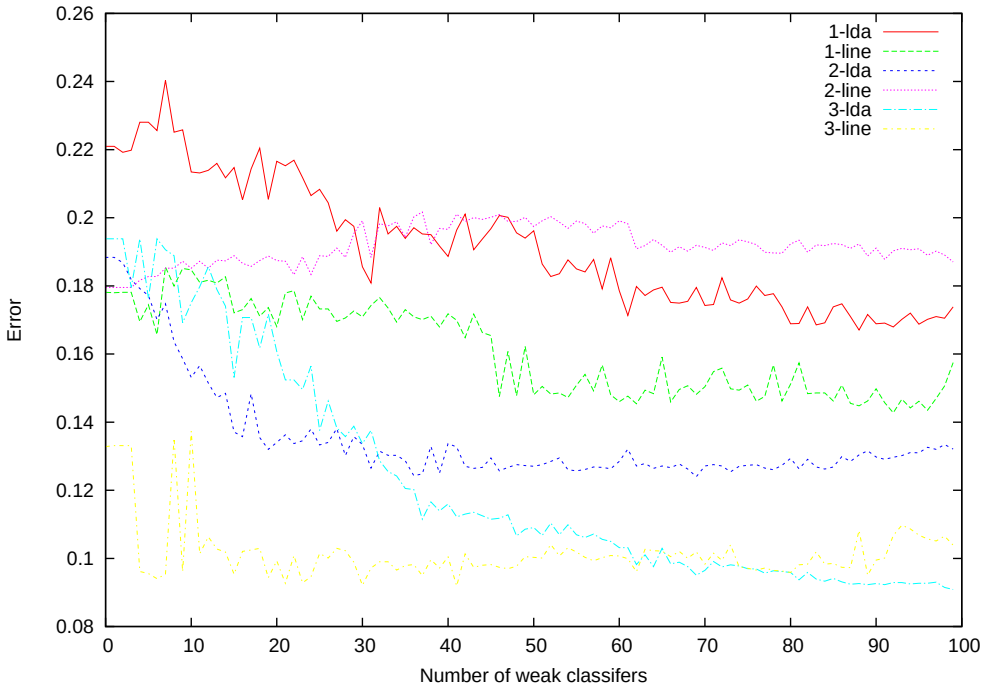

図 4.9　汎化誤差

第 5 章

結論

5.1 まとめ

本論文では，マルウェアについての基本性質，対策技術等を説明した．それらを踏まえた上でマルウェアトラヒックを検知するための手段として Boosting を用いる識別方式を提案した．提案手法の評価のために弱識別器として特徴量に直接閾値を与える方法と LDA を用いた方法の 2 通りでマルウェアトラヒックの検知実験を行った．

その結果，識別率や訓練誤差，汎化誤差といった面で精度向上が示され，Boosting 適用の有効性を実験的に確認することが出来た．従来用いられた特徴量を用いた際にも精度の向上が確認された．また特徴量を組み合わせる場合，1 つだけ用いる場合についても実験を行い，特徴を組み合わせた識別の有効性を示すことが出来た．

本研究により Boosting という技術がマルウェア検知に適用可能であることを示した．Boosting の適用の方法という点ではまだ検討の余地があるといえ，今後の検討で長所・短所の整理をし実用化を目指していく．

5.2 今後の課題

本論文で検討を行った項目に関して，以下のような課題が挙げられる．

- 特徴量の調査
  今回の実験では過去の文献を参考に特徴量を選択した．感染時，正常時のトラヒックの特徴を調査することで適切な特徴量を用いることが出来れば，さらなる精度向上が見込めると考えられる．

- 識別器の構成方法
  Boosting の効果を確認することはできたが，識別器の構成による精度の評価はまだ十分ではないといえる．そこで工程ごとに条件を変化させた場合の Boosting の効果の違いを検討していく必要がある．

