ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/228447003

Aprendizagem de Máquina para Segurança em Redes de Computadores: Métodos e Aplicações

Chapter · November 2011

DOI: 10.5753/sbc.9559.1.2

CITATION
1

READS
3,520

6 authors, including:

Márcia Henke
Federal University of Amazonas
3 PUBLICATIONS **10** CITATIONS
SEE PROFILE

Clayton Santos
Universidade Federal do Oeste do Pará
13 PUBLICATIONS **49** CITATIONS
SEE PROFILE

Angelo Eduardo Nunan
Federal University of Amazonas
8 PUBLICATIONS **65** CITATIONS
SEE PROFILE

Eduardo Luzeiro Feitosa
Federal University of Amazonas
131 PUBLICATIONS **391** CITATIONS
SEE PROFILE

All content following this page was uploaded by Eulanda Santos on 01 June 2014.

The user has requested enhancement of the downloaded file.

Capítulo

2

Aprendizagem de Máquina para Segurança em Redes de Computadores: Métodos e Aplicações

Márcia Henke, Clayton Santos, Eduardo Nunan, Eduardo Feitosa, Eulanda dos Santos, Eduardo Souto.

Instituto de Computação (IComp)
Universidade Federal do Amazonas
69077000 - Manaus - AM - Brasil

{henke, clayton.maia, nunan, efeitosa, emsantos, esouto}@dcc.ufam.edu.br

Abstract

The growth of malicious activities owing to fraud such as phishing, malicious code proliferation, Distributed Denial of Service attack, and others, may be clearly observed through Internet traffic analysis. This chapter presents Machine Learning tools which can be used for the design of effective automatic malicious activity detection systems. Moreover, it points out that Machine Learning has already been successfully applied to improve malicious activity detection rates in problems dealing with phishing, cross-site scripting, malware, denial of service and intrusion detection. Finally,it also highlights a list of research topics and open problems, as well as potential application of Machine Learning on network security domain.

Resumo

Uma breve análise do tráfego Internet comprova o crescente aumento de atividades maliciosas originadas por ações fraudulentas como phishing, proliferação de códigos maliciosos, ataques de negação de serviço, entre outras. Este capítulo apresenta ferramentas de aprendizagem de máquina (AM) que podem ser usadas para o desenvolvimento de sistemas efetivos de detecção de atividades maliciosas. Além disso, destaca que significativos avanços nessa área de aplicação já estão sendo obtidos graças ao emprego de técnicas de AM. São apresentados exemplos do uso de AM em diversos problemas de detecção, tais como: phishing, cross-site scripting, malware, negação de serviço e detecção de intrusos. Também destaca pesquisas em aberto e potenciais aplicações de AM em problemas de segurança de redes de computadores.

2.1. Introdução

Com uma rápida consulta às estatísticas de tráfego na Internet, percebe-se que o tráfego conhecido, solicitado (ou em outras palavras, legítimo) foi dominado pelo tráfego de mensagens eletrônicas não solicitadas (*spam*), atividades fraudulentas como *phishing*[1] e *pharming*[2], ataques de negação de serviço, proliferação de vírus, *worms* e outros código maliciosos (ou seja, tráfego ilegítimo, improdutivo, não desejado e não solicitado).

O instituto de segurança americano CSI (*Computer Security Institute*) [Richardson, 2010], depois de entrevistar 149 empresas nos Estados Unidos, constatou o crescimento no número de infecções por *malware*, *botnets* e *phishing*. Considerando dados de 2007 e 2010, o número de *Malware* subiu de 52% para 67% em 2010; *botnets* contabilizava 21% em 2007, e em 2010 registrou 29%; e o número de páginas *phishing*, subiu de 26% para 39% nesse mesmo período. No Brasil, o CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil) também contabilizou aumentos expressivos de atividades maliciosas em 2010: 17.628 incidentes relativos a *worms*, 198 ataques de negação de serviço e 31.008 tentativas de fraude (*phishing*) [CERT.br, 2011].

O aumento das atividades maliciosas se deve, entre outras coisas, a ineficiência das atuais soluções em identificar, reduzir e interromper esse tipo de tráfego. Tipicamente, a efetividade fornecida pelas atuais soluções é comprometida pelas altas taxas de falsos alarmes e pela falta de cooperação com outras soluções ou mesmo com a infraestrutura de rede. Além disso, muitas das soluções dependem de ativação ou interferência humana (configuração, adequação e ajuste) para funcionar.

Na busca por soluções mais efetivas e corretas, especialmente aquelas relacionadas ao tráfego Internet malicioso, muitos pesquisadores e a indústria de software de segurança vêm investindo tempo e recursos para tornar a aprendizagem de máquina uma ferramenta poderosa e eficaz na segurança de redes de computadores. A idéia é bastante simples: se as máquinas podem aprender quando um sistema está funcionando normalmente e quando ele está sob ataque, então é possível construir mecanismos capazes de, automaticamente, responder a ataques e anomalias normalmente encontradas em uma rede.

Entretanto, em termos operacionais, observa-se um desequilíbrio no uso de aprendizagem de máquina. Enquanto em alguns em ambientes operacionais existem implementações comerciais bem sucedidas como, por exemplo, os sistemas de recomendação de produtos usados pela Amazon [Lindenet al., 2003] e Netflix [Bennett et al., 2007] e os sistemas de reconhecimento de caracteres (OCR) [Vincent, 2007] e [Smith, 2007], na área de segurança parece que o uso de aprendizagem de máquina ainda está em um estágio inicial. A explicação para essa diferença será discutida nesse minicurso.

---

[1] *Phishing* é um tipo de fraude eletrônica caracterizada pela tentativa de obter informações pessoais privilegiadas através de sites falsos ou mensagens eletrônicas forjadas.

[2] *Pharming* refere-se ao ataque de envenenamento de *cache* DNS cujo objetivo é preparar terreno para atividades de *phishing.*

2.1.1. Contexto

Detecção de anomalias (também chamada de detecção de *outliers*) se refere ao problema de encontrar padrões em dados que não estão de acordo com o comportamento esperado. Sistemas de detecção de anomalias objetivam encontrar desvios inesperados de comportamento. Com base em uma "noção de atividade normal", eles relatam desvios desse perfil como alertas. A importância da detecção de anomalias se deve ao fato de que as anormalidades nos dados podem traduzir informações úteis, significantes e muitas vezes críticas sobre o domínio de aplicação em questão. Chandola et al. [Chandola et al., 2009] fornecem um vasto documento sobre detecção de anomalias, incluindo várias áreas de aplicação. Na área de segurança de redes de computadores, por exemplo, um padrão anômalo de tráfego em um computador pode significar que o mesmo foi invadido e pode, possivelmente, estar enviando dados pessoais a um destino não autorizado. Em outro exemplo, anomalias nas transações de cartão de crédito podem indicar o roubo de cartões ou de identidades.

De modo geral, a detecção de anomalia vem sendo estudada desde o século 19, o que resultou no surgimento de uma variedade de técnicas e soluções. No contexto de segurança de redes, o conceito básico de qualquer sistema de detecção de anomalias foi introduzido pela primeira vez por Denning [Denning, 1987] em seu trabalho seminal em 1987. Desde então, muitas abordagens foram desenvolvidas. Muitas delas emprestando esquemas, técnicas e metodologias da área de aprendizagem de máquina.

Contudo, o uso da aprendizagem de máquina na detecção de anomalias é diferente de outros domínios de aplicação. A diferença está nas informações. Abordagens de detecção de anomalias precisam lidar com um conjunto de problemas bem conhecidos [Gates e Taylor, 2007]: os detectores tendem a gerar grandes quantidades de falsos positivos; existe uma grande dificuldade em encontrar dados livre de ataque para realizar o treinamento das soluções; e os atacantes podem escapar da detecção ensinando gradualmente ao sistema a aceitar atividades maliciosas como benignas.

Assim, quando comparado com ao resultado operacional de um grande número de pesquisas, a detecção de anomalias em redes não parece ser tão atraente no "mundo real". Os sistemas existentes são geralmente baseados em perfis estatísticos que usam tráfego agregado, como o *Arbor Peakflow*[3]e o *LANScopeStealthWatch*[4], embora operem com um foco muito mais específico quando comparado as generalizações feitas pelas pesquisas acadêmicas.

Desta forma, este minicurso tem por objetivo discutir os diferentes pontos que envolvem o uso da aprendizagem de máquina na detecção de anomalias de redes, incluindo as dificuldades de detecção e uso incorreto da aprendizagem.

2.1.2. Ameaças à Segurança na Internet

Para melhorar a compreensão dos leitores sobre assunto, esta seção apresenta algumas das principais ameaças (ataques e anomalias) à segurança da Internet.

---

[3] http://www.arbornetworks.com/en/peakflow-sp.html

[4] http://www.lancope.com/products/

2.1.2.1. *Cross-Site Scripting* (XSS)

A popularização de serviços pautados no uso de tecnologias *Web* e tendo como principal ferramenta, o *browser*, passou a ser o alvo frequente de diversos agentes que tentam explorar recursos computacionais de forma não autorizada como, capturar *cookies* e sessões de usuários, dados pessoais, número de cartões de crédito, senhas, *logins*, além de outras informações críticas. A demanda cada vez maior por novos recursos para essas aplicações, a fim de prover um número maior de serviços e funcionalidades para seus usuários promoveu um crescimento desordenado e o desalinhamento das políticas de segurança entre servidores e clientes, aumentando assim os riscos de segurança. Dentre eles, destacam-se os ataques de injeção de códigos maliciosos ou *Cross-Site Scripting* (XSS), viabilizados por *scripts* maliciosos injetados em páginas *Web*, no lado cliente ou servidor, de forma que os dados de entrada do usuário não são devidamente validados, permitindo o roubo de informações confidenciais e sessões do usuário, além de comprometer a integridade do sistema em execução. Os códigos *script* são tipicamente desenvolvidos nas linguagens *JavaScript*, *Java* ou *VBSript* e inseridos na estrutura HTML. Tecnologias *Active X*, *Flash* ou quaisquer outros suportados pelo navegador também são usados como vetores de ataques [Grossman et al., 2007].

Notadamente, o vetor mais empregado em ataques de XSS é a linguagem *JavaScript*, que é interpretada pelo *browser*, que aguarda por eventos ou ações, tais como o término de leitura de uma página, ou por ações do usuário, como o movimento do *mouse* sobre um *link* ou o *click* em um objeto, como um botão. Conforme destaca Yue e Wang [Yue e Wang, 2009], 96,9% das páginas *Web* mais populares do mundo fazem uso de *scripts* com base nessa linguagem, denotando a sua popularidade e o seu emprego potencial em ataques do tipo XSS. Isso se deve, principalmente, a inserção e geração insegura de elementos da linguagem *JavaScript* em esquemas potencialmente perigosos, empregando *tags* HTML, atributos e funções *script*. Pode-se citar como exemplo, a geração de código de outros domínios por meio do uso de funções como *document.write* ( ) ou *eval* (), que executa código passado por parâmetro, configurando-se em uma ação potencialmente insegura.

Os ataques são categorizados em XSS Reflexivo ou Não Persistente, Persistente e *DOM-Based*. Nos ataques do tipo Persistente e Reflexivo, o código malicioso está incorporado na página que é retornada para o *browser* cliente. Já no tipo *DOM-Based* o código ao manipular a estrutura DOM (*Document Object Model*), pode eliminar ou criar novos nós dessa estrutura, o que requer a execução ou simulação do preenchimento dos objetos DOM [Saha, 2009] para que esse tipo de ataque possa ser identificado. A figura 2.1 apresenta uma visão geral de um ataque de XSS.

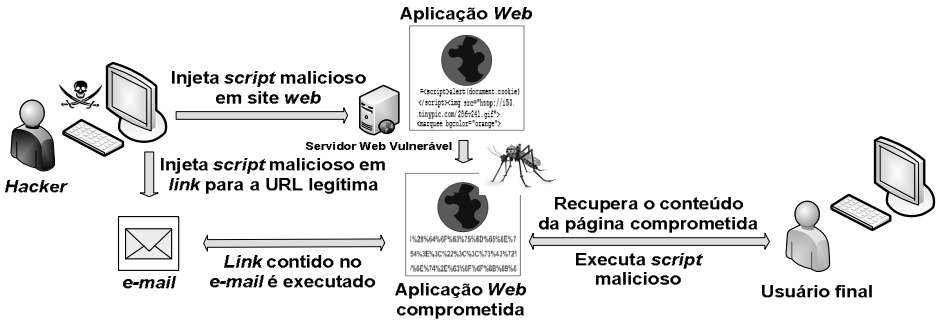

Figura 2.1 apresenta uma visão geral de um ataque de XSS.

Cross-Site Scripting Reflexivo ou Não Persistente

Em ataques não-persistentes, o código malicioso é executado de forma reflexiva no *browser* do cliente. O código não necessita estar armazenado no servidor. O ataque é viabilizado por meio da ação do usuário, que ao receber e executar um *link* contendo código malicioso é direcionado para a página confiável, porém vulnerável a XSS. Ao carregar a página o código é interpretado pelo *browser* e a ação maliciosa é efetivada. Normalmente, os *links* são enviados por *spam* ou por qualquer outro recurso que induza o usuário a visitar uma página por meio do *link* (URL) enviado. Este é o tipo de ataque XSS mais frequente [OWASP, 2010].

Cross-Site Scripting Persistente

Neste tipo de ataque, o código malicioso é permanentemente armazenado em recursos do servidor (banco de dados, sistema de arquivos, entre outros). Aplicações *Web* que permitem ao usuário armazenar dados, tais como fóruns de mensagens, livros de visita, são as mais vulneráveis. Ao acessar uma página contaminada, o usuário poderá executar o código malicioso por meio de uma ação sobre um campo de busca, por exemplo. Este é o tipo mais perigoso de XSS [OWASP, 2010].

Cross-Site Scripting baseado em DOM

Ataques *DOM-Based* são realizados por modificações no ambiente DOM (*Document Object Model*), que é um formato estrutural que pode ser usado para representar documentos no *browser*. O DOM habilita os *scripts* dinâmicos para referenciar componentes do documento, como um campo de texto ou uma sessão de *cookie*, por exemplo [OWASP 2010]. Em ataques deste tipo, funções dinâmicas *JavaScript* são modificadas por códigos maliciosos que realizam mudanças nesse ambiente [Saha, 2009]. Neste caso, o servidor não tem qualquer ingerência, pois o ataque modifica apenas o lado cliente e não envia qualquer código malicioso para o servidor.

Geralmente, para detecção de ataques XSS é necessário aderir às políticas da mesma origem, que são estabelecidas por meio de mecanismos que permitem aos *scripts* acessarem páginas apenas a partir do mesmo domínio, ou ainda, de alterações no código fonte de navegadores atuais. Logo, pesquisadores têm adotado diversos conceitos para identificação e classificação deste tipo de ataque como modelos estatísticos, mineração de dados, teoria da informação e aprendizagem de máquina.

2.1.2.2. *Phishing*

Segundo o *Anti-Phishing Working Group* [APWG, 2010], o *phishing* é uma prática fraudulenta em que criminosos (conhecidos como *phishers*) utilizam tanto engenharia social quanto artifícios técnicos para tentar roubar informações confidenciais das vítimas. Em ataques usando engenharia social, o *phisher* envia mensagens eletrônicas, normalmente e-mails ou mensagens instantâneas, alegando ser uma entidade legítima (bancos, SPC/SERASA, receita federal, entre outros) ou uma pessoa conhecida. O conteúdo dessas mensagens apresenta formulários para o preenchimento e o envio de dados pessoais e financeiros, links para baixar e abrir/executar programas ou links para uma página *Web* construída especificamente para que o destinatário da mensagem forneça dados pessoais e financeiros. Em ataques que utilizam artifícios técnicos, o criminoso instala no computador da vítima software maliciosos, como *trojan*s, *keylogger* e *spyware*s, para roubar dados sigilosos.

A seguir são apresentadas algumas situações envolvendo *phishing*, que vem sendo utilizadas por fraudadores na Internet. Cabe ao leitor deste minicurso observar que há diversas variantes para as situações apresentadas e que, constantemente, surgem novas formas de *phishing*.

- **Mensagens contendo *links* para programas maliciosos**: São mensagens enviadas por *e-mail,* na qual o texto procura atrair a atenção do usuário seja ela por curiosidade, ingenuidade ou na busca de obter vantagem financeira. Geralmente, o texto busca induzir o usuário a clicar em um *link* para baixar ou executar um arquivo, sob a pena de sofrer alguns danos, como por exemplo, a inclusão do seu nome no SPC/SERASA, o cancelamento de um cadastro, da sua conta bancária ou do seu cartão de crédito, etc.

- **Páginas de comércio eletrônico ou *Internet Banking* falsificadas:** Neste tipo de fraude o usuário recebe uma mensagem por e-mail, em nome de um *site* de comércio eletrônico ou de uma instituição financeira. O objetivo é tentar persuadir o usuário a clicar em um *link* contido no texto ou em uma imagem. Geralmente, os textos contidos nestas mensagens envolvem o recadastramento ou confirmação dos dados do usuário, ou a participação em uma nova promoção. Vale ressaltar que, o link direciona o usuário para uma página falsa, semelhante à página legítima, onde serão solicitados dados pessoais e financeiros.

- **Comprometimento do DNS**: Comumente conhecida como *pharming*, neste tipo de fraude, o usuário ao tentar acessar um *site* de comércio eletrônico ou *Internet Banking*, mesmo digitando o endereço corretamente e diretamente no seu *browser*, será redirecionado para uma página falsificada, semelhante ao *site* verdadeiro. Geralmente, a causa deste ataque é decorrente do comprometimento do DNS pelo atacante, de modo que todos os acessos passaram a ser redirecionados para páginas falsificadas. O atacante pode ter induzido o usuário a instalar, em algum momento, um *malware* específico para alterar o DNS.

- **Utilização de computadores de terceiros**: É um dos principais vetores para disseminação de fraudes do tipo *phishing*, pois geralmente o usuário utiliza uma *LAN house* ou *cybercaf*e, para acessar um site de comércio eletrônico ou *Internet Banking*. O risco se deve à utilização destes computadores por diversas pessoas,

que em algum momento, foram induzidas a instalar um código malicioso, tendo, portanto todas as ações monitoradas. A partir daí, qualquer usuário que necessitar utilizar estes computadores para realizar transações eletrônicas poderão ter seus dados comprometidos.

2.1.2.3. Negação de Serviço

Diferente de ataques como XSS e *phishing*, um ataque de negação de serviço (do Inglês *Denial-of-Service* ou DoS) consiste na tentativa de impedir usuários legítimos de utilizarem um determinado serviço de um computador ou rede, ou seja, tornar indisponíveis recursos ou serviços oferecidos por um servidor, sistema ou rede [Kumar e Selvakumar, 2011].

Atualmente, ataques de negação de serviço são executados de forma distribuída (DDoS – *Distributed Denial of Service*), onde o atacante busca potencializar e dimensionar o ataque. Desta forma, os ataques DDoS não são baseados no uso de um único computador, mas sim de centenas ou até milhares de computadores previamente comprometidos e ligados na Internet para lançar um ataque coordenado a determinada vítimas ou vítimas. Um ataque DDoS utiliza uma espécie de arquitetura (classe) social composta por: (i) **vítima**, alvo do ataque; (ii) **zumbi**, máquina que efetivamente concretiza o ataque DoS contra uma ou mais vítimas; (iii) **mestre**, computador intermediário localizado entre o atacante e os zumbis, que recebe os parâmetros (ordens) para o ataque. Cada mestre controla certo número (centenas ou milhares) de zumbis; (iv) **atacante**, responsável por coordenar o ataque.

A Figura 2.2 exemplifica a estrutura de um ataque de negação de serviço distribuído (DDoS).

