Universidade Federal Fluminense
Instituto de Ciência e Tecnologia
Curso de Graduação em Ciência da Computação

Cláudio André da Silva Alves

AVALIAÇÃO DO APRENDIZADO ONLINE NA DETECÇÃO DE INVASÃO EM REDES SEM FIO

Rio das Ostras-RJ
2018

ClÁUDIO ANDRÉ DA SILVA ALVES

AVALIAÇÃO DO APRENDIZADO ONLINE NA DETECÇÃO DE INVASÃO EM REDES SEM FIO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração: Aprendizado de Máquina e Inteligencia Artificial.

Orientador: Prof. Dr. FLÁVIA CRISTINA BERNARDINI

Rio das Ostras-RJ

2018

CLÁUDIO ANDRÉ DA SILVA ALVES

AVALIAÇÃO DO APRENDIZADO ONLINE NA DETECÇÃO DE INVASÃO EM REDES SEM FIO

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração: Aprendizado de Máquina e Inteligencia Artificial.

Aprovada em julho de 2018.

BANCA EXAMINADORA

Prof. Dr. FLÁVIA CRISTINA BERNARDINI - Orientador
UFF

Prof. Dr. EDWIN BENITO MITACC MEZA
UFF

Prof. Dr. LEANDRO SOARES DE SOUSA
UFF

Rio das Ostras-RJ
2018

Dedico este trabalho, aos meus amados pais Maria e Ciraldo pelo amor, carinho e orientação desde os meus primeiros passos, e minha irmã Tamires pelo companheirismo, amizade e paciência nessa minha jornada.

Agradecimentos

Agradeço primeiramente à Deus e minha família que sempre me apoiaram e em todos os momentos se dispuseram à ajudar. Não haveria espaço suficiente para listar todos aqueles que me ajudaram nesse plano espiritual e fora dele, à eles deixo meu profundo muito obrigado.

Lembro claramente quando fui fazer minha inscrição e um funcionário disse que eu só sairia da UFF formado. Esse dia está se aproximando, pelo menos na graduação. Sou grato a todos os professores e funcionários da UFF que, nessa trajetória universitária, contribuíram para o meu crescimento profissional e pessoal. Agradeço também à Prof. Dra. Leila Weitzel e ao PIBIC/CNPQ pela oportunidade de ter sido aluno de Iniciação Científica e por despertar o interesse de seguir na carreira acadêmica.

À Prof. Dra. Flávia Bernardini agradeço pela orientação, amizade, confiança, e por contribuir para minha formação pessoal e humana. Sou eternamente grato.

Não poderia encerrar meus agradecimentos sem citar as amizades que eu fiz durante esse período, e como minha última fala do agradecimento eu digo: ”Respeita a MULEKADINHA!”.

"História, nossas histórias
Dias de luta, dias de glória"
(Charlie Brown Jr., 2005)

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

Garantir a proteção em redes de computadores é uma tarefa cada vez mais difícil devido ao enorme número e variabilidade das ameaças encontradas atualmente. *Intrusion Detection System* (IDS) ou sistemas de detecção de invasão são utilizados para assegurar a segurança da informação, em redes de computadores, para qualquer conteúdo que tenha valor para uma pessoa ou empresa. Sistemas de detecção de invasão monitoram computadores ou redes de computadores buscando identificar atividades maliciosas ou acessos não autorizados. Um sistema de detecção de invasão baseado em rede realiza a detecção, capturando os pacotes da rede, analisando os cabeçalhos, carga útil e compara com padrões ou assinaturas conhecidas. A base de dados utilizada é proveniente de uma publicação acadêmica onde o tráfego de rede de um laboratório experimental foi capturado e rotulado como normal ou anormal.

O amadurecimento da internet das coisas (Internet of Things - IoT), exigirá soluções cada vez mais eficientes para comportar o crescente número de dispositivos conectados. Nesse estudo foi avaliado como algoritmos *online* podem auxiliar os sistemas de detecção de invasão, que analisam o tráfego da rede que utilizam a técnica de detecção baseada em anomalia, onde basicamente os IDS classificam o tráfego de rede como normal ou anormal. O aprendizado *online* é caracterizado pela contínua atualização do classificador ao contrário do aprendizado de máquina em lote onde a fase de treinamento, isto é, criação do classificador é realizada uma vez.