- トラヒックデータの送受信方向

今回はトラヒックデータの送受信方向には区別を付けなかった．マルウェアの活動には通信方向別に特徴が現れる可能性があるため, 受信方向, 送信方向に区別をつけた検討を行う．

- サンプルの取得時間の妥当性
  今回は 60 秒間という単位でサンプルを取得した．しかしその妥当性は確実とは言えない．単位を変えることで特徴量の表現の仕方も変化し，特徴が現れる可能性もある．よってサンプル取得時間を変化させた検討を行う．

- 時系列を考慮した識別方法
  トラヒックデータは時間的な変化が考えられるので，前後のサンプルを加味した判定も検討する必要がある．

謝辞

本研究を進めるにあたり，終始懇切丁寧な御指導，御助言を賜りました小松尚久教授に心から深く感謝の意を表します．

また，共同研究者として様々な御意見を賜りました NTT コミュニケーションズ株式会社の畑田充弘様，本研究において様々な御助言を頂きました本学理工学研究所研究員の市野将嗣氏，修士 1 年の市田達也氏，学士 4 年の川元研二氏に深く御礼申し上げます．そして，日頃から討論に御参加頂いた小松研究室の皆様に深く感謝いたします．

2011 年 2 月 4 日

森 悠樹

参考文献

[1] 瀬戸洋一他編著，情報セキュリティ概論，日本工業出版，2007，ISBN: 978-4819019170.

[2] トレンドマイクロ株式会社，“インターネット脅威マンスリーレポート - 2010 年上半期・6 月度，” Jun. 2010，http://jp.trendmicro.com/jp/threat/security_news/monthlyreport/article/20100702082212.html.

[3] G Data Software AG，“マルウェアレポート 2009 年下半期，” Feb. 2010，http://www.gdata.co.jp/securitylabs/whitepaper/index.htm.

[4] 畑田充弘，中津留勇，秋山満昭，三輪信介，“マルウェア対策のための研究用データセット～mws 2010，” MWS2010，Oct. 2010.

[5] 井上大介, 中尾康二, “1. マルウェアって?(特集 マルウェア)，” 情報処理, vol.51, pp.237–243, Mar. 2010.

[6] I. Nikkei Business Publications，“ヤフーがクラッカの攻撃を受けて 3 時間機能停止、イーベイ、バイ・ドット・コムなども相次いで被害，” Feb. 2000，http://www.nikkeibp.co.jp/archives/094/94127.html.

[7] インターネットコム株式会社，“韓国、ワームの影響でインターネットが約 9 時間マヒ，” Jan. 2003，http://japan.internet.com/public/news/20030127/20.html.

[8] T. Carothers, “Large botnet in the netherlands taken down,” Oct. 2005, http://isc.sans.org/diary.php?storyid=742.

[9] P. Bacher, T. Holz, M. Kotter, and G. Wicherski, “Know your enemy: Tracking botnets -using honeynets to learn more about bots-,” Mar. 2005, http://www.honeynet.org/book/export/html/50.

[10] 金井瑛，“通信の類似性に着目したネットワークインシデント検知手法，” 修士論文，慶應義塾大学大学院，Mar. 2009.

[11] ClamAV, “ClamAV,” 2010, http://www.clamav.net/.

[12] Cisco System, “IronPort,” 2010, http://www.ironport.com/jp/.

[13] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, and N. Modadugu, “The ghost in the browser analysis of web-based malware,” In Proceedings of the First USENIX Workshop on Hot Topics in Understanding Botnets, 2007.

[14] 藤原将志，寺田真敏，安部哲哉，菊池浩明，“マルウェアの感染方式に基づく分類に関する検討，” 情報処理学会 CSEC 研究報告，vol.PRMU97，no.21，pp.177–182，Mar. 2008.

[15] NTT 情報流通プラットフォーム研究所，“マルウェア対策技術，” NTT 技術ジャーナル，Mar. 2010.

[16] S.Kondo, and N.Sato, “Botnet traffic detection techniques by C&C session classification

using svm," IWSEC2007, Oct. 2007.