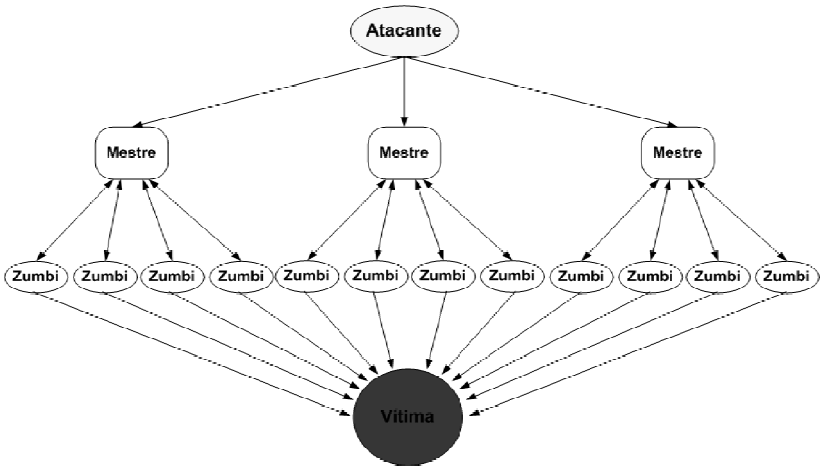

Figura 2.2. Estrutura de um ataque de negação de serviço distribuído.

Em [Feitosa et al., 2008] é encontrada uma descrição mais detalhada dos principais tipos de ataque de negação de serviço, incluindo inundação (*flooding*), ataques ao protocolo TCP, ataques por refletores, *backscatter*, entre outros.

2.1.2.4 Códigos Maliciosos (*Malware*)

A disseminação de código malicioso através da Internet tem sido a maior ameaça atual à segurança de sistemas de informação. Código Malicioso ou *Malware* é um termo

genérico que abrange todos os tipos de programas desenvolvidos especificamente para executar ações maliciosas em um computador. Esta denominação é comumente empregada a um conjunto de aplicações também denominadas individualmente como vírus, *worms*, *keyloggers*, *trojans*, *backdoors* e outros tipos de programas com a intenção de comprometer um sistema.

De acordo com [Feitosa et al., 2008], os principais exemplares *malware* podem ser descritos conforme a tabela 2.1.

Tabela 2.1. Principais exemplares de *malware.*

Tipo	Descrição
Vírus	São programas que modificam a operação normal de um computador, sem permissão e conhecimento do usuário. Assim como um vírus biológico, um vírus de computador se replica e se espalha introduzindo cópias suas em outros códigos ou programas executáveis.
Worms	São programas auto-replicantes capazes de se auto-propagar através da rede explorando principalmente falhas de segurança em serviços. A taxa de propagação dos *worms* é muito rápida e pode ameaçar a infraestrutura da Internet uma vez que cada máquina infectada torna-se um potencial atacante. De modo geral, prejudicam a rede consumindo largura de banda. A ativação de um *worm* pode ser tão rápida quanto sua velocidade de propagação.
Cavalo de Tróia (*Trojan Horse*)	São programas que, uma vez ativados, executam funções escondidas e não desejadas como, por exemplo, varreduras de endereços IP e porta, envio de grande volume de *spam*, ataques DDoS ou até mesmo adicionar o computador em uma *botnet*. Diferente dos *worms*, um cavalo de tróia não se auto-propaga, depende da interferência e curiosidade humana para se propagar.
Spyware	São programas espiões que automaticamente recolhem informações sobre o usuário e os transmite para seus "instaladores". Geralmente, os dados monitorados referem-se aos habítos de compras na Internet ou a informações confidenciais como contas bancárias e senhas pessoais. Além dos próprios *spywares*, na categoria de programas espiões também estão o *adware* e o *keylogger*.

Percebe-se que a crescente interação de dispositivos diversos, juntamente com a popularização da Internet, aliada à vasta provisão de serviços e falta de conhecimento adequado por parte dos usuários contribui para o cenário atual de ataques por meio de *malware*. A venda de informações sensíveis, tais como contas e senhas bancárias e números de cartões de créditos são grandes motivadores para ocorrência desses ataques.

Os sistemas computacionais vulneráveis são facilmente explorados por atacantes que utilizam códigos maliciosos. Há, portanto, diversas técnicas para que um código malicioso seja executado nos sistemas, tendo as mais comuns baseadas em engenharia social, exploração remota de serviços e injeção de código [Ceron et al, 2009]. Quando o sistema é comprometido, o mesmo fica sob o domínio do atacante, podendo realizar as mais diversas ações fraudulentas, tais como roubo de dados sigilosos, envio de *spam*, e *emails phishing*.

Uma das maneiras mais utilizadas para o comprometimento de sistemas é através da propagação autonômica de *malwares.* A busca por máquinas vulneráveis com o intuito de explorar vulnerabilidades e comprometer o sistema é característica de *malwares* como *worms* e *bots.* Uma forma de entender as características dos diferentes tipos de *malware* é realizar a análise de tráfego malicioso.

O principal meio de defesa contra códigos maliciosos é através de softwares antivírus, pois se baseiam em um banco de dados contendo diversas assinaturas para caracterizar um *malware* conhecido. Porém, quando um *malware* não possui uma assinatura é necessário analisá-lo e entender como o seu comportamento afeta o sistema, para então desenvolver sua assinatura, e posteriormente, conhecer suas funcionalidades para que seja removido do sistema.

2.1.2.5. *Spam*

O problema de e-mail não solicitado ou *spam* é bem conhecido pela maioria dos usuários Internet. O *spam* causa a perda de tempo dos usuários e a sobrecarga dos recursos computacionais, ocasionando assim grandes perdas financeiras.

De acordo com [Feitosa et al., 2008], o *spam* pode ser classificado conforme a Tabela 2.2.

Tabela 2.2. Classificações de *Spam*.

Tipo	Descrição
Boatos (*hoaxes*)	São mensagens que tentam impressionar os usuários através de estórias falsas e assim garantir sua divulgação. Alguns exemplos deste tipo de mensagens incluem o roubo de rins, desaparecimento de crianças, difamação de empresas, etc.
Correntes (*chainletters*)	São mensagens que prometem algum de tipo de lucro financeiro ao leitor, caso seja feito o repasse destas mensagens a um determinado número de usuários.
Propagandas ("*pumpanddump*")	São as mensagens mais comuns e mais divulgadas na Internet, sendo representadas por mensagens eletrônicas não solicitadas com oferta de produtos ou artigos baratos com links para companhias pequenas ou inexistentes.
Golpes (*scam*)	São mensagens enganosas que afirmam que o leitor foi "agraciado" com algum produto. Exemplos corriqueiros incluem sorteios, oferta de empregos, abertura do próprio negócio, etc.
Estelionato (*phishing*)	São mensagens usadas para obter dados pessoais (tais como números de contas bancárias, cartão de crédito, senhas, entre outros) de destinatários. Normalmente, induzem o leitor a acessar a URL indicada na mensagem, clicar na imagem de uma empresa ou preencher algum tipo de formulário.
Malware	São mensagens enviadas ao leitor contendo códigos maliciosos, com a finalidade de enganar o leitor, levando-o a executar um determinado programa enviado junto a mensagem. O resultado é a instalação de vírus, *worms* e cavalos de tróia, sempre visando alguma tentativa de fraude ou ataques de negação de serviço.

2.1.3. Soluções Existentes

A Internet é uma das poucas plataformas operacionais existentes que não possui centros de controle. Tal característica serve tanto como fator de sucesso quanto serve como ponto o de falhas, o que explica o aumento do tráfego malicioso. Na tentativa de lidar com esse lado negativo da Internet, várias soluções têm sido desenvolvidos ao longo dos anos. Esta seção apresenta algumas dessas soluções que visam tornar o ambiente de redes mundiais de computadores um ambiente mais seguro e menos hostil em relação às ameaças a segurança da Internet apresentadas na seção anterior.

2.1.3.1. Filtragem

Entre as soluções mais usadas atualmente na segurança da Internet, sem dúvida nenhuma os esquemas de filtragem ou *firewall* são principais. Basicamente, *firewalls* utilizam regras (filtros) que definem o que deve ser feito. Desta forma, todo o tráfego

que entra ou sai da rede ou máquina é comparado com as regras e o resultado é uma ação, geralmente permitir ou negar o tráfego.

Embora não exista um consenso sobre a classificação dos tipos de *firewall*, a mais usual considera a existência de três: filtro de pacotes, filtro de estados e filtros de aplicação (*gateways*). O primeiro tipo, filtro de pacote, compara as informações do cabeçalho de cada pacote (endereços IP, portas e protocolo) com as regras definidas para decidir qual ação tomar. A segunda, filtros de estado, mantêm registros do estado das conexões de rede (TCP e UDP) que estão ativas. Assim, a filtragem é feita baseada na tabela de estados de conexões estabelecidas e não apenas no cabeçalho. O último tipo, *firewall* de aplicação, opera na camada de aplicação, vasculhando o conteúdo dos pacotes a procura de indícios de anomalias como, por exemplo, sequências de caracteres específicos (palavras ou frases) que indicam a presença de ataques, código maliciosos e até mesmo de determinadas aplicações como SMTP (*Simple Mail Transfer Protocol*), FTP (*File Transfer Protocolo*), HTTP (*Hyper Text Transfer Protocol*), P2P, entre outros.

Entretanto, a efetividade das soluções de filtragem não tem sido suficiente para limitar o tráfego malicioso. Em primeiro lugar, embora prática, a filtragem é imperfeita, porque depende de heurísticas e configurações manuais, o que por muitas vezes acaba filtrando informações legítimas. Além disso, a filtragem mais eficiente é destinada a aplicações específicas. Mecanismos de filtragem como *proxies* e *gateways* de aplicação são desenvolvidos para avaliar tráfego específico de determinadas aplicações. Desta forma, para cada novo serviço ou aplicação que se desejar filtrar, uma solução específica será necessária.

2.1.3.2. Sistemas de Detecção de Intrusão

A detecção de intrusão refere-se à descoberta de atividades maliciosas (intrusões, penetrações e outras formas de abuso de computadores) dentro de um sistema de computadores relacionados [Phoha, 2002]. A detecção de intrusão pode ser classificada em duas categorias [Anderson, 1995] [Rhodes et al., 2000]: detecção de abuso (*misuse detection*) e detecção de anomalias (*anomaly detection*). Na primeira categoria, as soluções utilizam bases de assinaturas, assim os ataques conhecidos são detectados com bastante rapidez e com baixa taxa erro. Por outro lado, a principal limitação dessas ferramentas é que elas não podem detectar novas formas de códigos maliciosos que não sejam compatíveis com as assinaturas existentes. A segunda categoria, detecção de anomalias, é baseada na construção de perfis de comportamento para padrões considerados como atividade normal. Desvios da normalidade são então tratados como ameaças. Entretanto, é difícil saber o que procurar quando atividades não autorizadas sob um sistema assumem diferentes formas ou mesmo imitam atividades legítimas. Na tentativa de evitar que atividades com potencial malicioso sejam autorizadas, muitos sistemas emitem uma taxa elevada de falsos alarmes, reduzindo substancialmente sua efetividade.

A Figura 2.3 ilustra uma visão genérica da organização de um sistema de detecção de intrusão. O administrador recebe relatórios do servidor de alarmes que também envia relatórios para o sistema monitorado. O administrador conta com um servidor de modelos de intrusão que auxilia com o reconhecimento de uma intrusão.

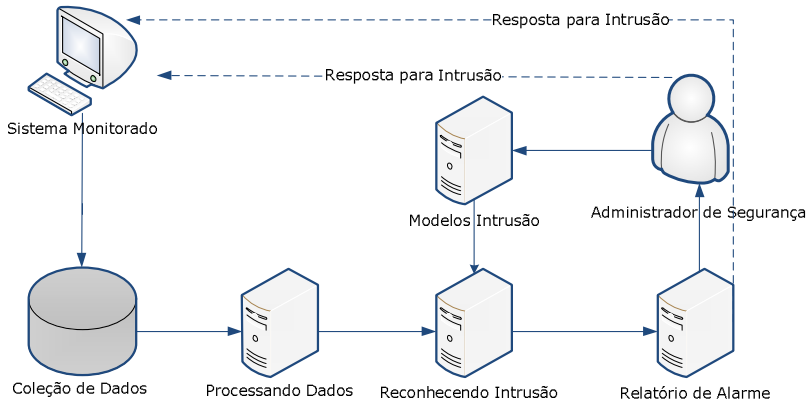

Figura 2.3. Organização genérica de um Sistema de Detecção de Intrusão

O principal problema dos IDS é a precisão. Como existe uma constante mutação no tráfego de rede e nos padrões de assinatura, a detecção de intrusão se torna cada vez mais difícil e sujeita a erros.

2.1.3.3. Software anti-*

A terceira e última categoria de soluções são os software anti-*, programas desenvolvidos para detectar e eliminar potenciais ameaças aos computadores e redes como, por exemplo, antivírus, anti-*spyware*, anti-*phishing* e anti-spam. Antivírus são programas que detectam e removem vírus de computador. Uma vez que novos vírus e variantes de vírus conhecidos são "lançados" quase que diariamente, um *software* antivírus deve manter-se automaticamente ou ser mantido sempre atualizado. Já os anti-*spywares* são usados no combate a programas e códigos espiões como *spyware*, *adware*, *keyloggers*. Softwares anti-*phishing* visam bloquear possíveis tentativas de fraude através de sites *Web* ou mensagens de correio eletrônico. Normalmente, este tipo de solução é apresentado na forma de barras de tarefas (*toolbar*) integradas com navegadores *Web* ou clientes de correio eletrônico.

As soluções anti-*spam* também se enquadram nesta categoria. Basicamente, a detecção de *spam* é baseada na filtragem de mensagens não solicitadas através dos campos do cabeçalho ou do conteúdo da mensagem. A filtragem de cabeçalho verifica o endereço de origem, nome do remetente e assunto de uma mensagem para validá-la ou não. Este tipo de solução anti-*spam* é mais simples e sujeita a erros de configuração, uma vez que é preciso definir regras do que se quer ou não receber, ou seja, quais endereços, remetentes e assuntos são indesejados. Listas negras (*blacklist*), listas brancas (*whitelist*) e listas cinza (*greylist*) são exemplos de filtragem de cabeçalho. A filtragem baseada no conteúdo da mensagem é a mais utilizada. Normalmente esta técnica realiza buscas por palavras chaves tais como "Viagra" no conteúdo das mensagens. Quando configurados corretamente, a filtragem baseada em conteúdo é bem eficiente, mas também podem cometer erros (barrar "especialista" porque contém "cialis", por exemplo). Além disso, a inspeção de conteúdo não verifica a origem da mensagem. Atualmente, o uso de filtros com mecanismo de aprendizagem de máquina tem sido muito utilizado.

2.1.4. Organização do Capítulo

Diante do exposto, torna-se claro a necessidade de desenvolver novas soluções que contribuam para a diminuição das atividades maliciosas na Internet. Este minicurso tenta fornecer uma visão estruturada e abrangente da pesquisa sobre detecção de anomalias que utiliza técnicas e algoritmos de aprendizagem de máquina, com o objetivo de facilitar a compreensão das diferentes direções em que a pesquisa nesta área tem evoluído e como as técnicas desenvolvidas podem ser aplicadas na prevenção e detecção de atividades maliciosas.

O restante deste capítulo está organizado da seguinte forma. A Seção 2.2 apresenta de forma clara e didática as técnicas de aprendizagem de máquina. Para tanto, os conceitos relacionados à aprendizagem de máquina são explicados, objetivando definir formalmente aprendizagem de máquina e os contextos dos problemas em que pode ser utilizada com sucesso. Também serão apresentadas definições e terminologias importantes como classes, características, parâmetros, entre outras. Além disso, as três principais categorias dos métodos de aprendizagem (supervisionada, não supervisionada e por reforço) são apresentadas. Por fim, são descritas as principais técnicas de aprendizagem de máquina tais como SVM (*Support Vector Machines*), *k*-NN (*k-Nearest Neighbors*), RNAs (Redes Neurais Artificiais), *k-Means*, *NaiveBayes*, conjuntos de classificadores, entre outras, aplicáveis na área de detecção de anomalias na Internet.

A Seção 2.3 discute o emprego de técnicas de aprendizagem de máquina em detecção de anomalias. A primeira seção deste tópico descreve uma série de desafios de uso e, orientações indicando quando e como usar aprendizagem de máquina na detecção de anomalias de forma apropriada. Em seguida, as técnicas de aprendizagem de máquina descritas na seção anterior são analisadas no contexto de aplicações de segurança, onde exemplos de aplicações práticas bem sucedidas de cada técnica são apresentados. Os principais problemas abordados são: *Cross-Site Scripting* (XSS), *phishing*, *malware*, negação de serviço e *spam*. Por fim, é apresentada uma síntese dessas aplicações, sumarizando questões como aplicabilidade, vantagens e desvantagens.

Por último, a Seção 2.4 apresenta objetos de pesquisa que devem crescer em relevância nos próximos anos, destacando o emprego de diferentes técnicas de aprendizagem de máquina na detecção de anomalias na Internet. São mencionadas as questões de pesquisa em aberto com o propósito de indicar novas oportunidades. Por fim, são apresentados os comentários finais sobre o tema.

2.2. Aprendizagem de Máquina

Entidades inteligentes destacam-se pela capacidade de adequar-se a novos ambientes e de resolver novos problemas. Um computador pode ser orientado a interpretar as informações recebidas de uma forma que melhore gradualmente seu desempenho [Rich e Knight, 1991]. Essa é a base na qual a área de pesquisa em aprendizagem de máquina está fundamentada.

2.2.1. Definição

Existem inúmeras atividades associadas à noção de aprendizagem, dificultando a definição exata do termo e tornando-o dependente de contexto. Porém, no contexto de

aplicações computacionais, uma definição muito precisa de aprendizagem de máquina pode ser encontrada em [Alpaydin, 2010]:

> *"... são programas de computador utilizados para otimizar um critério de desempenho, usando dados de exemplo ou experiência do passado".*

Independente da definição exata de aprendizagem é amplamente aceito na literatura que sistemas de aprendizagem confiáveis são de importância estratégica em diversas áreas de aplicação, pois há muitas tarefas que não podem ser solucionadas através do uso de técnicas de programação clássicas. Podemos citar como exemplo, a classificação de e-mails em duas classes: legítimos e s*pam*. Nesse tipo de problema a entrada é conhecida: um documento e-mail que em seu caso mais simples é um arquivo texto; e a saída também é conhecida: *spam* ou não *spam*. Porém, a forma como a entrada deverá ser convertida na saída é desconhecida.

Os algoritmos de aprendizagem de máquina, portanto, podem ser a chave para solucionar problemas dessa natureza, pois são algoritmos que podem "aprender" a definir padrões das classes envolvidas no problema, a partir de exemplos reais obtidos do ambiente. A Figura 2.4 mostra um modelo genérico de aprendizagem de máquina. O ambiente fornece informação para um elemento de aprendizagem que usa essa informação para fazer melhoramentos em uma base de conhecimento e então, um elemento de desempenho usa essa base para executar sua tarefa.

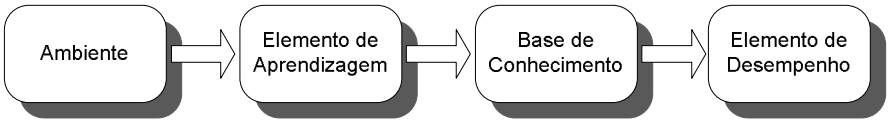

Figura 2.4. Modelo genérico de aprendizagem de máquina.