Além de estudar a aplicabilidade do aprendizado *online* em sistemas de detecção de invasão, objetiva-se averiguar se em relação ao tempo de processamento, os algoritmos *online* são mais eficientes do que os algoritmos em lote. A aplicação do aprendizado *online*, resultou em uma diminuição considerável, cerca de 46%, no tempo necessário para concluir o processo de treinamento. Essa redução no tempo de treinamento, praticamente não impactou os resultados obtidos em comparação com a publicação original que disponibilizou a base de dados.

Palavras-chave: Aprendizado Online, Sistemas de Detecção de Invasão, Redes Wi-fi

Abstract

Ensuring protection in computer networks is an increasingly difficult task because of the sheer number and variability of threats currently encountered. Intrusion Detection Systems (IDS) is usually used to ensure the security of information, including any contenet that has value to a person or company, in computer networks are intrusion detection systems.

Intrusion Detection Systems monitor computers or computer networks to identify malicious activity or unauthorized acess. A network based intrusion detection system performs detection by capturing network packets, analyzing headers, payload field and compares with standards or signatures known. The used database comes from ana academic publication, where network traffic from an experimental laboratory was captured and labeled as normal or abnormal.

The maturing of the Internet of Things (IoT) will require efficient solutions to accommodate the growing number of connected devices. In this study we evaluated how online learning algorithms can help the intrusion detection systems that analyze the network traffic using the anomaly based detection technique, where IDS basically classifies the network traffic as normal or abnormal. Online learning is characterized by continuous upgrading of the classifier as opposed to batch machine learning where the training phase, i.e. classifier creation is performed once.

In addition to studying the applicability of online learning in intrusion detection systems, we aim to investigate whether in terms of processing time, online algorithms are more efficient than batch algorithms. The application of online learning has resulted in a considerable decrease, about 46 %, in the time required to complete the training process. This reduction in training time practically did not impact the results obtained in comparison with the original publication that made available the database.

Keywords: Online Learning, Intrusion Detection System (IDS), Wireless Networks

Capítulo 1

Introdução

A facilidade de comunicação disponibilizadas pelas redes IEEE 802.11 ou redes sem fio trouxe consigo uma enorme gama de ataques associados à não existência de uma conexão física. Esses ataques acometem os princípios fundamentais da segurança da informação: integridade, confidencialidade e disponibilidade. Essas ameaças podem comprometer permanentemente a atividade de uma empresa, como aconteceu com três *startups* que encerraram suas atividades devido a falhas de segurança em seus produtos [1], ou até mesmo afetar pessoas, como aconteceu com os usuários do *Yahoo!*, que tiveram seus dados expostos [2]. A identificação desses ataques é de vital importância para que as pessoas responsáveis possam tomar medidas apropriadas ao passo que cada vez mais as redes de computadores, em especial a internet, estão inseridas no uso cotidiano.

Um mecanismo útil que compõe o leque de ferramentas de segurança de redes são os *Intrusion Detection System* (IDS), ou sistemas de detecção de invasão, que monitoram os sistemas operacionais ou redes de computadores em busca de ações potencialmente prejudiciais [1]. É importante ressaltar que. os sistemas de detecção de invasão não substituem o uso de *firewalls*, sendo considerados como mais uma ferramenta criada para garantir prioritariamente a segurança em ambientes corporativos [39].

O objetivo principal desse trabalho é avaliar como o aprendizado *online* pode auxiliar os sistemas de detecção de invasão que analisam o tráfego da rede sejam os NIDS (*Network Intrusion Detection Systems*) ou WIDS (*Wireless Intrusion Detection Systems*), que utilizam a técnica de detecção baseada em anomalia. Destacam-se como objetivos secundários: confirmar através dos experimentos, se em relação ao tempo de processamento, o aprendizado *online* é mais eficiente do que o aprendizado em lote; aperfeiçoar o conhecimento técnico por meio das diversas implementações necessárias para executar os experimentos e, por último mas não menos importante, enriquecer o conhecimento pessoal sobre o aprendizado de máquina e assuntos correlatos.

A tarefa de identificação de um ataque em uma rede de computadores pode ser modelada como um problema de classificação. Nessa abordagem, as classes de saída são previamente conhecidas e cada registro deve ser classificado de acordo com uma classe. Nesse trabalho, foi utilizado o aprendizado de máquina supervisionado para tentar classificar os registros do tráfego de rede.

[1] Disponível em: https://prooncall.com/3-companies-went-business-due-security-breach/

[2] Disponível em: http://mashable.com/2016/09/22/yahoo-confirms-data-breach/#aC_BirBvIOqw

O aspecto iterativo do aprendizado de máquina é essencial para, na medida que novos ataques forem apresentados, o classificador possa ser retreinado, e num momento futuro, consiga classificar esses ataques. Esse comportamento fica bastante evidente no aprendizado *online*.