[17] 市野将嗣，坂野鋭，小松尚久，"核非線形相互部分空間法による話者認識，" 信学論 (D-Ⅱ)，no.8，pp.1331–1338，2005.

[18] 阿部義徳，田中英彦，"C&C セッション分類によるボットネットの検出手法の一検討，" FIT2007，Sep. 2007.

[19] 東角芳樹，鳥居悟，"DNS 通信の挙動からみたボット感染検知方式の検討，" MWS2008，Oct. 2008.

[20] 及川達也，和泉勇治，太田耕平，加藤寧，根元義章，"統計的クラスタリング手法によるネットワーク異常状態の検出，" 信学技報，vol.NS2002，no.143，pp.166–173，Oct. 2002.

[21] 宮本貴朗，小島篤博，泉正夫，福永邦雄，"SVM を用いたネットワークトラヒックからの異常検出，" 電子情報通信学会論文誌，vol.B，通信 J87-B(4)，pp.593–598，Apr. 2004.

[22] 石井健太郎，上田修功，前田英作，村瀬洋，わかりやすいパターン認識，オーム社，1998.

[23] Y．Freund, R．E．Schapire, "A decision theoretic generalization of on-line learning and an application to boosting，" Journal of Computer and System Science，vol.55，no.1，pp.119–139，1997.

[24] 三田雄志，"Adaboost の基本原理と顔検出への応用 CVIM 研究会チュートリアルシリーズ，" CVIM，vol.159，pp.265–272，May 2007.

[25] 浅見太一，岩野公司，古井貞熙，"マルチストリーム話者照合におけるブースティングに基づく重み最適化法の検討，" 信学技報，vol.SP2004，no.110，pp.85–90，Dec. 2004.

[26] M. ong, and D. Xiaoqing, "Real-time multi-view face detection and pose estimation based on cost-sensitive adaboost," TSINGHUA SCIENCE AND TECHNOLOGY, vol.10, no.2, pp.152–157, Apr. 2005.

付録 A

線形判別分析 [22]

唇輪郭抽出の際に使用している線形判別分析について補足する．

A.1 Fisher の線形判別分析

線形判別法は，ある基準に基づいて特徴空間から識別に適した部分空間を決定する方法，すなわち，特徴空間をより次元の小さい部分空間に変換する方法である．その簡便さと高い有効性のため，パターン認識の応用として広く使われていると同時に，統計学の分野では判別分析 (discriminant analysis) と呼ばれ，多変量解析の基本技法として知られている．

パターン認識において最もよく利用されるのは 2 クラスに対する線形判別であり，これを Fisher の判別分析法 (Fisher's linear discriminant method) と呼ぶ．Fisher の判別分析法は，特徴空間上にある 2 クラスのパターン分布から，図 A.1 のようなクラスを識別ために最適である 1 次元軸を求める手法である．

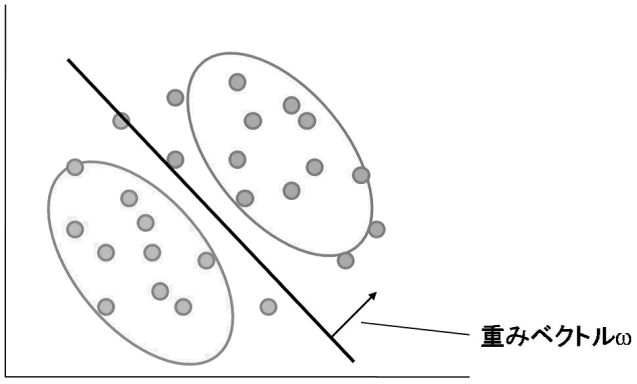

図 A.1 Fisher の判別分析

クラス ω_i の変動を表す行列である変動行列 S_i を

$$S_i = \sum_{x \in X_i} (x - m_i)(x - m_i)^t \tag{A.1}$$

と定義する．ここで m_i はクラス ω_i のパターン平均である．変動行列 S_i は，クラス ω_i に属するベクトル x とクラス平均 m_i との差の 2 乗和の形で定義される．次に，2 クラスの全特徴ベクト