2.2.2. Categorização

Aprendizagem de máquina pode ser dividida em dois paradigmas fundamentais: **aprendizagem com professor** e **aprendizagem sem professor** [Haykin, 2008]. No primeiro, normalmente chamado **aprendizagem supervisionada**, o sistema precisa conhecer o ambiente. Esse conhecimento é representado por um conjunto de exemplos de pares de entrada-saída que são transmitidos em uma sequencia de instruções que o computador seguirá para alcançar o efeito desejado. Na aprendizagem sem professor, não há exemplos rotulados da função a ser aprendida. Nesse paradigma são identificadas duas subdivisões:

- **Aprendizagem de reforço**: aprendizagem a partir de um mapeamento de entrada-saída alcançada através de interação continuada com o ambiente para minimizar um índice escalar de desempenho [Haykin, 2008].
- **Aprendizagem não-supervisionada**: não há valores de saída desejada. A tarefa de aprendizagem envolve a obtenção de alguma compreensão do processo que gerou os dados e desenvolver habilidades para formar representações internas capazes de codificar características da entrada e, portanto, criar novas classes automaticamente [Haykin, 2008]. Esse tipo de aprendizagem inclui estimação de densidade, formação de agrupamentos, dentre outros.

2.2.3. Taxonomia

Para uma compreensão mais abrangente dos algoritmos de aprendizagem de máquina, é necessário que alguns dos termos e conceitos relevantes sejam definidos. As definições apresentadas a seguir, foram extraídas de [Kuncheva, 2004]:

1. Classes

Uma classe possui objetos similares, enquanto objetos de classes diferentes são dissimilares. Por exemplo, sites *phishing* podem ser categorizados como objetos da classe *phishing*, enquanto sites não *phishing* são objetos da classe *sites* legítimos. Algumas classes são facilmente definidas, por exemplo, um e-mail deverá obrigatoriamente ser *spam* ou não *spam.* Outras classes, porém, são de difícil definição, como por exemplo, a classe das pessoas destras e das pessoas não destras.

Resumidamente, pode-se definir que um problema de classificação de dados possui *c* classes, rotuladas de ω_1 a ω_c, organizadas em um conjunto de rótulos $\Omega = \{\omega_1,..,\omega_c\}$e que cada objeto pertence a apenas uma classe.

2. Características ou atributos

Os objetos que compõem as classes são descritos através de características ou atributos. As características podem ser nominais, tais como endereço e profissão, e podem ser numéricas, como tamanho da URL e quantidade de palavras-chaves. Os valores de características de um objeto *x* são organizados em um vetor *n*-dimensional $x = [x_1,...,x_n] \in \Re^n$. O espaço $\Re^n$ é chamado espaço de características, sendo que cada eixo corresponde a uma característica do objeto.

3. Base de Dados

A informação fundamental para algoritmos de aprendizagem de máquina, independentemente do tipo de aprendizagem, é proveniente dos dados disponíveis do problema. Essa informação normalmente está organizada na forma de um conjunto de dados $Z = \{z_1,...,z_N\}, z_j \in \Re^n$. Em problemas de aprendizagem supervisionada, o rótulo da classe de z_j é definido por $l(z_j) \in \Omega,\ j = 1,...,N$. Para construir uma base de dados, as instâncias do problema devem ser transformadas em vetores de características. Nem sempre essa tarefa é simples. Por exemplo, dadas diversas instâncias de sites com código *script*, não é uma tarefa trivial representar as páginas através de vetores de características como: tamanho de URL, existência de código ofuscado, etc. Entretanto, o uso de características relevantes para representar as instâncias do problema a ser tratado é fundamental para o sucesso do processo de classificação automática.

2.2.4. Técnicas de Aprendizagem de Máquina

São descritos nesta seção cinco (5) métodos de aprendizagem supervisionada: *k*NN, SVM, *Naive Bayes*, Redes Neurais Artificiais e Árvores de Decisão; um (1) método de aprendizagem não supervisionada: *k-means*; e métodos que utilizam a combinação de classificadores.

2.2.4.1. KNN (*K-Nearest Neighbor*)

KNN é uma técnica de classificação baseada em instâncias, isto é, consiste em atribuir a classe de cada elemento desconhecido (amostras de teste) a partir da classe majoritária obtida entre os seus vizinhos mais próximos identificados no conjunto de treinamento. A definição de vizinhança é feita segundo uma medida de similaridade que normalmente é uma medida de distância calculada no espaço de características. A distância Euclidiana é uma das medidas de similaridade mais utilizada na literatura [Theodoridis e Koutroumbas, 2006]. Essa medida é calculada pela seguinte fórmula,

$$d(x,y) = \sqrt{\sum_{i}^{n} (x_i - y_i)^2} \qquad (1)$$

Onde x é um exemplo de treino, representado por $x = [x_1,...,x_n] \in \Re^n$ sendo n a dimensão do vetor de características, enquanto y representa a amostra de teste, sendo $y = [y_1,...,y_n] \in \Re^n$.

O algoritmo KNN é um dos mais simples dentre as técnicas de aprendizagem de máquina. O funcionamento desse algoritmo é descrito abaixo, de acordo com [Theodoridis e Koutroumbas, 2006]:

1. Defina um valor para *k*, ou seja, a quantidade de vizinhos mais próximos;
2. Calcule a distância da nova amostra a ser classificada a todas as amostras de treinamento;
3. Identifique os *k* vizinhos mais próximos, independentemente do rótulo das classes;
4. Conte o número de vizinhos mais próximos que pertencem a cada classe do problema;
5. Classifique a nova amostra atribuindo-lhe a classe mais frequente na vizinhança.

Este processo de classificação pode ser computacionalmente exaustivo quando o conjunto de treinamento possui muitos dados. Por isso, a grande desvantagem de KNN é a complexidade computacional envolvida na obtenção dos *k* vizinhos mais próximos. Por outro lado, KNN tem como uma de suas vantagens a independência quanto à distribuição de dados no espaço de características.

2.2.4.2. SVM (*Support Vector Machines*)

É uma técnica de classificação amplamente aplicada em problemas de segurança de redes tais como detecção de *phishing* [Miyamoto et al. 2009] e detecção de intrusos [Xiao *et al.,* 2007]. Basicamente, o funcionamento de SVM pode ser descrito da seguinte forma: dadas duas classes e um conjunto de instâncias de treinamento cujas amostras pertencem a essas classes, SVM constrói um hiperplano que divide o espaço de características em duas regiões, maximizando a margem de separação entre as mesmas. Esse hiperplano é conhecido como hiperplano de separação ótima. As amostras desconhecidas (exemplos de teste) são então mapeadas para esse mesmo espaço, e atribuídas a uma das classes [Alpaydim, 2010].

A Figura 2.5 mostra o hiperplano de separação ótima (reta separadora) para um problema bidimensional típico e linearmente separável. As retas pontilhadas H_1 e H_2, paralelas ao hiperplano, constituem o par de hiperplanos que geram a margem máxima pela minimização do vetor peso *w*. Além disso, $|b|/||w||$ é a distância perpendicular do hiperplano à origem e $||w||$ é a norma Euclidiana de *w*. Os pontos que estão em um dos hiperplanos H_1 e H_2 são chamados vetores de suporte. Esses pontos, indicados na Figura 2.5 por círculos extras, alteram a solução encontrada caso sejam removidos. Além disso, em fase de uso de SVM, apenas os vetores de suporte são necessários para que dados desconhecidos sejam classificados.

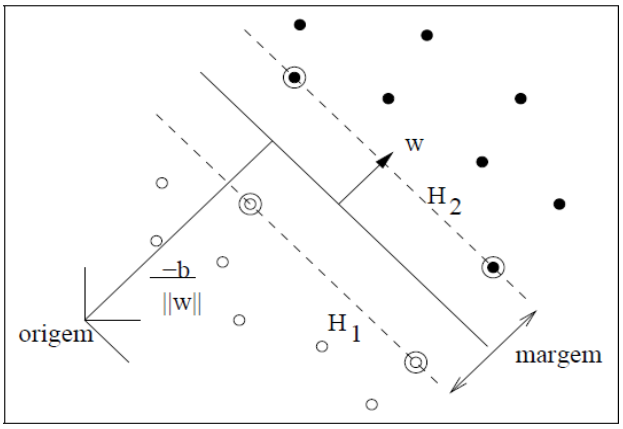

Figura 2.5. Hiperplano de separação ótima para um problema com duas classes.

O algoritmo original de SVM não encontra a solução desejada quando aplicado a dados não linearmente separáveis, característica presente na maioria dos problemas reais [Alpaydim, 2010]. Entretanto, tais problemas podem ser solucionados por SVM através da utilização de funções de separação de dados mais complexas do que funções lineares. Sendo assim, o uso de diferentes funções *kernel* possibilita a construção de SVM com diferentes tipos de superfícies de decisão não-linear no espaço de entrada.

Com o uso de funções *kernel,* as instâncias são inicialmente mapeadas para um espaço de características de maior dimensão que o espaço de características original. Permitindo dessa forma, a classificação em espaços não linearmente separáveis. A Figura 2.6 mostra o processo de transformação de um domínio não linearmente separável, em um problema linearmente separável através do aumento da dimensão, consequência do mapeamento feito por uma função *kernel* $F(x)$.

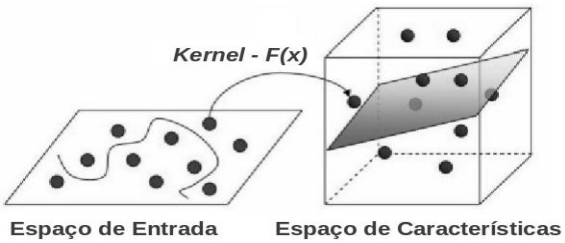

Figura 2.6. Mapeamento do espaço de entrada via função *kernel*.

Dentre as funções *kernel* mais usadas destacam-se: Polinômios, Funções de Base Radial (RBF) ou Gaussiana e Sigmóide, definindo diferentes máquinas de

aprendizagem conforme Tabela 2.3. Nessa tabela, *x* representa os vetores de suporte, enquanto *y* representa os dados de teste.

Tabela 2.3. Funções de *Kernel* mais utilizadas com SVM

Tipo de *Kernel*	Função $K(x,x_i)$	Tipo de Classificador
Polinomial	$[(x*y)+1]^d$	Máquina de Aprendizagem Polinomial
Gaussiano ou (RBF)	$\exp\left(-\frac{(x-y)^2}{2\sigma^2}\right)$	Rede RBF
Sigmoidal	$\tanh[\beta_0(x*y)]+\beta_1$	Perceptron de duas camadas

A principal vantagem de SVM é a baixa probabilidade de erros de generalização [Vapnik, 1995]. Quanto à utilização de SVM para detecção de intrusão, por exemplo, há duas vantagens principais. A primeira está relacionada à rapidez do algoritmo em fase de uso, uma vez que o desempenho em tempo real é de primordial importância para esse tipo de aplicação. A segunda razão é a escalabilidade, pois SVM é relativamente insensível ao número de pontos de dados. Dessa forma, a taxa de precisão na classificação não depende diretamente da dimensão do espaço de características [Mukammala *et al*., 2002]. Entretanto, SVM tem como desvantagem a demanda por elevada complexidade computacional na fase de treinamento, fato que pode inviabilizar o uso de SVM em problemas que necessitem de treinamento online.

2.2.4.3. *Naive Bayes*

Esse método de aprendizagem de máquina baseia-se em fundamentações simples, mas frequentemente produz elevada taxa de classificação em problemas reais. O modelo busca uma resposta para questionamentos como: "Qual a probabilidade do ataque ser de um determinado tipo, dado alguns eventos do sistema observado?" [Tsai et al., 2009].

Considerando uma instância $x = \{x_1, x_2, \dots, x_n\} \in \Re^n$, o problema de classificação tratado por *Naive Bayes* consiste em atribuir à amostra *x* uma das *c* classes envolvidas no problema, em que $\Omega = \{\omega_1, .., \omega_c\}$ é o conjunto de classes. Um erro mínimo de classificação pode ser obtido quando a classe com maior probabilidade *a posteriori* $P(\omega_i|x)$ é atribuída à *x.* Essa probabilidade é calculada através da fórmula de *Bayes* definida na Equação 2.

$$P(\omega_i|x) = \frac{P(\omega_i)p(x|\omega_i)}{\sum_{j=1}^{c} P(\omega_j)\, p(x|\omega_j)}, i = 1, \dots, c \qquad (2)$$

em que $P(\omega_i)$ é a probabilidade *a priori* de ocorrência da classe ω_i e $p(x|\omega_i)$ é a função de densidade de probabilidade condicional de cada classe (PDF– *Probability Distribution Function*).

A obtenção de uma estimação precisa das PDFs conjuntas é difícil, especialmente se a dimensão do espaço de características $\Re^n$ é elevada. Para superar essa dificuldade, *Naive Bayes* assume que as características são condicionalmente independentes. Neste caso, a PDF conjunta para uma classe dada é obtida da seguinte forma:

$$p(x|\omega_i) = \prod_{j=1}^{n} p(x_j|\omega_i), \qquad i = 1,..,c \tag{3}$$

Precisamente, a PDF conjunta é obtida através da verificação da frequência decorrência de cada valor de atributo para cada classe do problema nas amostras de treinamento. Além disso, a probabilidade *a priori* também pode ser obtida através da frequência de ocorrência de cada classe na base de treinamento. A hipótese de independência entre características pode parecer restritiva, mas resultados práticos encontrados na literatura em diversas áreas de aplicação mostram que *Naive Bayes* produz elevada taxa de classificação mesmo quando as características são claramente dependentes [Kuncheva e Hoare, 2008].

Baixa complexidade na fase de treinamento é uma das principais vantagens de *Naive Bayes*, uma vez que essa fase envolve apenas o cálculo de frequências para que as probabilidades sejam obtidas. Essa característica torna *Naive Bayes* indicado para aplicações online, tais como problemas envolvendo segurança de redes cujo treinamento precisa ocorrer de forma online e com frequência regular. Outra característica que torna *Naive Bayes* atrativo para a área de segurança de redes é o fato de possibilitar a manipulação de atributos nominais e numéricos. Atributos nominais são frequentes em detecção de *spam*, *cross-site scripiting*, páginas *phishing*, dentre outros problemas de classificação de documentos textuais.

Naive Bayes tem sido usado para detecção de anomalia em definições de múltiplas classes [Chandola et al., 2009]. Diversas variantes da técnica básica foram propostas para detecção de intrusão em redes de computadores através de detecção em vídeo de vigilância, detecção de anomalia em dados de texto e para detecção de surto de doenças [Chandola et al., 2009].

2.2.4.4. Redes Neurais Artificiais

Redes Neurais Artificiais ou simplesmente redes neurais (ou neuronais) são inspiradas no sistema nervoso biológico. Tal inspiração deve-se à seguinte razão: o cérebro é um dispositivo de processamento de informação com inúmeras habilidades, como por exemplo, visão, reconhecimento de voz, etc. [Haykin, 2008]. Com as redes neurais busca-se, portanto, definir soluções algorítmicas para os mesmos problemas solucionados pelo cérebro.

Inicialmente, rede neural foi proposta com a expectativa de ser um método computacional massivamente paralelo, tal qual o cérebro humano. Porém, com a evolução das pesquisas, esse objetivo tornou-se remoto. Por outro lado, a técnica baseada em rede neural é atualmente uma ferramenta de aprendizado de máquina importante, na teoria e na prática, em muitas áreas de aplicação. Pode-se dizer que as redes neurais artificiais assemelham-se ao cérebro humano em dois (2) aspectos:

1. O conhecimento é adquirido pela rede através de um processo de aprendizado;
2. Interconexões entre os neurônios são usadas para armazenar o conhecimento.

Diversos protocolos e algoritmos de treinamento foram e ainda são desenvolvidos, fator que é a chave do sucesso das redes neurais [Kuncheva, 2004].

Alguns dos principais tipos de redes são: Função de Base Radial (RBF); redes de Hoppfield; *Adaptive Resonant Theory* (ART); Redes de Kohonen (*Self Organizing Map* - SOM); *Perceptron* de uma única camada e Perceptron Multi-Camadas (MLP). Nesta seção, a rede MLP é descrita sucintamente. Trata-se, provavelmente, do tipo de rede neural mais utilizado [Kuncheva, 2004]. Antes, porém, é necessário descrevermos o componente fundamental da rede, neurônio artificial.

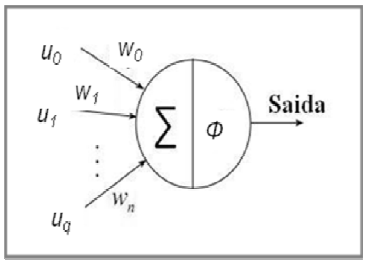

Figura 2.7.Esquema básico do processamento de um neurônio

Os neurônios são as unidades de processamento no cérebro humano. Da mesma forma, os neurônios artificiais, também chamados de nós, são as unidades de processamento de uma rede neural. Considerando $u = [u_0, \dots, u_q]^T \in \Re^{q+1}$ o vetor de entrada do neurônio (os vetores de entrada iniciais são as amostras do conjunto de treinamento) e $v \in \Re$ sua saída, o vetor $w = [w_0, \dots, w_q]^T \in \Re^{q+1}$ é chamado vetor de pesos sinápticos. Conforme pode ser observado na Figura 2.7, o neurônio executa um processamento que ocorre através do cálculo da seguinte função:

$$v = \phi(\xi); \qquad \xi = \sum_{i=0}^{q} w_i \, u_i \qquad (4)$$

em que $\phi: \Re \to \Re$ é a função de ativação e ξ é o somatório da rede. As escolhas mais comuns de ξ são:

Função Limiar: $\phi(\xi) = \begin{cases} 1, & se\ \xi \geq 0, \\ 0, & caso\ contrário \end{cases}$

Função Sigmóide: $\phi(\xi) = \frac{1}{1+\exp(-\xi)}$

Função identidade: $\phi(\xi) = \xi$

O peso w_0 é usado como *bias* (tendência), e sua entrada correspondente u_ié normalmente definido como 1. Com essa nova informação, a equação 4 pode ser reescrita como:

$$v = \phi\left[\sum_{i=1}^{q} w_i u_i - (-w_0)\right] \qquad (5)$$

Geometricamente, a equação 6 define um hiperplano em $\Re^q$.

$$\sum_{i=1}^{q} w_i u_i - (-w_0) = 0 \qquad (6)$$

Considerando um problema de classificação com apenas duas classes envolvidas, um neurônio com função de ativação limiar, por exemplo, atribui o valor +1 para as amostras que se localizam de um dos lados do hiperplano, enquanto as demais amostras recebem valor 0.

Rede Perceptron Multicamadas (MLP)

É uma rede composta por três grupos de camadas: (1) camada de entrada - recebem as amostras de dados na forma de um vetor de características; (2) camadas escondidas – processam os dados de entrada e podem compor várias camadas; e (3) camada de saída – processa a saída da rede. A figura 2.8 mostra um modelo genérico de uma rede MLP. Nesse exemplo, a arquitetura da rede é composta por apenas uma camada escondida. Além disso, a estrutura das redes MLP é progressiva, ou seja, as saídas de cada camada são enviadas progressivamente às camadas subsequentes. O número de camadas escondidas e o número de neurônios em cada camada escondida não são limitados. As características mais comuns são [Kuncheva, 2004]:

- A função de ativação ϕ na camada de entrada é a função identidade.
- Não há conexões laterais entre os nós da mesma camada.
- Camadas não adjacentes não são conectadas diretamente.
- Todos os nós de todas as camadas escondidas têm a mesma função de ativação ϕ.

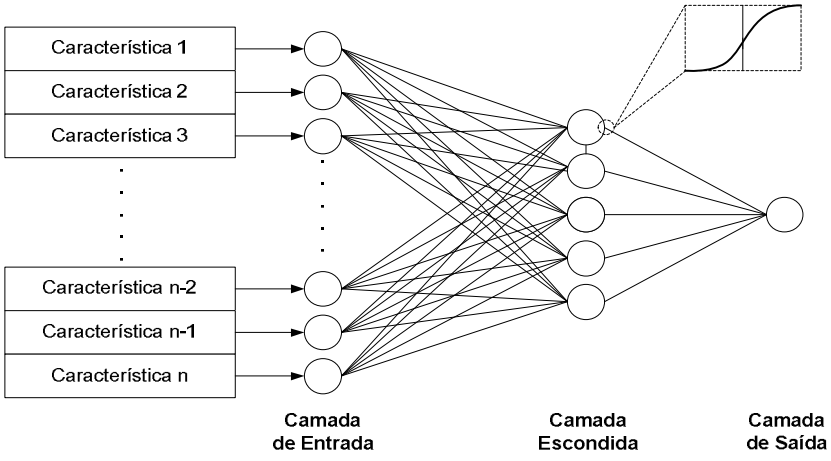

Figura 2.8. Modelo genérico de um classificador do tipo MLP.

O algoritmo usado para o treinamento do classificador MLP é o algoritmo de retro-propagação (*backpropagation)* descrito abaixo, de acordo com [Kuncheva, 2004]:

1. Defina uma arquitetura para a rede: número de camadas escondidas, número de neurônios em cada camada e funções de ativação;
2. Inicialize os pesos aleatoriamente (incluindo *bias*). Defina um número máximo *T* de iterações e um valor de erro de treinamento máximo $\epsilon > 0$;
3. Enquanto (erro de treino $>\epsilon$ e iteração $\leq$ T)
   a) Submeta a próxima amostra de treinamento z_j;

b) Calcule a saída de cada neurônio da camada de saída da rede usando os valores dos pesos atuais;

c) Calcule o erro de cada neurônio de saída.

d) Atualize os pesos de cada neurônio da camada de saída e das camadas escondidas em função do erro.

e) Calcule o erro de treino.

f) Quando toda a base de treino for submetida à rede, incremente o número de iterações.

4. Fim

Redes neurais têm sido aplicadas em detecção de intrusão [Chandola et al., 009], detecção de DDoS [Gavrilis e Dermatas, 2005] e em outros problemas de segurança de redes. Uma das principais vantagens do uso de redes neurais é a elevada taxa de detecção normalmente alcançada pela rede (especialmente em situações com muitas amostras de treinamento disponíveis). Além disso, apresenta baixa complexidade na fase de uso, uma vez que a classificação de uma amostra desconhecida é feita através de um simples produto entre os atributos da amostra e os vetores de peso da rede obtidos durante o treinamento. Porém, o classificador baseado em rede neural é pouco indicado para aplicações cujo treinamento ocorre de forma online devido à elevada complexidade computacional envolvida no treinamento da rede. Outro fator negativo, especialmente em MLP, é o elevado número de parâmetros que devem ser definidos experimentalmente pelo usuário, tais como: número de camadas escondidas, número de neurônios em cada camada escondida, valores inicias dos pesos, número de iterações, dentre outros.

2.2.4.5. Árvores de Decisão

Árvore de decisão é uma das principais técnicas de aprendizagem de máquina, especialmente quando problemas de classificação envolvem atributos nominais. Uma DT (do Inglês *Decision Tree*) é composta por três elementos básicos:

- Nó raiz: corresponde ao nó de decisão inicial que normalmente é gerado utilizando-se o atributo mais discriminante entre as classes envolvidas no problema;
- Arestas: correspondem aos diferentes valores possíveis das características;
- Nó folha: corresponde a um nó de resposta, contendo a classe a qual pertence o objeto a ser classificado.

Em árvores de decisão, duas grandes fases devem ser asseguradas:

1. Construção da árvore: uma árvore de decisão é construída com base no conjunto de dados de treinamento, sendo dependente da complexidade dos dados. Uma vez construída, regras podem ser extraídas através dos diversos caminhos providos pela árvore para que sejam geradas informações sobre o processo de aprendizagem.
2. Classificação. Para classificar uma nova instância, os atributos da amostra são testados pelo nó raiz e pelos nós subsequentes, caso seja necessário. O

resultado deste teste permite que os valores dos atributos da instância dada sejam propagados do nó raiz até um dos nós folhas. Ou seja, até que uma classe seja atribuída à amostra.

Para melhor ilustrar o processo de classificação de uma DT, a Figura 2.9 apresenta um modelo utilizado no treinamento de dados para a tarefa de detecção de intrusão. Inicialmente, uma base de dados supervisionada é submetida ao algoritmo, que por sua vez, gera uma árvore construída para separar amostras da classe "Benigna" de amostras da classe "Maligna". Após a construção da árvore, dados desconhecidos podem ser submetidos ao classificador, o qual atribuirá um das duas classes à amostra testada.

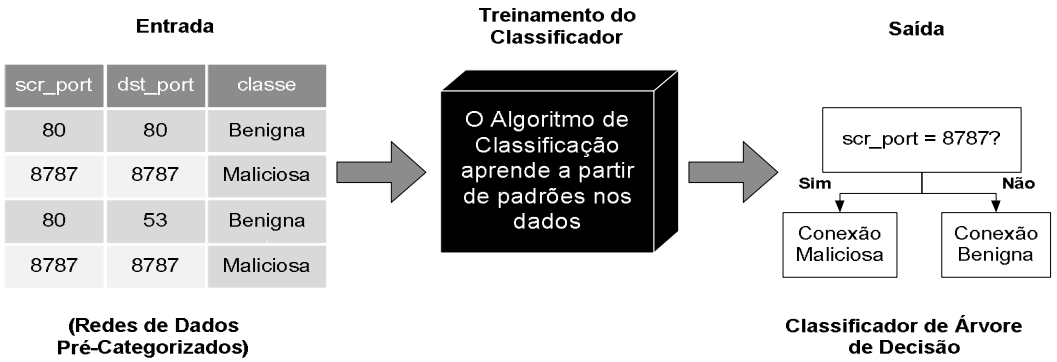

Figura 2.9: Treinamento dos dados em uma Árvore de Decisão.

Vários algoritmos foram desenvolvidos a fim de assegurar a construção de árvores de decisão e seu uso para a tarefa de classificação. O ID3 e C4.5 algoritmos desenvolvidos por [Quinlan, 1993] são provavelmente os mais populares. Vale também mencionar o algoritmo CART de Breiman [Breiman et al., 1984]. A maioria desses algoritmos utiliza uma estratégia descendente, ou seja, desde a raiz até as folhas. A seguir são exibidos detalhes desses algoritmos, de acordo com [Duda et al., 2001].

- ID3 (*Iterative Dichotomiser 3*) é um modelo de DT para dados nominais (não ordenados), embora seja possível a utilização do ID3 com dados contínuos, desde que sejam convertidos em valores discretos. É utilizada uma medida que calcula o ganho de informação para dividir os atributos ao longo da árvore. O algoritmo padrão não possibilita que a árvore gerada seja podada, ação necessária para evitar aprendizagem viciada e para reduzir a complexidade da árvore gerada. Porém, essa função pode ser incorporada ao algoritmo.

- C4.5 é o sucessor e um refinamento do algoritmo ID3, desenvolvido por Quinlan [Quinlan 1993]. O algoritmo usa heurísticas para podar as árvores geradas. Outra vantagem deste algoritmo é o fato de aceitar atributos contínuos e categorizados na construção da DT. C4.5 usa o método do ganho da relação de impurezas para avaliar a divisão dos atributos.

- CARTs (*Classification on Regression Trees*) foi proposto em [Breiman et al., 1984]. Uma das suas características principais é a grande capacidade de pesquisa das relações entre os dados, mesmo quando elas não são evidentes, bem como a produção de resultados sob a forma de árvores de decisão de grande simplicidade e legibilidade. Tal como o seu nome indica, CART pode

ser usado em problemas de classificação (classes discreta) ou regressão (valores contínuos) usando uma mesma tecnologia. O resultado deste algoritmo é sempre uma árvore binária que pode ser percorrida da sua raiz até as folhas respondendo apenas a questões simples do tipo sim/não. A análise é efetuada de forma completamente automática requerendo uma intervenção humana mínima.

Uma das vantagens do uso de Árvores de Decisão em aplicações envolvendo problemas de segurança em redes de computadores é o fato de ser um método não paramétrico, ou seja, não há necessidade de definição de valores de parâmetros. Outra propriedade interessante é que regras podem ser extraídas a partir das árvores. Essas regras explicam claramente o processo de aprendizagem, podendo ser usadas para uma compreensão mais completa dos dados e dos atributos mais relevantes para o problema de classificação. Entretanto, é difícil obter resultados exatos quando há ruídos, devido aos inúmeros detalhes dos dados definidos durante o treinamento. Vale também ressaltar que esta técnica permite a obtenção de resultados, que em geral, são superados apenas por algoritmos de complexidade muito superior.

Além do uso de DTs para extração de regras que explicam o aprendizado do dados, métodos unicamente baseados em indução de regras, como o método RIPPER, também podem ser aplicados em problemas de detecção de anomalias, como em [Likarish et al., 2009].

2.2.4.6. *K-means*

Algoritmos de agrupamento são geralmente usados de forma não supervisionada. Ou seja, é apresentado um conjunto de instâncias de dados que devem ser agrupados de acordo com alguma similaridade. O algoritmo tem como informação de entrada somente o conjunto de características que descrevem cada instância dada.

K-means [MacQueen, 1967] é um popular algoritmo de agrupamento comumente usado para particionar automaticamente um conjunto de dados em *k* grupos. Dado o conjunto de amostras $Z = \{z_1, ..., z_N\}, z_j \in \Re^n$. O usuário deverá definir a quantidade de grupos *c* que *k-means* criará, sendo que para cada grupo haverá um centróide $\mu_1, \mu_2, \ldots, \mu_c$. São selecionados inicialmente *k* centróides, os quais são iterativamente refinados pelo algoritmo da seguinte forma:

1. Inicialize os centróides $\mu_1, \mu_2, \ldots, \mu_c$.
2. Agrupe cada uma das *N* amostras ao centróide mais próximo.
3. Calcule novamente o valor de cada centróide μ_i.
4. Repita os passos 2 e 3 até que não ocorra mais mudanças em μ_i.

As *k* primeiras amostras da base são normalmente escolhidas como os *k* centróides iniciais, enquanto que medidas de distância, como a distância Euclidiana, são usadas para o cálculo da similaridade entre as demais amostras e os centróides de cada grupo. O cálculo das distâncias é, portanto, a etapa que mais exige processamento. Se existem *N* pontos e *k* centróides calcula-se *N* x *k* distâncias neste passo. O centróide mais próximo de cada amostra vai 'incorporá-la', ou seja, a amostra vai pertencer ao grupo representado pelo mesmo. Após, os valores das coordenadas dos centróides são refinados, pois para cada grupo que possui mais de uma amostra, o novo valor do

centróide é calculado através da média dos atributos de todas as amostras do grupo. Esse passo é repetido até a convergência, isto é, o algoritmo calcula a distância entre as amostras da base e os centroides iterativamente, até que mais nenhuma amostra mude de classe.

Chandola et al. [Chandola et al, 2009] destaca que a utilização de técnicas baseadas em agrupamento proporciona algumas vantagens como a operacionalização de maneira não supervisionada; sendo frequentemente adaptada a tipos complexos de dados e; rapidez na fase de teste, desde que o número de grupos com que cada instância precisa ser comparada seja uma constante pequena. Porém, ainda segundo os autores, o uso destas técnicas acarreta desvantagens, como por exemplo, em relação ao desempenho, onde há uma enorme dependência da eficácia dos algoritmos em capturar a estrutura central das instâncias. Além disso, é difícil de ser calculada por não existir conhecimento sobre a saída desejada. Outra desvantagem se deve a complexidade computacional para agrupamento dos dados, sendo frequentemente um gargalo, especialmente para aplicações online como os problemas de segurança em redes de computadores.

2.2.4.7. Combinação de Classificadores

A combinação de classificadores tenta superar a difícil tarefa de construir um único classificador para resolver problemas específicos, através da combinação da decisão de classificadores menos específicos, ou menos definidos para o problema a ser solucionado. Existem diversas razões para a combinação de múltiplos classificadores, algumas dessas razões são encontradas em [Jain et al., 2000]:

1. Um desenvolvedor tem acesso a diferentes classificadores, treinado sem um contexto diferente e para uma representação ou descrição inteiramente diferente do mesmo problema;
2. Muitas vezes, mais de um conjunto de treino é disponível, coletados em um tempo diferente ou em um ambiente diferente. Esses conjuntos de treinamento podem ainda usar características diferentes;
3. Diferentes classificadores treinados sobre o mesmo dado não diferem somente em seu desempenho global. Possivelmente, os classificadores apresentam fortes diferenças locais. Cada classificador tem sua própria região no espaço de características na qual atua com um desempenho melhor;
4. Alguns classificadores, tais como redes neurais, mostram diferentes resultados com diferentes inicializações dos parâmetros devido à aleatoriedade inerente ao procedimento de treino. Nesse tipo de situação, normalmente é feita uma escolha da melhor rede e descarte das demais. Porém, é possível combinar varias redes, na tentativa de fortalecer a aprendizagem dos dados.

Portanto, diferentes conjuntos de características, diferentes conjuntos de treinamento, diferentes métodos de classificação, ou diferentes sessões de treino, podem resultar em um conjunto de classificadores cuja saída é combinada buscando-se melhorar a precisão da classificação global.

Classificadores híbridos e conjuntos de classificadores podem ser considerados dois tipos de combinação de classificadores. A idéia básica de classificadores híbridos [Tsai et al., 2009] é combinar diversas técnicas de aprendizagem de máquina no intuito de melhorar o desempenho geral do sistema. Mais especificamente, uma abordagem hibrida é composta por dois componentes funcionais. O primeiro usa os dados de entrada e gera um resultado intermediário. O segundo atua nos resultados intermediários para produzir o resultado final.

Em [Peddabachigare et al., 2007] tem-se uma abordagem de combinação de classificadores híbridos para detecção de intrusão. É utilizado DT na primeira fase da classificação, resultando em uma saída intermediária. Na segunda fase, SVM é aplicado para que uma saída final seja obtida. A Figura 2.10 mostra a arquitetura desta combinação de classificadores híbridos. DT gera um nó informação com o conjunto de características inicial. Esse nó informação é determinado por regras geradas pela DT. Todo conjunto de dados é assinalado a um nó terminal o qual representa uma classe particular ou um subconjunto. A informação do nó terminal é adicionada ao conjunto de atributos original. Esse novo vetor de características é então manipulado por SVM para a classificação final.

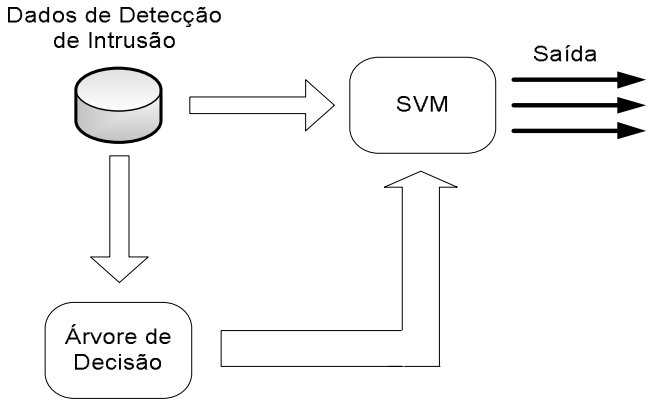

Figura 2.10. Uma abordagem para combinação de classificadores híbridos. [Peddabachigare et al., 2007]

A estratégia de conjuntos de classificadores refere-se à combinação de algoritmos de aprendizagem fraca. Os classificadores são chamados fracos porque são treinados sem definições ideais de parâmetros, com diferentes conjuntos de treino e ou de características, dentre outras estratégias. O objetivo é que o desempenho global seja efetivamente melhorado, quando comparado ao desempenho individual dos membros do conjunto [Tsai et al., 2009].

Existem três estratégias clássicas para a geração de conjuntos de classificadores: (1) *Adaboost;* (2) *Bagging;* e (3) Subespaços Aleatórios [Kuncheva, 2004]. A primeira estratégia é um método iterativo, que cria um conjunto de classificadores da seguinte forma sequencial: cada amostra recebe uma probabilidade (peso) de ser selecionada para compor a base de treinamento usada pelos membros do conjunto. A cada nova iteração, os pesos das amostras incorretamente classificadas pelos membros anteriores são ajustados, para que essas amostras tenham maior probabilidade de serem selecionados para compor a base de treinamento dos classificadores posteriores. *Bagging* também atua fazendo uma seleção de amostras de treinamento. Porém, o

processo de seleção das amostras é totalmente aleatório, ou seja, para cada classificador, é feita uma seleção aleatória das amostras presentes na base de treinamento. Por fim, o método Subespaços aleatórios altera os vetores de características usados por cada membro do conjunto. Cada classificador é treinado com um subconjunto de características escolhidas aleatoriamente.

A técnica conhecida como Florestas Aleatórias, tem sido aplicada em vários problemas de detecção de anomalias [Abu-Nimeh et al., 2007] [Debarr e Wechsler, 2009]. Essa técnica combina *Bagging* e Subespaços Aleatórios para gerar diferentes conjuntos de treinamento, usados para treinar um conjunto de DT. Outro exemplo de método aplicado ao problema de detecção de anomalias encontra-se em [Peddabachigare et al., 2007]. A Figura 2.11 mostra a arquitetura proposta pelos autores utilizando DT, SVM e a abordagem híbrida (DT – SVM). Esta abordagem de combinação de conjunto de classificadores procura usar a pontuação mais alta como saída final entre os classificadores (DT, SVM e DT-SVM).

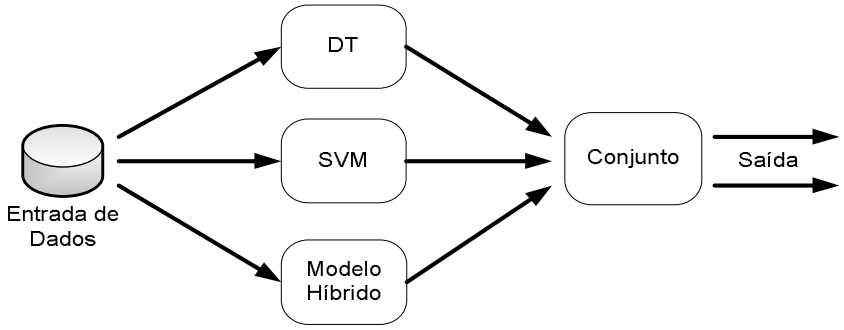

Figura 2.11. Uma abordagem para combinação de conjuntos de classificadores [Peddabachigari et al., 2007]

A combinação de classificadores, seja através de conjuntos ou de classificadores híbridos, tem apresentado resultados muito promissores em trabalhos que investigam segurança em redes de computadores. A principal vantagem desse método está no fato de que os membros do conjunto normalmente não precisam ter seus parâmetros internos ajustados para o problema. Segundo, sistemas de segurança de redes podem ser desenvolvidos utilizando-se conjuntos de classificadores para que dados provenientes de diferentes fontes sejam considerados em conjunto. Por fim, a literatura indica que conjuntos de classificadores normalmente superam as taxas de classificação obtidas por classificadores individuais. A complexidade computacional envolvida na geração e uso de um conjunto de classificadores pode ser considerada a principal desvantagem do método.