ルを用いて，クラス内変動行列 S_W とクラス間変動行列 S_R をそれぞれ

$$S_W = S_1 + S_2 = \sum_{i=1,2} \sum_{x \in X} (x - m_i)(x - m_i)^t \tag{A.2}$$

$$S_R = \sum_{i=1,2} n_i (m_i - m)(m_i - m)^t = \frac{n_i n_2}{n} (m_1 - m_2)(m_1 - m_2)^t \tag{A.3}$$

と定義する．m は全パターンの平均，n_i はパターン数，n は全パターン数を表す．式 A.3 から，クラス平均間の距離によって決まる量である．ここで，d 次元特徴空間から 1 次元空間への変換を表す $(d, 1)$ 行列を ω とする．このとき，パターン x を ω によって変換したパターンはスカラー量であり，これを y とすると

$$y = \omega^t x \tag{A.4}$$

と書ける．変換された空間でのクラス平均 $\tilde{m}_i$ は

$$\tilde{m}_i = \frac{1}{n_i} \sum_{y \in Y_i} y = \frac{1}{n_i} \sum_{x \in X} \omega^t x = \omega^t m_i \tag{A.5}$$

となる．Y_i は変換後の空間での ω_i に属するパターン集合を表す．変換後の空間上でのクラス内変動行列 $\tilde{S_W}$，クラス間変動行列 $\tilde{S_R}$ も同様に求めることができ，式 A.4 および式 A.5 を用いて

$$\tilde{S_W} = \tilde{S}_1 + \tilde{S}_2 = \sum_{i=1,2} \sum_{y \in Y} (y - \tilde{m}_i)^2 = \omega^t S_w \omega \tag{A.6}$$

となる．ここで，$\tilde{S}_i (i = 1, 2)$ はクラス ω_i に属するパターンの変換後におけるクラス内変動であり，$\tilde{S}_i$(式 A.1) と同様にして

$$\tilde{S}_i = \sum_{y \in Y} (y - \tilde{m}_i)^2 \tag{A.7}$$

で定義される．このとき，ω は $(d, 1)$ 行列であるから，$\tilde{S_W}$ および $\tilde{S_R}$ はスカラー量であり，変換後の 1 次元空間におけるクラス平均と分散をそれぞれ $\tilde{m}_i$，$\tilde{\sigma^2}$ とおくと

$$\tilde{S_W} = n_1 \tilde{\sigma_1^2} + n_2 \tilde{\sigma_2^2} \tag{A.8}$$

$$\tilde{S_R} = n_1 (\tilde{m}_1 - \tilde{m})^2 + n_2 (\tilde{m}_2 - \tilde{m})^2 = \frac{n_1 n_2}{n} (\tilde{m}_1 - \tilde{m}_2)^2 \tag{A.9}$$

となる．

フィッシャーの判別手法の基本的な考え方は，クラス間変動のクラス内変動に対する比，クラス内変動・クラス間変動比を最大にする 1 次元軸を求めることにある．すなわち，変換後の空間において 2 つのクラスがよく分離するためには，$\tilde{S_W}$ がなるべく小さく，そして $\tilde{S_R}$ がなるべく大きくなるように変換 ω を定めるわけである．このクラス内変動・クラス間変動比を $J_S(\omega)$ と表すと，

$$J_S(\omega) = \frac{\tilde{S_R}}{\tilde{S_W}} = \frac{n_1 n_2}{n} \frac{(\tilde{m}_1 - \tilde{m}_2)^2}{n_1 \tilde{\sigma^2} + n_2 \tilde{\sigma^2}} = \frac{\omega^t S_R \omega}{\omega^t S_W \omega} \tag{A.10}$$