As técnicas de aprendizagem de máquina descritas nesta seção estão sendo utilizadas no desenvolvimento de diversos sistemas aplicados a problemas relacionados à segurança de redes de computadores. É importante destacar que muitas outras técnicas também podem ser aplicadas, tais como Regressão Logística, BART (*Bayesian Addictive Regression Trees*), HMM (*Hidden Markov Models*), métodos baseados em aprendizagem por reforço e outros métodos de aprendizagem não supervisionada. Alguns exemplos do uso de aprendizagem de máquina em problemas de segurança de redes são apresentados na próxima seção.

2.3. Aprendizagem de Máquina na área de Segurança na Internet

Nesta seção será examinado o uso de aprendizagem de máquina no domínio de prevenção e detecção de anomalias (também chamado de detecção de intrusão).

As subseções subsequentes discutem os desafios e problemas, bem como as recomendações de uso da aprendizagem de máquina na detecção de anomalias.

2.3.1. Os desafios do uso de aprendizagem máquina na detecção de anomalias

Quando comparada a outras áreas ou domínios de conhecimento, a aprendizagem de máquina aplicada na detecção de anomalias em redes de computadores parece ser mal sucedida. Isso porque, embora existam centenas de pesquisas acadêmicas, o número de ferramentas e soluções resultantes e efetivamente operacionais é muito reduzido. Quais seriam as razões que explicam tal fato? Existem limitadores ou até mesmo culpados?

Respostas a essas perguntas são elucidadas em grande parte nos trabalhos de [Sommer e Paxson, 2010], [Nelson, 2010] e [Tygar, 2011], que identificam algumas características especiais que tornam o emprego da aprendizagem de máquina mais difícil na detecção de anomalias do que em outros domínios. Essas características são:

1. **Detecção de *Outliers***

Analisando o objetivo da detecção de anomalias (identificar atividades que não pertencem a determinado contexto) é possível estabelecer uma relação com o conceito de *outlier*, uma observação que parece ser inconsistente com o restante de um conjunto de dados [Barnett e Lewis, 1978]. Assim, é possível dizer que a detecção de anomalias é a descoberta de *outliers* significativos que representam um comportamento anormal [Sommer e Paxson, 2010]. Nesse contexto, a detecção de *spam* é um exemplo extremamente adequado, visto que o objetivo é classificar e-mails não válidos (*spam*), ou seja, fora da normalidade.

Entretanto, usar técnicas de aprendizagem de máquina na detecção de anomalias para encontrar *outliers* parece contraditório, pois uma regra básica da aprendizagem de máquina é a necessidade de treinar o sistema com um conjunto grande de dados que englobe todos os tipos (espécimes) de todas as classes [Duda et al., 2001]. No entanto, como na detecção de anomalias o objetivo é encontrar novos ataques, não se pode "treinar" os ataques de interesse, uma vez que somente o tráfego normal (sem ataque) é conhecido. Em outras palavras, o treinamento de um sistema de detecção de anomalias acaba com um resultado oposto do esperado.

2. **Alto Custo dos Erros**

Em geral, um sistema de detecção de anomalias é regido por limites rigorosos sobre o número de erros que pode tolerar. Um falso positivo requer gastos (principalmente tempo e dinheiro) para examinar o incidente relatado, o que por muitas vezes acaba por não revelar nada. Por outro lado, falsos negativos têm o potencial de causar sérios danos, já que um único sistema comprometido pode prejudicar gravemente a integridade da infraestrutura de TI (Tecnologia da Informação) de toda uma empresa. Desta forma, o uso da aprendizagem de máquina na detecção de anomalias acrescenta um custo elevado no caso de erros de classificação, principalmente quando comparado a outros domínios de aplicação como, por exemplo, sistemas de recomendação. Uma exceção é a detecção de *spam*, cujo modelo de custo é altamente desequilibrado: falsos

positivos (*e-mails* válidos declarados como *spam*) podem custar caro, como a perda de um negócio, enquanto falsos negativos (*spam* identificados como *e-mails* válidos) não têm impacto significativo.

3. Diferença semântica

Sistemas de detecção de anomalias enfrentam um problema de diferença semântica, ou seja, a dificuldade em transformar seus resultados em ações para os operadores de rede. Em muitos estudos, observa-se uma tendência em limitar a avaliação do sistema apenas à capacidade de identificar, confiavelmente, os desvios de um perfil normal. Ao fazer isso, o resultado é que o operador é mais exigido e, muitas vezes é obrigado a se perguntar: o que isso significa?

Mesmo que, por definição, a detecção de anomalias seja focada em identificar comportamento malicioso (se um evento não visto antes é benigno ou não), não se pode parar nesse ponto, afinal, em ambientes operacionais, sistemas de detecção de anomalias produzem muitos falsos positivos. Vale lembrar que um algoritmo de aprendizado de máquina não comete erros dentro de seu modelo de normalidade e que no fim o que importa para o operador de rede é a interpretação dos resultados.

Por fim, é preciso considerar que existe uma diferença semântica entre o ambiente acadêmico e redes operacionais. Tipicamente, redes operacionais possuem políticas de segurança local (restrições de segurança). Assim, é muito provável que a atividade de um sistema de detecção de anomalias operado em um ambiente acadêmico seja proibida em uma rede corporativa.

4. Diversidade do Tráfego de Rede

Ao contrário do que muitos usuários e administradores de redes imaginam, o tráfego de uma rede apresenta grande diversidade, devido, por exemplo, a fatores como a correlação entre determinado tipo tráfego e a transferência de grandes volumes (*flash crowds*, por exemplo). Essa "inesperada" diversidade de tráfego é muitas vezes regular e comum, e na maioria dos casos não apresenta malefícios às redes. Entretanto, para um sistema de detecção de anomalias, essa variabilidade pode ser manipulada com dificuldade, pois torna a tarefa de encontrar uma noção estável de "normalidade" mais complexa.

A solução mais comum para que um sistema que usa aprendizagem de máquina reconheça a variabilidade como algo corriqueiro é através da agregação do tráfego. A idéia é estabilizar as propriedades do tráfego observando-as em períodos mais longos de tempo (horas e dias, talvez semanas) ao invés de intervalos pequenos (segundos e minutos). Observações feitas durante "horas do dia" e "dias da semana" exibem padrões mais confiáveis. Por exemplo, se durante o intervalo do almoço de um dia, o volume de tráfego é duas vezes maior do que durante os intervalos de tempo correspondentes a última semana, isso provavelmente reflete algo incomum acontecendo.

5. Dificuldades com a Avaliação

A avaliação de um sistema de detecção de anomalias é essencial para comprovar sua eficácia e eficiência. Muitas abordagens promissoras acabam, quando postas em prática, abaixo das expectativas. Por outro lado, a correta elaboração de um esquema de avaliação não é fácil e, muitas vezes, acaba sendo mais difícil do que a construção da própria solução. Devido à opacidade do processo de detecção, os resultados de um

sistema de detecção de anomalias são mais difíceis de prever. Basicamente, três (03) fatores contribuem para isso:

- **Dificuldades de Dados**: O maior desafio para avaliação de sistemas de detecção de anomalias é a falta de conjuntos de dados públicos adequados para testes e validação. A principal causa dessa falta é a "sensibilidade" dos dados. Tipicamente, a inspeção do tráfego de rede pode revelar informações sensíveis como dados pessoais, comunicações confidenciais e segredos de negócios. Diante desse risco, os "donos" do tráfego de rede se sentem ameaçados e criam barreiras organizacionais e legais que impedem sua utilização. Para burlar a falta de dados, pesquisadores têm adotado três estratégias [Sommer e Paxson, 2010]: *i) Simulação* - livre de preocupações quanto à sensibilidade dos dados, mas dificilmente é realista; *ii) Sanitização* (*sanitization*) - informações potencialmente sensíveis são removidas ou "anonimizadas" de dados reais, entretanto, ainda persiste o temor, por parte dos donos, que alguma informação confidencial possa tornar-se pública; e *iii) criação de conjuntos de dados próprios* - dados são gerados em ambientes controlados e o tráfego acaba se tornando muito diferente do real.
- **Diferença semântica**: Além de identificar corretamente os ataques, um sistema de detecção de anomalias precisa também dar suporte ao operador na compreensão da anomalia, permitindo assim uma rápida avaliação do seu impacto. Sempre que o sistema encontra, corretamente, um *exploit* para servidor *Web* desconhecido, mas apenas repassa ao operador a mensagem "o tráfego HTTP do *host* não corresponde ao perfil normal", um esforço adicional é necessário para entender o que aconteceu, mesmo que haja total confiança nos resultados do sistema. Por outro lado, usando novamente o exemplo do *spam*, se o detector informa que um e-mail é *spam*, não há muito espaço para interpretações equivocadas.
- **Configuração contraditória**: Sistemas de detecção de anomalias operam em ambientes contraditórios, em uma verdadeira briga entre gato e rato (atacantes e defensores),onde cada grupo tenta aperfeiçoar suas ferramentas em resposta à elaboração de novas técnicas providas pelo grupo rival. Uma preocupação em particular são as técnicas de evasão, nas quais os atacantes tentam ajustar suas atividades para evitar a detecção. Embora autores como [Sommer e Paxson, 2010], [Nelson, 2010] argumentem que explorar a especificidades de um sistema de detecção de anomalias que usa aprendizagem de máquina requer significativo esforço, tempo e *expertise* do atacante, Fogla e Lee [Fogla e Lee, 2006] provaram, através de uma abordagem automatizada, que é possível gerar ataques mutantes que combinam exatamente com o perfil normal de um sistema.

2.3.2. Quando e Como usar Aprendizagem de Máquina na Detecção de Anomalias

Após ler a subseção anterior e se deparar com vários problemas e dificuldades, o leitor deste minicurso deve estar se perguntando: será que vale a pena usar aprendizagem de máquina para encontrar anomalias em redes de computadores? A resposta é sim.

Para isso, a primeira coisa a ser feita é entender o que o sistema faz ou irá fazer Tipicamente, quando uma solução consegue resultados melhores que outros trabalhos, usando os mesmos dados, é óbvio aceitá-la como uma contribuição. Entretanto, no domínio de detecção de anomalias, é sempre possível achar uma variação de solução que funcione um pouco melhor do que outra em uma determinada configuração. Assim, o ponto chave para o uso da aprendizagem de máquina na detecção de anomalias é a percepção (visão) do comportamento do sistema. Sommer e Paxson [Sommer e Paxson, 2010] enumeram algumas recomendações para o uso de aprendizagem de máquina na detecção de anomalias.

1. Compreender o Modelo da Ameaça

Ao iniciar o desenvolvimento de um detector de anomalia é preciso considerar as ameaças às quais a solução será submetida. Para tanto, algumas perguntas precisarão ser respondidas:

- **Que tipo de ambiente é o alvo do sistema?** A operação em uma rede pequena enfrenta desafios bem diferentes do que para uma grande empresa ou *backbone*. Ambientes acadêmicos são diferentes de empresas comerciais.
- **Quais habilidades e recursos terão os atacantes?** Se um sítio *Web* é considerado de alta importância e, consequentemente, de alto risco a ataques, é preciso se antecipar aos ataques mais sofisticados.
- **Qual o grau de preocupação com evasão?** O grau com o qual os atacantes podem analisar as técnicas de defesa e tentar contorná-las determina os requisitos de robustez para qualquer detector de anomalias.

Em resumo, embora não existam detectores perfeitos, melhores decisões podem ser tomadas quando existe um modelo de ameaças ao sistema.

2. Manter o âmbito restrito

Imaginar que a aprendizagem de máquina é a "lâmpada mágica" e que seu emprego em uma solução de detecção de anomalias é garantia de sucesso é um equívoco comum. Na detecção de anomalias é essencial ter a visão clara dos objetivos da solução proposta, isto é, quais ataques deverão ser detectados. A forma mais simples e direta de se alcançar essa visão é restringindo atividade alvo. Assim, pode-se adaptar o detector as especificidades e reduzir o potencial de erros de classificação. Uma dica para escolher qual algoritmo ou técnica usar é sempre ser capaz de responder o porquê da escolha.

3. Reduzir custos

O principal argumento para o uso de sistemas de detecção de anomalias é a redução dos custos associados com segurança. Contudo, de forma curiosa, a principal queixa sobre sistemas de detecção de anomalias é o número excessivo de falsos positivos, o que aumenta os custos. Uma solução para diminuir custos é reduzir o escopo do sistema, uma vez que sem um objetivo claro a configuração do problema com aprendizagem de máquina terá impacto direto no número de falsos positivos. Outra solução é o uso de estratégias para lidar com a diversidade do tráfego. O uso de esquemas de agregação, que usam intervalos de tempo maiores, e o exame cuidadoso das propriedades particulares do tráfego têm se mostrado bastante úteis. Por fim,

esquemas de pós-processamento com suporte a informações adicionais também têm ajudado a reduzir os falsos positivos [Gu et al., 2007] [Anagnostakis et al., 2005].

4. Avaliação

A avaliação de um sistema de detecção de anomalias deve responder as seguintes perguntas: O que ele pode detectar e por quê? O que não é possível detectar e por que não? Quão confiável ele funciona? Onde pode falhar?

De acordo com a experiência de [Sommer e Paxson, 2010] como revisores, a razão número 1 de rejeições em submissões a conferências de detecção de anomalias reside em falhas ao se explorar adequadamente estas questões em termos de como trabalhar com dados e interpretação de resultados.

- **Como trabalhar com os dados**: Como previamente discutido, o passo mais importante para avaliação é a obtenção de dados para trabalhar. O "nirvana" na detecção de anomalias é obter acesso a um conjunto de dados com tráfego de rede de um grande ambiente real, de vários pontos diferentes da rede. Trabalhar com tráfego real fortalece muito a avaliação, pois pode demonstrar que o sistema pode funcionar na prática. É importante sempre lembrar que para avaliar um sistema de detecção de anomalias é preciso vários conjuntos de dados, isso porque é preciso treinar o sistema com dados diferentes daqueles utilizados para a avaliação final. Uma abordagem padrão para a realização de treinamento e detecção em tráfegos diferentes é a técnica de subdivisão, onde subconjuntos dos dados disponíveis são selecionados através de uma amostragem aleatória.
- **Interpretação dos resultados**: Uma avaliação confiável frequentemente requer o relacionamento das entradas e saídas em baixo nível, ou seja, os pesquisadores precisam examinar manualmente os falsos positivos e negativos. Sempre que uma investigação nesse nível é realizada e não se é capaz de identificar a razão pela qual o sistema incorretamente relatou um caso particular, isso indica uma falta de visão sobre a operação do sistema de detecção de anomalias. Um ponto importante, e muitas vezes esquecido, é a inspeção dos verdadeiros positivos e verdadeiros negativos. Isto porque com a aprendizagem de máquina, muitas vezes não é aparente que o sistema aprendeu, mesmo quando produz resultados corretos. Um exemplo clássico vem de uma rede neural treinada para detectar tanques em fotos, cuja avaliação inicial era realmente capaz de fazê-lo. Contudo, descobriu-se depois que conjuntos de dados usados para treinamento e avaliação compartilhavam uma propriedade sutil: as fotos dos tanques foram tiradas em um dia nublado, enquanto todos os outros elementos não tanques tinham sido tiradas com céu azul. Assim, a rede neural simplesmente aprendeu a detectar a cor do céu.

2.3.3. Exemplos de aplicações da Aprendizagem de Máquina na Detecção de Anomalias

Para melhorar a compreensão dos leitores sobre o assunto, esta seção apresenta alguns dos principais trabalhos, ferramentas e soluções que empregam aprendizagem de máquina em problemas de detecção de anomalias.

2.3.3.1. *Cross-Site Scripting* (XSS)

Ainda pouco explorado neste contexto, o uso de aprendizagem de máquina tem surgindo como um meio eficiente de tratamento de ataques XSS. As propostas mais recentes são descritas abaixo.

[Likarish et al., 2009] apresentam um algoritmo que realiza a busca automática de *scripts* maliciosos na Internet com o apoio de um coletor *Web* (*Crawler*) denominado *Heritrix*. Nos experimentos foi utilizada uma base contendo cerca de 63 milhões de *scripts* benignos (*Alexa*[5]), sendo que destes somente 50.000 foram utilizados. O *Webcrawler* coletou ainda 62 *scripts* maliciosos para a composição da base. Logo, a partir destes dados, o vetor de características foi composto e repassado como parâmetro para diversos classificadores como SVM, DT, *Naive Bayes* e RIPPER para detectar *scripts* maliciosos e avaliar seus respectivos desempenhos, a partir de características focadas na detecção de ofuscação, uma técnica comum para contornar detectores de *malwares* tradicionais. Resultados experimentais demonstraram que SVM obteve resultados superiores aos demais algoritmos de aprendizagem, podendo alcançar até 92% de detecção.

[Riech et al., 2010] apresentam um sistema para detecção automática e prevenção de ataques XSS do tipo *drive-by-download* (*downloads* que ocorrem sem o consentimento do usuário) denominado CUJO (*Classification of Unknown JavaScript Code)*. O sistema foi incorporado em um *Proxy Web* para inspecionar páginas e partes de blocos de códigos *JavaScript* maliciosos. Recursos de códigos estáticos e dinâmicos foram extraídos e analisados a partir de padrões utilizando técnica de aprendizagem de máquina como SVM. O sistema realizou uma avaliação empírica com 200.000 páginas *Web* e 600 ataques do tipo *drive-by-download* para demonstrar a eficácia da abordagem. CUJO foi capaz detectar 94% dos ataques com uma taxa de falso positivo de 0,002%, o que corresponde a 2 alarmes falsos em100 mil sites visitados. Em termos de tempo de execução, CUJO forneceu uma média de 500 *milisegundos* por página *Web*, incluindo o *download* do conteúdo da página *Web* e a análise completa de código *JavaScript*. Na prática, CUJO foi o primeiro sistema capaz de bloquear com eficiência ataques do tipo *drive-by-downloads*.

2.3.3.2. *Phishing*

De acordo com [Sheng et al., 2009], a detecção de *phishing* é basicamente feita através da filtragem de e-mails e páginas *Web*. Com foco na detecção de e-mails *phishing*, [Fette et al., 2007] apresentam um algoritmo capaz de detectar *phishing*, chamado PILFER (do Inglês *Phishing Identification by Learning on Features of Email Received*). A idéia é extrair um conjunto de características (como a presença de *links*, presença de *JavaScript*, entre outras) para treinar e testar classificadores. PILFER usa Floresta Aleatória como classificador, mas outros classificadores como SVM, Árvore Decisão e *Naïve Bayes* também foram testados e podem ser utilizados.

Em termos de resultado, PILFER reduziu significativamente a quantidade de e-mails *phishing* comum custo mínimo em termos de falsos positivo. O método foi avaliado em dois conjuntos de dados publicamente disponíveis: *corpora ham* (projeto *SpamAssassin*), com 6.950 e-mails não-*phishing*, e *phishingcorpus*, com 860 e-mails

[5] *Alexa Top Sites*, http://www.alexa.com/topsites

phishing. Foi utilizada a estratégia de validação cruzada em 10 partições nas duas bases, sendo que a ferramenta atingiu 99,5% de precisão, com uma taxa de falso positivo de aproximadamente 0,0013 e uma taxa de falso negativo de 0,035.

Em [Abu-Nimeh et al., 2007], os autores compararam a precisão preventiva de diversos métodos de aprendizagem de máquina na detecção de e-mails *phishing*, incluindo: Regressão Logística, CART, Floresta Aleatória, *Naive Bayes,* SVM, e BART (*Bayesian Addictive Regression Trees*) [Chipman et al, 2006]. Foram analisados 2.889 e-mails, dos quais 59,8% eram legítimos com 43 características. Ficou demonstrado que, limitando-se ao uso da abordagem *bag-of-words* [Csurka et al., 2004], os classificadores estudados poderiam predizer com sucesso mais de 92% dos e-mails *phishing*. As taxas de falso positivo e falso negativo foram, respectivamente, melhores com Regressão Logística (4,89%) e Floresta Aleatória (11,12%).

Como salientado por [Chandrasekaran et al., 2006] a maioria dos ataques de *phishing* atuais empregam e-mails como principal meio de acesso às vítimas inocentes, levando-as a visitar os *sites* mascarados. Enquanto os mecanismos de defesa recente focam na detecção por validar a autenticidade do site, poucas abordagens têm sido propostas que se concentram na detecção de e-mails baseados em ataques de *phishing* com base nas propriedades estruturais inerentemente presentes nesses emails. Com uma objetivo de atacar o problema de *phishing* através de e-mails, os autores propõem um estratégia de classificação baseada na propriedade estrutural dos e-mails *phishing* (25 características), aplicando *one-class* SVM. Os experimentos foram realizados com uma base de 400 e-mails, sendo 200 do tipo *phishing* e 200 legítimos. A aplicação proposta pelos autores atingiu uma taxa de detecção de 95% de e-mails *phishing* com uma baixa taxa de falso positivo.

No âmbito de detecção de *phishing* em páginas *Web*, [Sanglerdsinlapachai e Rungsawang, 2010] propuseram o uso de um conjunto de classificadores. Os autores utilizaram voto majoritário para definir a classe atribuída à instância pelos seis (6) classificadores (*classifier ensemble*) usados no conjunto (*AdaBoost*, árvore de decisão J48, *Naive Bayes*, Redes Neurais, Floresta Aleatória e SVM) e Média Bayesiana Simples (*Simple Bayes Average* - SBA) para calcular a probabilidade média de cada classe. Nos experimentos foram testadas todas as combinações possíveis entre os classificadores para compor conjuntos de classificadores. Para o treinamento foram usadas 500 páginas *phishing* e 500 páginas *não-phishing*, e para classificação foram usadas 1.500 páginas *phishing* e 1.500 *não-phishing*. Como resultado, os autores apontam uma melhora na precisão de detecção em aproximadamente 30% em relação a métodos heurísticos, com a vantagem de serem métodos de fácil de implementação.

[Miyamoto et al., 2009] utilizou nove (9) técnicas de aprendizagem de máquina (*AdaBoost, Bagging,* SVM, CART, Regressão Logística, Floresta Aleatória, Redes Neurais, *Naive Bayes* e BART) para detecção de páginas *Web* de *phishing*. Na proposta dos autores, todas as técnicas foram usadas para classificar se um site era ou não *phishing*. Para avaliação das técnicas, foram utilizadas 3.000 amostras, sendo 50% para sites *phishing* e 50% para sites legítimos. Os autores empregaram validação cruzada com quatro (4) partições para melhor obter a média do resultado. A menor taxa de falso positivo foi 13,64%, usando *Naive Bayes*, enquanto a menor taxa de falso negativo foi 13,54%, usando Redes neurais.

A Tabela 2.4 sumariza as soluções apresentadas nesta seção para detecção de *phishing* que utilizam técnicas de aprendizagem de máquina.

Tabela 2.4. Sumarização das soluções para detecção de *phishing*

Publicações	Técnica de AM	Bases	Resultados		
[Fette et al., 2007] PILFER	Floresta aleatória	*SpamAssassin* (6.950 e-mails) e *Phishingcorpus* (860 *phishing*)	Precisão de 99,5%, com FP de 0,0013% e FN de 0,035%		
[Abu-Nimeh et al., 2007]	Regressão logística, CART, SVM, NN, BART e Floresta aleatória	1.718 e-mails legítimos e 1.171 *spams*	FP de 4,89% (Regressão logística) e FN de 11,12% (Floresta Aleatória)		
[Chandrasekaran et al., 2006]	*one-class* SVM	400 *e-mails*, sendo 200 *phishing* e 200 legítimos	Precisão de 95%		
[Sanglerdsinlapachai e Rungsawang, 2010]	*AdaBoost*, Árvore de decisão J48, *Naïve Bayes*, NN, Floresta aleatória e SVM	*CleanMX* e *PhishTank* (*phishing)* e *anti-phishing toolbar*, *Google* e *Yahoo!*, (não-*phishing)*.	*F-measure (Medida Harmônica de Precisão)* de 0,8 e uma taxa de erro de 20%		
[Miyamoto et al., 2009]	*AdaBoost*, *Bagging*, SVM, CART, Regressão logística, Floresta aleatória, NN, *Naive Bayes* e BART	*PhishTank*, *3Sharp*, *Alexa Web Search* e *Yahoo!*	Técnica	FP	FN
			AdaBoost	14.49%	13.83%
			Bagging	14,27%	15,36%
			SVM	15,02%	13,57%
			CART	14,58%	18,16%
			Regr. logística	14,12%	15,10%
			Flor. aleatória	14,54%	14,57%
			NN	14,88%	13,54%
			Naive Bayes	13,64%	15,74%
			BART	14,50%	14,39%

Legenda: FP (Falso Positivo), FN (Falso Negativo) e NN (*Neural Networks* – Redes Neurais)

2.3.3.2. Negação de Serviço

No âmbito de combater ataques de negação de serviço, pesquisadores também têm proposto o uso de técnicas de aprendizagem de máquina. Algumas aplicações de detecção de ataques de negação de serviço usando aprendizagem de máquina são apresentadas a seguir.

Em [Gavrilis e Dermatas, 2005], os autores apresentam e avaliam um sistema de detecção de negação de serviço em redes públicas baseado em Redes Neurais de Base Radial (RBF-NN) e características estatísticas. Um pequeno número de descritores estatísticos foram utilizados para descrever o comportamento de ataques DDoS, sendo que uma elevada taxa classificação foi alcançada por RBF-NN. O método proposto foi simulado e avaliado em uma rede pública, obtendo taxa de detecção superior a 98% dos ataques DDoS, utilizando apenas três características estatísticas.

[Nguyen e Choin, 2010] apresentam um método de detecção proativa de ataques DDoS, contendo duas fases sequenciais: Detecção e Prevenção. Primeiramente,

analisam a arquitetura DDoS para descrição e obtenção das fases iniciais como controle e ataque. A partir deste ponto, investigam os procedimentos de ataques DDoS para selecionar variáveis relevantes no reconhecimentos de ataques DDoS, uma vez que o comportamento anormal é inconstante e altera sempre que o ataque ocorre. Por fim, KNN é aplicado para classificar o estado das redes para cada fase do ataque DDoS. O classificador obteve taxa de precisão equivalente a 91,8%.

[Xu et al.,2007] propuseram um método de detecção de DDoS baseado em Modelos Ocultos de Markov (HMMs) [Rabiner, 1989] e aprendizado por reforço. O método utiliza um esquema de modelo colaborativo para detecção baseado no monitoramento do IP de origem. Para realizar a detecção automática de ataques DDoS, os detectores são distribuídos nos nós intermediários da rede ou perto das fontes de ataques DDoS. HMMs são utilizados para estabelecer um perfil para o tráfego normal baseados nas frequências dos novos endereços IP. O algoritmo de aprendizagem por reforço calcula as estratégias para as trocas de informações entre os múltiplos detectores distribuídos de forma que a precisão na detecção possa ser melhorada sem exigência de alta carga de informações na comunicação entre os detectores. O método mostrou que a utilização de HMMs e algoritmos colaborativos pode ser aplicado para equilibrar a precisão na detecção de ataques DDoS e a carga de comunicação. A precisão obtida pelo sistema foi 98,8%.

No trabalho de [Kumar e Selvakumar, 2011] é apresentada a proposta do algoritmo denominado RBPBoost, com o objetivo de aumentar o desempenho do classificador RBP (do Inglês *Resilient Back Propagation*). Para tanto, RBPBoost combina o conjunto de saídas de RBP e uma estratégia de minimização dos custos, para a decisão final do classificador. Um conjunto de dados disponíveis publicamente foi utilizado nos experimentos, tais como KDD Cup[6], DARPA 1999[7], DARPA 2000[8] e CONFICKER[9]. RBPBoost foi treinado e testado com DARPA,CONFICKER, e um conjunto de dados coletados através de *webcrawlers*. Como forma de avaliar o desempenho do algoritmo de classificação, foram utilizadas duas métricas: a precisão na detecção e o custo por amostra. Os resultados de simulação evidenciaram que o algoritmo RBPBoost alcançou uma precisão de 99,4%, com uma redução na quantidade de falsos alarmes, superando os demais algoritmos de Redes Neurais, comumente utilizados na literatura.

[Wu et al, 2011] propõem um sistema de detecção de ataques DDoS que se baseia em decidir o padrão de fluxo de tráfego analisando ocorrências em diversas situações de ataque. A detecção de ataques em situação normal foi considerada como um problema de classificação. Nesse contexto, os autores utilizaram 15 atributos, que não apenas monitoram a entrada/saída da taxa de pacotes/*bytes*, mas também compilam o TCP SYN e taxa de *flags* ACK, para descrever o padrão de fluxo de tráfego. Para detectar o fluxo de tráfego anormal, os atributos de testes foram aplicados utilizando a técnica de árvore de decisão C4.5. Este novo padrão de tráfego foi usado combinando procedimentos para identificar fluxo de tráfego normal, que segundo os autores é semelhante ao fluxo de ataque, para rastrear a origem de um ataque. O sistema foi capaz

---

6 http://www.ll.mit.edu/IST/ideval/data/1998/1998_data_index.html

7 http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.htm

8 http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/2000data.html

9 http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml

de detectar ataques DDoS com uma taxa de falso positivo variando entre 1,2% e 2,4%, e a razão falso negativo sobre 10% e 20%. O sistema também foi capaz de mensurar os fluxos de tráfegos anormais, com taxas de falso negativo variando de 8% a 12% e taxa de falsos positivos de 12% a 14%.

A Tabela 2.5 sumariza as soluções apresentadas nesta seção para detecção de ataques de negação de serviço que utilizam técnicas de aprendizagem de máquina.

Tabela 2.5. Sumarização das soluções para detecção de ataques de negação de serviço

Publicação	Técnica de AM	Base	Resultados (%)
[Gavrilis e Dermatas, 2005]	RBF-NN	Tempo Real	Precisão de 98%
[Nguyen e Choin, 2010]	*k*-NN	[Darpa 2000]	Precisão de 91,8%
[Xu et al.,2007]	HMMs	[TFN2K[10]]	Precisão de 98,8%
[kumar e Selvakumar, 2011]	NN	[Darpa 1999]	Precisão de 99,4%
[Wu et al., 2011]	DT	[Benzel et al., 2006]	Precisão de 98%

2.3.3.3 Códigos Maliciosos (*Malware*)

Considerando-se a ameaça de *malware* e sua prevalência, não é surpreendente que uma quantidade significativa de pesquisadores tenha concentrado esforços no desenvolvimento de técnicas para coletar, estudar e mitigar código malicioso. Por exemplo, há estudos que medem o tamanho de *botnets*, a prevalência de *sites* mal-intencionados e a infestação de executáveis com *spyware*. Além disso, existem ferramentas que podem executar amostras desconhecidas e monitorar o seu comportamento. Algumas ferramentas fornecem relatórios que resumem as atividades de programas desconhecidos em nível da API (*Application Programming Interface*) do *Windows* ou chamadas de sistema. Esses relatórios podem ser avaliados para encontrar conjuntos de amostras que se comportam da mesma forma, ou para classificar o tipo de atividade maliciosa observada. Outras ferramentas incorporam fluxo de dados em análise, o que resulta numa visão mais abrangente da atividade de um programa na forma de grafos.

Abordagens baseadas em aprendizagem de máquina para detecção de códigos maliciosos utilizam algoritmos para treinamento dos dados no intuito de detectar *malwares*. No entanto, uma vez que classificadores baseados em aprendizagem supervisionada requerem um elevado número de dados rotulados para cada uma das classes (isto é, *malware* e benigno), a indústria anti-*malware* vem auxiliando e investindo pesado em pesquisas nesta área. Alguns exemplos de aplicações para detecção de *malwares* utilizando aprendizado de máquinas são apresentados a seguir.

[Schultz et al., 2001] foram os primeiros a introduzir o conceito de aplicação de algoritmos de aprendizado automático para a detecção de *malware*, com base em seus respectivos códigos binários, aplicando diferentes classificadores como RIPPER, *Naive Bayes* e a combinação de classificadores e considerando três tipos de conjuntos de características: (i) cabeçalhos de programa, (ii) cadeias e (iii) sequencias de *bytes*. Posteriormente, [Kolter e Maloof, 2004] melhoraram os resultados de Schultz, aplicando *n-grams* (ou seja, sobrepondo sequências de *bytes*). Os resultados

[10] http://www.cert.org/incident_notes/IN-99-07.html

experimentais obtidos com Árvore de Decisão foram superiores aos demais algoritmos de aprendizagem de máquinas investigados nos experimentos.

[Rieck et al., 2008] apresentaram a proposta de uma abordagem de classificação automática de comportamento *malware*, tendo como base os seguintes itens: *i)* a incorporação de rótulos atribuídos pelo software antivírus para definir classes para a construção de modelos supervisionados; *(ii)* o uso de recursos específicos, como cadeia de caracteres, que descrevem padrões de comportamento de *malware*; *iii)* a construção automática de modelos discriminativos utilizando algoritmos de aprendizagem como SVM e; *iv)* identificação das características dos modelos explicativos que mais influenciaram o aprendizado, produzindo um *ranking* de padrões de comportamento de acordo com seus pesos. Na prática o método funciona da seguinte forma: coleta-se um número de amostras de *malware* e analisa-se seu comportamento usando um ambiente *Sandbox*, identificam-se *malware* típicos a serem classificados, executando um software antivírus padrão, e constrói-se um classificador baseado no comportamento de *malware* aprendendo as classes simples dos modelos. A precisão obtida pelo método foi equivalente a 70%.

[Wang et al., 2003] propõem um método de detecção de vírus até então desconhecidos usando técnicas de aprendizagem de máquina como DT e *Naive Bayes*. O método utiliza uma base de dados contendo 3.265 códigos maliciosos e 1.001 benignos. Todo o processo de extração de características foi realizado automaticamente, a partir de uma abordagem dinâmica. Por fim, os vetores foram utilizados para treinar os classificadores e detectar os possíveis vírus. Resultados experimentais mostraram que a precisão do modelo utilizando DT foi superior a *Naive Bayes*, obtendo precisão de 91,4% e 77,1%, respectivamente.

[Kolter e Mallof, 2006] desenvolveram MECS (*Malicious Executable Classification System*) para detectar automaticamente executáveis maliciosos, sem pré-processamento ou remoção de qualquer ofuscação. O sistema utiliza um conjunto de 1.971 executáveis benignos e 1.651 executáveis maliciosos, com uma variedade de mecanismos de transporte e carga úteis (por exemplo, *key-loggers* e *backdoors*). Embora todos os códigos sejam executados no *MSWindows*, vale ressaltar que o métodos não se restringe unicamente a este sistema operacional. As sequências de *bytes* dos executáveis foram convertidas em *n-grams* e, então, indexadas no vetor de características, que por sua vez foi repassado como parâmetro de entrada para treinamento e teste de diversos classificadores como: KNN, *Naive Bayes*, SVM e DT. Neste domínio, os autores atentaram para o problema de falsos alarmes, o que ocasiona custos desiguais de classificação. Desta forma, os métodos foram avaliados usando análise ROC (*Receiver Operating Characteristic*) como métrica de desempenho. DT superou os diversos outros métodos, com área ROC de 0,996.

Por fim, [Ye et al., 2009] apresentam o IMDS (*Intelligent Malware Detection System*), ou seja, um sistema inteligente para detecção de *malwares*, mais precisamente, vírus polimórficos. O IMDS é aplicado sobre o OOA (*Objective-Oriented Association*) baseado em técnicas de mineração de dados. O sistema realiza análise de sequências de execução de diversas APIs do *Windows*, que refletem o comportamento dos códigos maliciosos. A captura da semântica da associação de APIs é essencial para a detecção de *malware*. Em seguida, a detecção de *malware* é realizada diretamente no PE (*Windows Portable Executable*) em três etapas: *i)* Construção das sequencias de

execução de API através do desenvolvimento de um analisador PE; *ii)* Extração de regras usando OAA e técnicas de mineração de dados e; *iii)* Classificação baseada nas regras de associação geradas na segunda etapa. O sistema utiliza uma base de dados contendo 29.580 executáveis, sendo 12.214 códigos benignos e 17.366 maliciosos. O sistema IMDS foi comparado com diversos antivírus populares como *Norton AntiVirus* e *McAfee VirusScan*[11], sendo que os resultados experimentais foram superiores a eficiência dos demais antivírus. Para isso, foram utilizadas técnicas de aprendizagem de máquina como *Naive Bayes*, SVM e DT. Este é um exemplo clássico que mostra o resultado de aplicações de técnicas de aprendizagem de máquina em segurança de redes, pois a abordagem introduzida pelo IMDS resultou na incorporação do sistema à ferramenta de verificação do *Kingsoft Antivirus*.

A Tabela 2.6 sumariza as soluções apresentadas nesta seção para detecção de *malwares* que utilizam técnicas de aprendizagem de máquina.

Tabela 2.6: Sumarização das soluções para detecção de *malwares*.

Publicação	Técnica de AM	Base	Resultados (%)
[Schultz et al., 2001]	RIPPER, *Naive Bayes*	*MacAfee*	Precisão de 71,05% (RIPPER), 97,76% (*Naive Bayes*)
[Rieck et al., 2008]	SVM	8.082 amostras, sendo que 3.139 *malwares*	Precisão de 70%
[Ye et al., 2007]	SVM, *Naive Bayes*, DT	29.580 amostras, sendo 17.366 maliciosos	Precisão de 90,54% (SVM), 83,86% (*Naive Bayes*) e 91,49% (DT)
[Wang et al., 2003]	DT, *Naive Bayes*	3.265 executáveis maliciosos e 1.001 benignos	Precisão de 91,4% (DT) e 77,1% (*Naive Bayes*)
[Kolter e Mallof, 2006]	*k*-NN, *Naive Bayes*, SVM, DT	1.971 benignos e 1.651 malignos	Precisão de 98,99% (*k*-NN), 98,87% (*Naïve Bayes*), 99,03% (SVM) e 99,58% (DT)

2.3.3.5. *Spam*

Em relação às soluções para detecção de *spam* baseadas em aprendizagem de máquina, o foco principal inclui a proteção para páginas *Web* através da coleta de e-mails, filtros de lista negra (*blacklist*), filtros de listas brancas (*whitelist*), filtros de listas cinza (*greylist*), análise de *spam* baseado em texto e combate a *spam* baseado em imagem.

A solução proposta por [Debarr e Wechsler, 2009] usa uma classificação híbrida com foco na construção eficiente de modelos para detecção de *spam*. A idéia é fazer o agrupamento de mensagens, permitindo assim, a rotulagem eficiente de um exemplo representativo de mensagens para um aprendizado de modelo de detecção usando Florestas Aleatórias para classificação e aprendizado ativo (seleção efetuada por um algoritmo do computador) para refinar o modelo de classificação. Os autores usam *Area Under the receirver operating characterisitc Curve* (AUC) [Bradley, 1997], como forma de medir o desempenho dos classificadores. Com essa medida Florestas Aleatórias atingiram um desempenho de 95%.

[11] *MacAfee*, http://www.mcafee.com.

O trabalho de [Blanzieri e Bryl, 2007] propõe a avaliação de desempenho de SVM vizinho mais próximo. Em linhas gerais, o classificador proposto funciona da seguinte maneira: a fim de classificar uma amostra *x*, o SVM-NN seleciona as *k* amostras treinadas mais próximas da amostra *x* e então, usa essas *k* amostras para treinar um modelo SVM o qual é posteriormente usado para tomar a decisão. Para avaliar o classificador foram utilizadas 4.150 mensagens legítimas e 1.897 mensagens *spam* do *SpamAssassin Corpus*[12]como conjunto de dados para os experimentos, conseguindo atingir uma taxa de verdadeiro positivo de 99%.