となる．この評価基準 $J_S(\omega)$ をフィッシャーの評価基準 (Fisher's criterion) と呼ぶ．この J_S を最大にする ω を求める問題は,

$$\tilde{S_W} = \omega^t S_W \omega = I \tag{A.11}$$

という制約条件の下で

$$\tilde{S_B} = \omega^t S_B \omega \tag{A.12}$$

を最大にする変分問題に帰着する．ただし I は，$\tilde{d}$ 次元単位行列である．λ をラグランジュ乗数とし,

$$J(\omega) = \omega^t S_B \omega - \lambda(\omega^t S_W \omega - I) \tag{A.13}$$

を ω で変微分して 0 とおくと，S_B，S_W は対称行列であるから

$$S_B \omega = \lambda S_W \omega \tag{A.14}$$

を得る．したがって，S_W が正則であるならば

$$(S_W^{-1} S_B - \lambda I)\omega = 0 \tag{A.15}$$

となるので，$S_W^{-1} S_B$ の最大値を λ_1 とすると

$$\max J_S(A) = \lambda_1 \tag{A.16}$$

が得られる．また J_S を最大にする ω は，最大固有値 λ_1 に対する固有ベクトルとして定まる．さらに式 A.14 より

$$\lambda S_W \omega = S_B \omega = \frac{n_1 n_2}{n}(m_1 - m_2)(m_1 - m_2)^t \omega \tag{A.17}$$

となり，$(m_1 - m_2)^t \omega$ がスカラー量であることに注意すると,

$$\omega \propto S_W^{-1}(m_1 - m_2) \tag{A.18}$$

となる．こうして求まる変換行列 ω によって変換された特徴空間は，クラス内変動・クラス間変動比を最大にする 1 次元空間となる．線形判別法における変動比最大という基準は変換後の識別を考慮した基準であり，この点が KL 展開の場合とは異なる．この線形判別法で特に注意が必要な点は，線形判別法によって決まるのが空間 (軸) のみであって，軸上に設けるべき識別のための境界は定まらないという点である．このように決定境界の法線ベクトルは求まるが，境界の位置が決まらない場合には別の方法によって決定しなければならない．その方法としては，例えば次のようなものが考えられる．

- 変換後のクラス平均の中点を境界とする方法
- 変換後の各クラス毎の分散で内分する方法,
  あるいは変換後の各クラス毎の標準偏差で内分する方法
- 事前確率も考慮して内分を行う方法

付録 B

CCC Dataset について

今回実験で使用した CCCDataset[4] について補足する．

B.1　データの概要

CCC Dataset は，Cyber Clean Center(CCC) によって収集されたマルウェア対策研究を対象とした研究用データセットである．マルウェアの対策研究をする上で「共通なデータセットがない」という課題があり，提案手法の評価に用いるマルウェアのサンプルや，感染前後の通信データといった研究用データセットとして提供されている．これによりマルウェア研究の成果の比較が容易になり，新たに研究を始める人にとっても参入が容易になったため，研究の裾野が広がったといえる．

研究成果を共有する場・切磋琢磨する環境として開催された「マルウェア対策研究人材育成ワークショップ (MWS)」において使用された．MWS は 2008 年から現在まで毎年開催され，その都度 CCCDataSet2008，2009，2010 と用意されている．これにより過去のデータとの傾向を比較分析が可能になっている．

B.2　CCC Dataset 2010 について

本節では今回の実験で用いた CCC Dataset 2010 の概要について述べる．

CCC2010 では，マルウェアの解析技術の研究のための「マルウェア検体」，感染手法の検知ならびに解析技術の研究のための「攻撃通信データ」，ボットの活動傾向把握技術の研究のための「攻撃元データ」の 3 つから構成される．以下，それぞれについて概要を述べる．