A aplicação de [Castillo et al., 2007] está fundamentada em um sistema de detecção de *spam* usando Árvore de Decisão, *Bagging* e agrupamento. Os autores trabalham em três níveis para incorporar a topologia gráfica da *Web*: *1)* agrupando o gráfico host; *2)* rotulando todos os hosts no grupo por voto majoritário; e *3)* propagando os rótulos esperados para os hosts vizinhos. Os rótulos esperados dos hosts vizinhos são usados como novas características e o classificador é re-treinado. O conjunto de dados usado para os experimentos foi o *WEBSPAM-UK2006*[13], uma coleção de *spam Web* disponível publicamente. Os resultados obtidos foram de 78,7% de verdadeiro positivo e 5,7% de falso negativo com *Bagging*.

[Markines et al., 2009], focando a detecção de *spam* social, utilizaram a ferramenta Weka[14] para conduzir experimentos usando trinta (30) algoritmos de aprendizagem de máquina. O conjunto de dados utilizado para demonstrar os experimentos é uma coleção privada, disponível como parte do ECML/PKDD 2008[15] (*Discovery Challenge on Spam Detection*), com um total de 27.000 usuários, dos quais 25.000 são usuários *spammers* e 2.000 são usuários legítimos. Os resultados obtidos foram de 98% de precisão para detecção de spam social e 2% de falso positivo, para o classificador de Regressão Logística. Os demais classificadores também tiveram um desempenho com precisão em torno de 96% e falsos positivos menores que 5%. Os autores se utilizaram da estratégia de validação cruzada com 10 partes para conduzir o treinamento e teste.

[Wu, 2009] apresenta um método híbrido baseado em regras de processamento e redes neurais *back-propagation* para a filtragem de spam. Em vez de usar palavras chaves, usa o comportamento *spamming* como característica para descrever e-mails. Um processo baseado em regras é usado primeiro para identificar e digitalizar o comportamento *spamming* observado de cabeçalhos e *logs* de e-mails. A coleção de dados foi obtida através do MTA (do Inglês *Mail Transfer Agent*) com um total de 120.207 amostras das quais 50.651 são e-mails legítimos e 69.556 é *spam*. Os autores atingiram uma precisão de 99,6% com seu método.

A Tabela 2.7 sumariza as soluções apresentadas nesta seção para detecção de *spam* que utilizam técnicas de aprendizagem de máquina.

---

12 http://wiki.apache.org/spamassassin/CorpusCleaning

13 http://barcelona.research.yahoo.net/webspam/datasets/uk2006/

14 http://www.cs.waikato.ac.nz/ml/weka/

15 http://www.kde.cs.uni-kassel.de/ws/rsdc08

Tabela 2.7. Sumarização das soluções para detecção de spam

Publicações	Técnicas de AM	Base	Resultados (%)
[Blanzieri e Bryl, 2007]	Combinação de SVM e k-NN	*SpamAssassin Corpus* - 4.150 mensagens legitimas e 1.897 *spam*	99% de TP
[DeBarr e Wechsler, 2009]	Agrupamento, Floresta Aleatória	TREC *Spam EmailCorpus*[16] com 75.419 amostras	95,2% AUC
[Markineset al., 2009]	30 algoritmos de aprendizagem	ECML/PKDD 2008 com um total de 27.000 usuários	98% precisão e 2% FP com Regressão Logística
[Castillo et. al., 2007]	DT, *Bagging* e agrupamento	WEBSPAM-UK2006	78,7% de TP e 5,7% de FN com *Bagging*
[Wu, 2009]	MLP	Coleção privada usando MTA, 120.207 e-mails	99,60 de precisão

Legenda: FP (Falso Positivo) , FN (Falso Negativo) e TP (Verdadeiro Positivo)

2.3.3.6. Detecção de Intrusão

Na literatura, a detecção de intrusão tem sido tratada através da proposta de diversos sistemas que utilizam técnicas de aprendizagem de máquina, tais como: redes neurais artificiais [Souza e Monteiro, 2009] [Xia et al., 2010], *k*-means [Tian e Jianwen, 2009], *k*-NN [Tsai e Lin, 2010] e SVM [Xiao et al., 2007], entre outras. Tais técnicas têm sido usadas como: classificadores individuais, classificadores híbridos e conjunto de classificadores.

Um exemplo de uma aplicação recente encontra-se em [Horng et al., 2011]. Os autores propuseram um sistema de detecção de intrusão baseado em agrupamento hierárquico, seleção de características e SVM. O método foi avaliado na base KDD Cup 1999, composta por 41 características e 4.898.431 amostras para treino e 311.029 amostras para teste. Os resultados atingidos foram de 95% de precisão e 0,7% de falso positivo.

Semelhante a aplicação anterior, [Khan et al., 2007] usaram análise de agrupamento para aproximar vetores de suporte a fim de acelerar o processo de treino de SVM. Desta forma, os autores propuseram o agrupamento de árvores baseado em SVM para melhorar a precisão do classificador. Os experimentos foram conduzidos sobre a base de dados DARPA 1998 com 41 características e cinco classes (Normal, DoS, *Probe*, U2R[17], R2L[18]). A avaliação desses experimentos sobre as cinco classes obteve os seguintes resultados em relação à precisão: 95% (normal), 97% (DoS), 23% (U2R), 43% (R2L) e 91% (*Probe*).

A solução de [Tsai e Lin, 2010] é conhecida como TANN (*Triangle Area based Nearest Neighbor*). Esse método transforma o espaço de características original em um novo espaço de características. Inicialmente, *k-means* é utilizado para agrupar as amostras da base de dados. Com base nos grupos encontrados, são calculadas áreas de triângulos formados entre todas as amostras e os centros de cada grupo. Na segunda

16 http://trec.nist.gov/data/spam.html
17 U2R – User to Root
18 R2L – Remote to Local

fase, um novo vetor de características formado pelas áreas triangulares é usado como entrada para o classificador *k*-NN que definirá a classe de cada amostra. A base de dados investigada foi a KDD Cup 1999, atingindo uma precisão de 99,01%, uma taxa de detecção de 99,27% e uma taxa de falso alarme de 2,99%.

Fazendo uso de Redes Neurais de Kohonen e SVM, [Mafra et al. 2008] propuseram um sistema multicamadas, chamado POLVO–IIDS (*Intelligent Intrusion Detection System*). A primeira camada analisa o tráfego da rede e o classifica em quatro categorias: *DoS*, *Worm*, *Scan*, *R2L*. A segunda camada é responsável pela detecção de intrusão propriamente dita. Os experimentos foram realizados com a base de dados KDD Cup 99, sendo que os seguintes resultados foram obtidos: taxa de acerto de 96,55% e um desvio máximo de 9,53%.

[Xiao et al., 2007] propuseram uma aplicação que aborda a técnica de conjunto de classificadores, utilizando três (3) SVMs. O primeiro classificador é treinado com dados originais, sem alterações, como normalmente é feito com classificadores individuais. O segundo classificador é treinado com dados originais submetidos a um processo de seleção de características. Por fim, o último SVM é treinado apenas com uma parte dos dados de entrada, isto é, é feita uma seleção de amostras. A combinação da decisão do conjunto é obtida através de uma função de voto ponderado. A base de dados utilizada nos experimentos foi à base KDD Cup 99. Os autores alcançaram os seguintes resultados: 99% para taxa de detecção e 0,0018% para falso positivo.

A Tabela 2.8 sumariza as soluções apresentadas nesta seção para detecção de intrusão que utilizam técnicas de aprendizagem de máquina.

Tabela 2.8. Sumarização das soluções para detecção de intrusão

Publicações	Técnicas de AM	Base	Resultados (%)
[Horng et al., 2011]	Agrupamento e SVM	KDD Cup 1999	95,71% de Precisão e 0,7% FP
[Khan et al., 2007]	Agrupamento de Árvores e SVM	DARPA 1998	Precisão para cada classe: 95% (normal), 97% (DoS), 23% (U2R), 43% (R2L) e 91% (Probe)
[Mafra et al, 2008]	Redes Neurais de *Kohonen* e SVM	KDD Cup 1999	Taxa de acerto 96,55% e desvio máximo de 9,54%
[Tsai e Lin, 2010]	*k-means* e *k*-NN	KDD Cup 1999	Precisão de 99,01%, taxa de detecção 99,27% e falso alarme de 2,99%
[Xiao et al., 2007]	Conjunto de Classificadores (SVM)	KDD Cup 1999	Taxa de Detecção 99% e falso positivo 0,0018%

2.3.4. Síntese

Supondo que neste ponto, o leitor se sinta confortável e familiarizado com as diferentes técnicas de aprendizagem de máquina para detecção de anomalias, os autores deste minicurso têm algumas considerações a fazer em relação às diversas aplicações apresentadas na seção anterior. Essas considerações não têm como objetivo abordar ou esgotar todos os aspectos positivos, negativos e de complexidade computacional relacionados à aprendizagem de máquina para detecção de anomalias.

Um aspecto que se pode observar nas varias aplicações apresentadas anteriormente é a falta de dados rotulados para treino/validação, o que também é apontado por [Chandola et al., 2009] como uma desvantagem em detecção de intrusão utilizando técnicas de aprendizagem de máquina supervisionada.

Outro ponto a considerar são as várias técnicas de aprendizagem de máquina aplicadas nas soluções exibidas confirmando certa tendência de classificadores específicos para problemas peculiares, conforme mostra a Tabela 2.9. Por exemplo, o classificador SVM, que é considerado na literatura como um classificador estável e com bom desempenho individual [Tsai et al., 2009] para detecção de anomalias, continua se mostrando da mesma forma em combinações com outros classificadores seja em classificadores híbridos ou em conjuntos de classificadores.

A técnica de classificação baseada em Árvores de Decisão também tem sido utilizada em muitos problemas relacionados à detecção de anomalia. Observa-se também que as técnicas de aprendizagem supervisionada são utilizadas como muito mais frequência quando comparadas aos métodos não supervisionados. Por fim, a combinação de classificadores em conjuntos e em estratégias híbridas parece estar tornando-se uma estratégia padrão na área de segurança de redes, assim como ocorre em diversas outras áreas de aplicação.

Com um enfoque diferenciado das tabelas apresentadas anteriormente que sumarizam aplicações para determinadas ameaças, a Tabela 2.9 sintetiza as aplicações para detecção de anomalia correlacionada com as técnicas de aprendizagem de máquinas. Essa síntese proporciona visualizar quais técnicas de aprendizagem de máquina tem uma melhor eficácia e eficiência em problemas de detecção de anomalia em redes de computadores para determinadas abordagens de problemas (*spam*, *phishing*, XSS, *malware, DDoS*).

2.4. Comentários Finais e Direções Futuras

Este capítulo procurou introduzir o leitor no universo da aprendizagem de máquina aplicada à prevenção e detecção de anomalias. Primeiramente, a Seção 2.1 introduziu o problema envolvendo o tráfego malicioso, onde foram descritas algumas das principais ameaças, bem como algumas das soluções existentes. Uma contextualização sobre o uso de aprendizagem de máquina na detecção de anomalias também foi discutida.

Em seguida, a Seção 2.2 forneceu uma visão geral da área de aprendizagem de máquina. Foram apresentados conceitos básicos essenciais e a terminologia utilizada, além da categorização dos métodos de aprendizagem. Por fim, foram descritas as principais técnicas de aprendizagem de máquina. Com as informações apresentadas na seção 2.2, os autores esperam que os leitores possam ser capazes de entender de maneira geral cada técnica ou sintam-se aptos a consultar alguns dos livros e artigos citados. Para os interessados, o livro de Witten e Frank [Witten e Frank, 2000] é um excelente próximo passo, principalmente porque os leitores podem baixar o Weka, um software livre, de código aberto, com implementações em Java de um grande número de algoritmos para construção de modelos, seleção de atributos e outras operações. Experimentar é uma ótima maneira de compreender melhor os aspectos práticos da aprendizagem de máquina. Existe ainda um livro complementar ao primeiro, Mena [Mena, 2003] que trata de várias abordagens de mineração de dados aplicadas a segurança de computadores, onde técnicas de aprendizagem de máquina e mineração de

dados são aplicadas em várias áreas do conhecimento, incluindo a detecção de intrusão, detecção de fraude e de perfis criminais.

A Seção 2.3 apresentou o emprego da aprendizagem de máquina no contexto da segurança de redes de computadores. Primeiramente foram discutidos os desafios do uso e implementação, onde se pode observar claramente que, por se tratar de um domínio específico e cheio de nuances, diversas questões, que não interferem em outras áreas do conhecimento, são problemáticas para detecção de anomalias, incluindo: *i)* a necessidade de detecção de *outlier*, embora a aprendizagem de máquina apresente melhor desempenho na atividade de encontrar semelhanças; *ii)* os elevados custos dos erros de classificação, que fazem com que, quando comparados a outros domínios, as taxas de erro cheguem a ser irrealistas; *iii)* uma lacuna semântica entre os resultados de detecção e sua interpretação operacional; *iv)* a enorme variabilidade do tráfego benigno, o que torna difícil encontrar noções estáveis de normalidade; e *v)* desafios significativos com a realização de avaliação e testes das soluções.

Em seguida, de forma a superar estes desafios, foram fornecidas algumas diretrizes para a aplicação de aprendizagem de máquina para detecção de anomalias de rede, incluindo: *i)* a importância de se obter uma visão completa, em termos de capacidades, limitações e operação, do sistema de detecção de anomalias a ser desenvolvido; e *ii)* o reconhecimento do papel da avaliação para comprovação da efetividade e eficiência do sistema de detecção.

2.4.1. Oportunidades de Pesquisa

É evidente a partir das discussões apresentadas neste minicurso que a aprendizagem de máquina é naturalmente uma valiosa ferramenta para a detecção e prevenção de atividades maliciosas. Algumas sugestões que poderiam ser passíveis de abordagens de aprendizagem de máquina são destacadas a seguir:

- Embora a maioria das pessoas que utiliza redes sociais não represente uma ameaça, pessoas mal-intencionadas estão sendo atraídas para estas redes por causa da acessibilidade e da quantidade de informação pessoal disponível. Tais informações podem ser usadas para conduzir ataques de engenharia social [Lam e Yeung, 2007] e distribuição de códigos maliciosos. O emprego de técnicas de aprendizagem de máquina nesta área pode ser bastante útil para a definição de perfis de usuários que não tenham conhecimento de boas práticas de segurança;
- Agentes automatizados tais como, *worms*, vírus ou *trojans*, podem ser detectados com base em padrões de tráfego de rede de saída. As técnicas de aprendizagem de máquina são conhecidas por apresentarem bons desempenhos no reconhecimento de padrões;

Tabela 2.9. Síntese das técnicas de AM empregadas em detecção de anomalias.

Aplicação	Publicação	Técnica de AM														
		SVM	***k*-means**	**Redes Neurais**	***k*-NN**	**Árvores de Decisão**	***Naive Bayes***	**RIPPER**	**Floresta Aleatória**	**HMM**	***AdaBoost***	**Regressão Logística**	***Bagging***	**Conj. de Classif.**	**BART**	***Clustering***
Cross Site Scripting	[Likarish et al., 2009]	**X**				**X**	**X**	**X**								
	[Riech et al., 2010]	**X**														
Phishing	[Fette et al., 2007]								**X**							
	[Abu-Nimeh et al., 2007]	**X**		**X**		**X**			**X**			**X**			**X**	
	[Chandrasekaran et al., 2006]	**X**														
	[Sanglerdsinlapachai., 2010]	**X**		**X**		**X**	**X**		**X**		**X**					
	[Miyamoto et al., 2009]	**X**		**X**		**X**	**X**		**X**		**X**				**X**	
Negação de Serviço	[Gavrilis, 2005]			**X**												
	[Nguyen e Choin, 2010]									**X**						
	[Xu et al.,2007]					**X**										
	[kumar, 2011]			**X**												
	[Wu et al., 2011]					**X**										
Códigos Maliciosos *(Malware)*	[Schultz et al., 2001]						**X**	**X**								
	[Rieck et al., 2008]	**X**														
	[Ye et al, 2007]	**X**				**X**	**X**									
	[Wang et al, 2003]					**X**	**X**									
	[Kolter e Mallof, 2006]	**X**			**X**	**X**	**X**									
Spam	[Blanzieri e Bryl, 2007]	**X**			**X**									**X**		
	[DeBarr, 2009]								**X**							
	[Markines et al., 2009]													**X**		
	[Castillo et. al., 2007]					**X**							**X**			**X**
	[Wu, 2009]			**X**												
Detecção de Intrusão	[Horng et al., 2011]	**X**														**X**
	[Khan et al., 2007]	**X**				**X**										**X**
	[Mafra et al, 2008]	**X**		**X**												
	[Tsai e Lin, 2010]		**X**		**X**											
	[Xiao et al., 2007]	**X**												**X**		

- Pode ser difícil classificar falhas na rede como intencional ou não. Por exemplo, um aumento repentino no consumo de largura de banda da rede pode indicar um ataque de negação de serviços, ou simplesmente um evento onde muitos usuários simultaneamente acessam um servidor devido a algum evento popular (por exemplo, um *flash crowd*). Pode ser possível diferenciá-los através do emprego de classificadores automatizados;
- Outro desafio mostrado por técnicas de detecção de anomalia em um domínio de grande tráfego de dados como a Internet é a natureza das anomalias que muda ao longo do tempo, como forma de iludir as soluções de detecção de intrusão, conforme [Chandola et al., 2009]. Nessa situação, os dados têm um aspecto temporal associado a ele e relevante para aplicação de detecção de padrão com aprendizagem de máquina.
- É importante salientar ao leitor, neste ponto do minicurso, que a inclusão de aprendizagem de máquina em um sistema deve ser feito com cuidado para evitar que o componente de aprendizagem em si ceda ao ataque. A atual literatura mostra que os atacantes podem atacar com sucesso máquina de aprendizagem dos sistemas, tanto no geral [Barreno et al., 2006] como em domínios de aplicação específicos como geração de uma assinatura automática [Chung e Mok, 2006] [Chung e Mok, 2007], sistemas de detecção de intrusão [Fogla e Lee, 2006] e e-mail com filtros *spam*. É necessario assegurar que a aprendizagem seja bem sucedida, apesar de tais ataques, em outras palavras, atingir um aprendizado seguro [Barreno et al., 2008].

A maioria dos trabalhos de aprendizagem de máquina na área de segurança tem se concentrado no processo de desenvolvimento de técnicas para detecção de ataques. Esta é apenas uma pequena parte do escopo que envolve aspectos de segurança. As idéias acima salientam alguns problemas de segurança que parecem demandar mais investigação científica. Há certamente, porém, muitos outros aspectos em que as abordagens de aprendizagem de máquina podem trazer novas melhorias na segurança de computadores.

Referências

[Abu-Nimeh et al., 2007] Abu-Nimeh, S., Nappa, D., Wang, X., Nair, S. (2007) "A Comparison of Machine Learning Techniques for Phishing detection", Em *Proceedings of the Anti-phishing Working Groups 2nd annual eCrime Researchers Summit* (*eCrime '07*), pp. 60-69.

[Abu-Nimeh et al., 2009] Abu-Nimeh S., Nappa Dario, Wang X., Nair S., (2009) "Distributed Phishing Detection by Applying Variable Selection using Bayesian Additive Regression Trees", Em *IEEE International Conference on Communications (ICC 2009)*. pp. 1-5.

[Alpaydim, 2010] Alpaydim, E. (2010) "Introduction to Machine Learning", *The MIT Press*, Cambridge, Massachusetts, EUA, 537 páginas.

[Anderson, 1995] Anderson, J. (1995). "An Introduction to Neural Networks", Cambridge: *MIT Press*, 650 páginas.

[Anagnostakis et al., 2005] Anagnostakis, K. G., Sidiroglou, S., Akritidis, P., Xinidis, K., Markatos, E., e Keromytis, A. D. (2005). "Detecting Targeted Attacks Using Shadow Honeypots", Em *Proceedins of 14thUSENIX Security Symposium*, pp. 9-9.

[APWG, 2010] Anti-Phishing Working Group. (2010) "Phishing Activity Trends ReportQ1 2010", Disponível em: http://www.antiphishing.org/reports/apwg_report_Q1_2010.pdf.

[Barnett e Lewis, 1978] Barnett, V. e Lewis, T. (1978) "Outliers in Statistical Data*".Wiley Series in Probability & Statistics*, *John Wiley and Sons*, 584 páginas.

[Barreno et al., 2006] Barreno, M., Nelson, B., Sears, R., Joseph, A. D., Tygar, J. D. (2006). "Can machine learning be secure?" Em *Proceedings of the ACM Symposium on InformAtion, Computer, and Communications Security (ASIACCS'06)*.

[Barreno et al., 2008] Barreno, M., Nelson, B., Bartlett, P. L., Chi, F. J., Rubinstein, B. I. P., Saini, U., Joseph, A. D., Tygar, J.D. (2008). "Open Problems in the Security of Leaning", Em *Proceedings of the 1st ACM Workshop on AISec - AISec '08*

[Bennett et al., 2007] Bennett, J., Lanning, S. e Netflix, N. (2007) "The Netflix Prize". Em *Proceeding of KDD Cup and Workshop*.

[Benzel et al., 2006] Benzel, T., Braden, R., Kim, D., Neuman, C., Joseph, A. and Sklower, K. (2006) "Experience with DETER: a Testbed for Security Research", Em *Proceedings of the 2nd International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities (TRIDENTCOM'06),* pp. 379-388.

[Blanzieri e Bryl, 2007] Blanzieri, E., Bryl, A. (2007). "Evaluation of the Highest Probability SVM Nearest Neighbor Classifier with Variable Relative Error Cost", Em *Proceedings of Fourth Conference on email and Anti-Spam (CEAS'2007)*.

[Bradley, 1997] Brandley P. A., (1997) "The Use of the Under the ROC Curve in the Evaluation of Machine Learning Algorithms". *Pattern Recognition*, Vol. 30 (7), pp. 1145-1159

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., Stone, C. J. (1984). "Classification and Regression Trees", Monterey, *CA, Wadsworth & Brooks*.

[Castillo et al., 2007] Castillo, C., Donato, D., Gionis, A., Murdock, V., Silvestri, F. (2007), "Know Your Neighbors: Web Spam Detection Using the Web Topology". Em *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pp. 423-430.

[Ceron et al., 2009] Ceron, J. M., Granville, L., Tarouco, L. "Taxonomia de Malwares: Uma Avaliac¸ao dos Malwares Automaticamente Propagados na Rede" *Anais do IX Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais.*

[CERT.br, 2011] CERT.br – Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil. (2011) *Estatísticas do CERT.br*. http://www.cert.br/stats/incidentes/

[Chandola et al., 2009] Chandola, V., Banerjee, A., Kumar, V. (2009), "Anomaly Detection : A Survey", *ACM Computing Surveys*, pp. 1-72.

[Chandrasekaran et al., 2006] Chandrasekaran, M., Narayanan, K., e Upadhyaya, S. (2006), "Phishing Email Detection Based on Structural Properties", Em *NYS Cyber Security Conference*.

[Chipman et al, 2006] Chipman, H. A., George, E. I., McCulloch, R. E. (2006) "BART: Bayesian Additive Regression Trees", *Journal of the Royal Statistical Society*, Vol. 4 (1), pp. 266-298.

[Chung e Mok, 2006] Chung, S., Mok, A. K. (2006), "Allergy Attack Against Automatic Signature Generation", Em *Recent Advances in Intrusion Detection*, pp. 61–80.

[Chung e Mok, 2007] Chung, S., Mok, A. K. (2007). "Advanced Allergy Attacks: Does a Corpus Really Help?", Em *Recent Advances in Intrusion Detection*, pp. 236–255.

[Csurka et al., 2004] Csurka, G., Dance, C., Fan, L., Williamowski, J., Bray, C. (2004) "Visual Categorization with Bags of Keypoints," Em *ECCV04 Workshop on Statistical Learning in Computer Vision*, pp. 59–74

[Debarr e Wechsler, 2009] Debarr, D., Wechsler, H. (2009) "Spam Detection Using Clustering, Random Forests, and Active Learning", Em *Sixth Conference on Email and Anti- Spam (CEAS 2009)*.

[Denning, 1987] Denning, D. E. (1987) "An Intrusion-Detection Model".*IEEE Transactions on Software Engineering*, Vol. 13 (2), pp. 222–232.

[Duda et al. 2001] Duda, R. O., Hart, P. E. e Stork, D. G. (2001) *Pattern Classification*.2ª. Edição. *Wiley Interscience*, 680 páginas.

[Feitosa et al., 2008] Feitosa, E. L., Souto, E. J. P., Sadok, D. (2008) "Tráfego Internet não Desejado: Conceitos, Caracterização e Soluções*",Livro-Texto dos Minicursos do VIII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, páginas 91-137.

[Fette et al, 2007] Fette, N., Sadeh, Tomasic, A. (2007) "Learning to Detect Phishing E-mails", Em *Proceedings of the 16th International Conference on World Wide Web (WWW 07)*, páginas 649–656.

[Fogla e Lee, 2006] Fogla, P. e Lee, W. (2006) "Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques", Em *Proceeding of ACM Conference on Computer and Communications Security*.

[Fraser, 1998] Fraser, N. (1998) "*Neural Network Follies"*. Disponível em: http://neil.fraser.name/writing/tank.

[Gavrilis e Dematas, 2004] Gavrilis, D., Dermatas, E. (2004) "Real-time Detection of Distributed Denial-of-Service Attacks Using RBF Networks and Statistical Features", *Computer Networks.* Vol. 48, (2), páginas 235-245

[Gates e Taylor, 2007] Gates, C., Taylor, C. (2007) "Challenging the Anomaly Detection Paradigm: A Provocative Discussion," Em *Proc. Workshop on New Security Paradigms.*

[Grossman et al., 2007] Grossman, J., Hansen, R., Petkov, D.P., Rager, A., Fogie, S. (2007) "Cross Site Scripting Attacks: XSS Exploits and Defense". Burlington, MA, EUA, *Syngress Publishing Inc*, 482 páginas.

[Gu et al., 2007] Gu, G., Porras, P., Yegneswaran, V., Fong, M. e Lee, W. (2007) "Bot Hunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation", Em *Proceedings of 16thUSENIX Security Symposium,* 12 páginas.

[Haykin, 2008] Haykin, S. (2008) "Neural Networks and Learning Machines" 3rd Edition. *Prentice Hall.*

[Horng et al., 2011] Horng, S., Su, M., Chen, Y., Kao, T, Chen, R., Lai, J., Perkasa, C. (2011) "A Novel Intrusion Detection System Based on Hierarchical Clustering and

Support Vector Machines", *International Journal on Expert System with Applications*, Vol. 38 (1), pp. 306-313.

[Jain et al., 2000] Jain, A. K., Duin, R. P. W., Jianchang, M. (2000), "Statistical Pattern Recognition: A Review", *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Vol. 22 (1), pp. 4-37.

[Khan et al., 2007] Khan, L., Awad, M., Thuraisingham, B. (2007), "A New Intrusion Detection System Using Support Vector Machines and Hierarchical Clustering"**,** Em *International Journal on Very Large Data Bases*. Vol. 16(4), pp. 507-521.

[Kolter e Maloof, 2004] Kolter, J., Maloof, M. (2004)."Learning to Detect Malicious Executables in the Wild". Em *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD)*, pp. 470–478.

[Kolter e Maloof, 2006] Kolter, J. Z., Maloof, M. (2006), "Learning to Detect and Classify Malicious Executables in the Wild" Em *Journal of the Machine Learning*. Vol. 7, pp. 2721-2744.

[Kumar e Selvakumar, 2011] Kumar, P. A. R., Selvakumar, S. (2011). "Distributed Denial of Service Attack Detection Using an Ensemble of Neural Classifier", *Computer Communication*, Vol. 34, pp. 1328-1341.

[Kuncheva, 2004] Kuncheva, L.K. (2004), "Combining Pattern Classifiers – Methods and Algorithms".*Wiley-Interscience.*

[Kuncheva e Hoare, 2008] Kuncheva, L.I., Hoare, Z.S.J (2008), "Error-Dependency Relationships for the Naive Bayes Classifier with Binary Features", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 30(4), pp. 735-740.

[Lam e Yeung, 2007] Lam, H., Yeung, D. (2007). "A Learning Approach to Spam Detection based on Social Networks". Em *4th Conference on Email and AntiSpam.*

[Likarish et al., 2009] Likarish, P., Jung, E., Jo, I. (2009), "Obfuscated Malicious Javascript Detection using Classification Techniques", Em *4th IEEE International Conference on Malicious and Unwanted Software (MALWARE)*.

[Linden et al., 2003] Linden, G., Smith, B. e York, J. (2003) "Amazon.com Recommendations: Item-to-Item Collaborative Filtering". *IEEE Internet Computing*, Vol. 7 (1), pp 76–80.

[MacQueen, 1967] MacQueen, J. B. (1967). "Some Methods for Classification and Analysis of Multivariate Observations", Em *Proceedings of the Fifth Symposium on Math, Statistics, and Probability,* pp. 281–297.

[Mafra et al., 2008] Mafra, M. P., Fraga, S. J., Moll, V., Santin, O. A. (2008), "POLVO-IIDS: Um Sistema de Detecção de Intrusão Inteligente Baseado em Anomalias". Em *VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*.

[Markines et al., 2009] Markines, B., Catutto, C., e Menczer, F., (2009), "Social Spam Detection". Em *Proceedings of the 5th International Workshop on Adversarial Information Retrieval on the Web*, pp. 41-48.

[Mena, 2003] Mena, J. (2003) "Investigative Data Mining for Security and Criminal Detection*"*. *Butterworth Heinemann*, New York, NY.

[Miyamoto et al., 2009] Miyamoto, D., Hazeyama, H., Kadobayashi, Y. (2009), "An Evaluation of Machine Learning-Based Methods for Detection of Phishing Sites"

Em *Proceedings of the 15th International Conference on Advances in Neuro-Information Processing*, pp. 539-546.

[Mukkamala et al., 2002] Mukkamala, S., Janowski, G., Sung, A. H., (2002) "Intrusion Detection Using Neural Networks and Support Vector Machines", *Proceedings of the International Joint Conference on Neural Networks*. pp. 1702-1707.

[Nelson, 2010] Nelson, B. A. (2010) "Behavior of Machine Learning Algorithms in Adversarial Environments". *PhD. Thesis*, University of California, Berkeley, 244 páginas.

[Nguyen and Armitage, 2008] Nguyen, T. T. T., Armitage, G. (2008). "A Survey of Techniques for Internet Traffic Classification using Machine Learning". *IEEE Communication Surveys and Tutorials.*Vol. 10 (4), pp. 56-76.

[Nguyen e Choi, 2010] Nguyen, H-V., Choi, Y. (2010), "Proactive Detection of DDoS Attacks Utilizing *k*-NN Classifier in an Anti-DDoS Framework", *International Journal of Electrical and Electronics Engineering*, Vol. 4 (4), pp. 247-252.

[Owasp, 2010] OWASP, The Open Web Security Project (2010) "Cross-site Scripting (XSS)", Disponível em https://www.owasp.org/index.php/Cross-site_Scripting_%28XSS%29.

[Peddabachigari et al., 2007] Peddabachigari, S., Abraham, A., Grosan, C., Thomas, J. (2007), "Modeling Intrusion Detection System Using Hybrid Intelligent Systems", *Journal of Network and Computer Applications,* vol. 30, pp. 114-132.

[Phoha, 2002] Phoha, V. V. (2002). "Springer Internet Security Dictionary", *Springer-Verlag*, 320 páginas.

[Quinlan, 1993] Quinlan, J. R. (1993) "C4.5, Programs for machine learning". *Morgan Kaufmann,* San Mateo, Ca.

[Rabiner, 1989] Rabiner, L.R. (1989) "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings of the IEEE* , Vol. 77 (2), pp. 257-286.

[Rich e Knight, 1991] Rich, E. Knight, K. (1991). "Artificial Intelligence", *McGraw-Hill.*

[Richardson, 2010] Richardson, R. (2010) CSI/FBI Computer Crime Survey. Em *15th Annual 2010/2011 Computer Crime and Security*, 44 páginas.

[Riech et al., 2010] Rieck, K., Krueger, T., Dewald, A. (2010), "Cujo: Effcient Detection and Prevention of Drive-by-Download Attacks", Em *26thAnnual Computer Security Applications Conference 2010 (ACSAC 2010)*, pp. 31-39.

[Rieck et al., 2008] Rieck, K., Holz, T., Willems, C., Dussel, P., Laskov, P. (2008) "Learning and Classification of Malware Behavior", Em *Proceedings of the 5th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA '08)*, pp. 108-125.

[Rhodes et al, 2000] Rhodes, B., Mahaffey, J., Cannady, J. (2000). "Multiple Self-Organizing Maps for Intrusion Detection", Em *Proceedings of the 23rd National Information Systems Security Conference*.

[Saha, 2009] Saha, S. (2009). "Consideration Points: Detecting Cross-Site Scripting", *International Journal of Computer Science and Information Security (IJCSIS)*, Vol. 4, No. 1 & 2.

[Sanglerdsinlapachai e Rungsawang, 2010] Sanglerdsinlapachai, N., Rungsawang, A. (2010) "Web Phishing Detection Using Classifier Ensemble", Em *Proceedings of the 12th ACM International Conference on Information Integration and Web-based Applications & Services (iiWAS2010)*, pp. 210-215.

[Schultz et al., 2001] Schultz, M., Eskin, E., Zadok, F., Stolfo, S. (2001)."Data Mining Methods for Detection of New Malicious Executables", Em *Proceedings of the 22nd IEEE Symposium on Security and Privacy*, pp. 38–49.

[Sheng et al, 2009] Sheng, S., Wardman, B., Warner, G., Cranor, L. F., Hong, J., Zhang, C. (2009) "An Empirical Analysis of Phishing Blacklists", Em *Proceedings on Conference on Email and Anti-Spam (CEAS 09).*

[Smith, 2007] Smith, R. (2007) "An Overview of the Tesseract OCR Engine".Em *Proceedings of International Conference on Document Analysis and Recognition*.

[Sommer e Paxson, 2010] Sommer, R. e Paxson, V. (2010) "Outside the Closed World: On Using Machine Learning for Network Intrusion Detection". Em *Proceedings of the IEEE Symposium on Security and Privacy 2010,* pp. 305-316.

[Souza e Monteiro, 2009] Souza, E. P., Monteiro, J. A. S (2009), "Estudo Sobre Sistema de Detecção de Intrusão por Anomalias, uma Abordagem Utilizando Redes Neurais". Em *XIV Workshop de Gerência e Operação de Redes e Serviços - WGRS. Sociedade Brasileira de Redes de Computadores – SBRC.*

[Theodoridis e Koutroumbas, 2006] Theodoridis, S., Koutroumbas, K. (2006). "Pattern Recognition", 3rd Edition. *Academic Press*.837 páginas.

[Tian e Jianwen, 2009] Tian, L., Jianwen, W. (2009), "Research on Network Intrusion Detection System Based on Improved K-means Clustering Algorithm".Em *Internacional Forum on Computer Science – Technology and Applications (IFCSTA 2009). IEEE Computer Science,* Vol. 1, pp. 76-79.

[Tygar, 2011] Tygar, J.D. (2011) "Adversarial Machine Learning". *IEEE Internet Computing*, Vol. 15 (5), pp. 4-6.

[Tsai e Lin, 2010] Tsai, C., Lin, C. (2010). "A Triangle Area Based Nearest Neighbors Approach to Intrusion Detection".*Pattern Recognition,* Vol. 43, pp. 222-229.

[Tsai et al., 2009] Tsai, C., Hsu, Y., Lin, C., Lin, W. (2009), "Intrusion Detection by Machine Learning" *Expert Systems with Applications*, vol. 36, pp. 11994-12000.

[Vapnik, 1995] Vapnik, V. N. (1995) The Nature of Statistical Learning Theory. *Springer*, Berlin Heidelberg New York.

[Vincent, 2007] Vincent, L. (2007) "*Google Book Search: Document Understandingon a Massive Scale"*. Em *International Conference on Document Analysis and Recognition (ICAR 2007)*, pp. 819-823

[Xiang e Zhou, 2005] Xiang, Y., Zhou, W. (2005), "A Defense System against DDoS Attacks by Large-Scale IP Traceback", *Third International Conference on Information Technology and Applications (ICITA'05)*, pp. 431-436.

[Xia et al., 2010] Xia, D. X., Yang, S. H. e Li, C. G., (2010). "Intrusion Detection System Based on Principal Component Analysis and Grey Neural Networks". Em *2nd International Conference on Networks Security Wireless Communications and Trusted Computing*, pp. 142-145.

[Xiao et al., 2007] Xiao, H., Hong, F., Zhang, Z., Liao, J. (2007). "Intrusion Detection Using Ensemble of SVM Classifier". *Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FKSD 2007)*, pp. 45-49.

[Xu et al., 2007] Xu, X., Sun, Y., Huang, Z. (2007) "Defending DDoS Attacks Using Hidden Markov Models and Cooperative Reinforcement Learning", Em *(PAISI) Pacific Asia Workshop on Intelligence and Security Informatics*, pp. 196–207.

[Ye et al, 2007] Ye, Y., Wang, D., Li, T., Ye, D. (2007), "IMDS: Intelligent Malware Detection System". Em *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (KDD '07), pp. 1043-1047.

[Yue e Wang, 2009] Yue, C., Wang, H. (2009) "Charatering Insecure JavaScript Practice on the Web. Em *18th International Conference on the World Wide Web*.

[Wang et al, 2003] Wang, J-H., Deng, P.S., Fan, Y-S., Jaw, L-J., Liu, Y-C. (2003), "Virus Detection Using Data Mining Techniques", Em *Proceedings of the 37th International Conference on Security Technology*, pp. 71-76.

[Witten e Frank, 2000] Witten, I.H., Frank, E. (2000) "Data Mining: Practical Machine Learning tools and Techniques with Java Implementations*". Morgan Kaufmann*.

[Wu, 2009] Wu, S. (2009) "Behavior-based Spam Detection Using a Hybrid Method of Rule-based Techniques and Neural Networks", *Expert Systems with Applications*, Vol. 36 (3), pp. 4321-4330.

[Wu et al., 2011] Wu, Y-C., Tseng, H-R., Yang, W., Jan, R.H. (2011) "DDoS detection and traceback with decision tree and grey relational analysis*", International Journal Ad Hoc and Ubiquitous Computing*, Vol. 7 (2), pp.121–136.

[Zhang et. al., 1996] Zhang, T., Ramakrishnan, R., & Livny, M., (1996) "BIRCH: An Efficient Data Clustering Method for Very Large Databases", Em *Proceedings of the ACM SIGMOD.*

[Zhong e Yue, 2010] Zhong, R., Yue, G. (2010) "DDoS Detection System Based on Data Mining", *Em Proceedings of the Second International Symposium on Networking and Network Security (ISNNS '10)*, pp. 062-065.

View publication stats