B.2.1　マルウェア検体

ハニーポットで収集したマルウェア検体のハッシュ値 (MD5，SHA1)50 個をテキスト形式で記載したファイルであり，以下の観点から選定している．

1. 解析結果を照合できる検体：10 検体
2. 未知検体：40 検体

1 は特徴的な機能を有し，技術的に目を通しておきたい検体である．これは事前に静的解析が完了しており，解析精度の評価に活用することを考慮している．具体的には，ユーザーの特定の動作をトリガとして動作する検体や，独自かつ高度な通信プロトコルを使用する検体である．2 は 2010 年 1 月から 3 月までに収集した未知検体のうち，収集日が偏らないように任意で選定した検体であり，相当数の検体の自動解析や自動分類を考慮している．

B.2.2　攻撃通信データ

ハニーポットの通信を tcpdump でパケットキャプチャした libpcap 形式のファイルである．ハニーポットは，ホスト OS 上の 2 台 (honey001，honey002) のゲスト OS がそれぞれインターネット接続されており，パケットキャプチャはホスト OS 上で行っている．ゲスト OS は 2 台とも WindowXP SP1 であり，これらは定期的にクリーンな状態にリセットされる．データ収集日は 2010 年 3 月 5 日から 3 月 11 日，総パケット数が 22,486,674 パケット，約 3.5GB のデータサイズである．

B.2.3　攻撃元データ

2009 年 5 月 1 日から 2010 年 4 月 30 日までの 1 年間にハニーポットで記録したマルウェア取得時のログで，表 B.1 に示す項目を 1 レコードとして記録した csv 形式のファイルである．Windows2000 の ISP にそれぞれ接続された 92 台のハニーポットで記録された約 156MB のデータである．攻撃元データの基本情報を表 B.2 に示す．

表 B.1　攻撃元データのログ項目と例

ログ項目	例（一部を*でマスク）
マルウェア検体の取得時刻	2010-03-05 03:02:41
送信元 IP アドレス	honey001
送信元ポート番号	1028
宛先 IP アドレス	**. 243.167
宛先ポート番号	5824
TCP または UDP	TCP
マルウェア検体のハッシュ値（SHA1）	*****bc3c850cf68a39c9e8013f2169d408a9d90
ウイルス名称	WORM_DOWNAD.AD
ファイル名	C:/WINDOWS/system32/dhnlr.dll

マルウェア検体のダウンロードを開始した時刻がマルウェア検体の取得時刻であり，ゲスト OS の Windows 上でのファイル作成日時となる．送信元 IP アドレスまたは宛先 IP アドレスにおい

て，ハニーポットの IP アドレスは各ハニーポットに対応する ID(honey001〜honey092) に置換されて記載されている．ウイルス名称は収集日の翌日午前 3 時の最新パターンファイルを適用したウイルススキャナ (トレンドマイクロ社製) により判定された名称であり，マルウェアとして判定されなかったものは UNKNOWN と表記される．このため，パターンファイルのウイルス名称が更新された場合，同一のハッシュ値であっても，異なるウイルス名称が付与される場合がある．

表 B.2　攻撃元データの基本情報

項目	件数
全レコード数	1,162,093
TCP によるダウンロードレコード数	1,053,977
UDP によるダウンロードレコード数	108,116
ダウンロードホスト IP アドレス種類数	176,522
マルウェア検体のハッシュ値種類数	29,858
ウイルス名称種類数 (UNKNOWN 含まない)	978

B.3　感染時データの切り出し

CCC2010 の攻撃通信データには感染するまでのトラヒックデータが含まれている．今回のマルウェアトラヒック検知実験には感染時のみのデータを用いる必要があったため，攻撃通信データから感染時のトラヒックデータを切り出した．

手順を以下に示す．

1. 取得環境独自の制御パケットをフィルタリングで除外
2. ハニーポットの OS のリセット間隔で切り出す
3. ファイルサイズで感染しているかのふるい落とす
4. 残ったファイルをログと照らし合わせて，感染を確認
5. 感染攻撃の開始パケットを探し，それ以降を感染トラヒックとして抽出する