Capítulo 2

Referencial Teórico e Revisão da Literatura

2.1 Aprendizado de Máquina

Como um campo da inteligência artificial, o aprendizado de máquina ou *machine learning* herda o objetivo de criar sistemas que sejam capazes de imitar o comportamento inteligente, realizando tarefas que normalmente são triviais para os humanos como tomada de decisão ou reconhecimento de padrões. Uma definição bastante utilizada de aprendizado de máquina é a de Tom Mitchel [46] no qual um programa de computador aprende com a experiência E em relação a alguma classe de tarefas T e medida de desempenho P se o desempenho em tarefas em T, conforme medido por P, melhora com a experiência E.[1]

No aprendizado de máquina os computadores são capazes de aprender sem serem explicitamente programados [55], diferentemente do que ocorre em um algoritmo convencional, onde regras são estritamente definidas. Por utilizar o conhecimento adquirido para realizar a tomada de decisões, os algoritmos de aprendizado de máquina necessitam de experiências representativas para que sua performance ao realizar essas tarefas possa ser aprimorada [30].

Apesar de não ser uma área nova, ela vem se destacando para o público geral por meio de inúmeras aplicações comerciais que fazem uso intensivo do aprendizado de máquina. Alguns exemplos de aplicação de aprendizado de máquina são serviços cognitivos de visão computacional para realizar o reconhecimento de objetos e identificação de emoções em imagens; reconhecimento de fala em assistentes pessoais; sugestão automática e detecção de idioma nos buscadores, dentre outros. Essa vasta utilização é devida à grande capacidade de generalização que os algoritmos apresentam [18]. Esse crescimento foi impulsionado por alguns fatores, como a crescente produção e disponibilização de dados, aumento do poder computacional, e por último, mas não menos importante, a criação e aperfeiçoamento dos algoritmos [66].

O processo genérico de aprendizado de máquina pode ser separado em algumas etapas [22], são

[1] "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E."

elas: coleta dos dados, pré-processamento, treinamento e teste. Na fase de coleta de dados, como o nome sugere, a base de dados relativa ao domínio é criada ou coletada. A segunda fase é o pré-processamento onde ocorre toda a preparação e manipulação dos dados. Algumas atividades pertencentes a essa fase são: normalização, redução de dimensões e transformação. Ainda nessa fase, o conjunto de dados é dividido em duas partes, chamadas de conjunto de treinamento ou *training set* e conjunto de teste ou *test set*. O conjunto de dados é dividido para fornecer dados novos ao classificador quando este for validado, para avaliar se houve o *overfitting*, que é um super ajuste aos dados, que dificulta a generalização para registros nunca vistos anteriormente [2], [50]. Nas fases seguintes, o classificador é treinado com o *training set* na fase de treinamento, e validado com o *test set* na fase de teste. Esse ciclo de treinamento e teste pode ser repetido diversas vezes, até mesmo quando o classificador está sendo utilizado em produção. Esse aspecto iterativo do aprendizado de máquina é essencial para que os modelos possam se adaptar com a chegada de novos dados de treinamento, comportamento este que fica evidente no aprendizado online que será abordado adiante [57].

Os paradigmas primários dos métodos de aprendizado de máquina em relação aos algoritmos são: aprendizado supervisionado e aprendizado não supervisionado [6], [22]. No primeiro, aprendizado supervisionado, os algoritmos fazem a predição de eventos futuros baseado nos dados conhecidos. Para cada exemplo no conjunto de treinamento, existe um rótulo associado a cada instância definindo o valor de interesse que pode ser discreto ou contínuo. Com esse rótulo, é possível calcular um erro na predição, e aprimorar o modelo para futuramente fazer o máximo de predições corretas com dados futuros. O aprendizado não supervisionado tem um caráter exploratório, ou seja, não existem rótulos associados aos dados, dessa forma, o objetivo de um algoritmo pertencente a esse grupo é organizar os dados para que seja possível descrever sua estrutura. Um exemplo é a clusterização de segmentos de mercado [31].

Em termos gerais, o objetivo de um problema de classificação supervisionado é minimizar uma função $J(\theta)$ utilizando uma função $h(\theta)$, também chamada de hipótese, que, ao receber uma entrada (x_i, y_i) produzirá um resultado $h(x_i)$. Utilizando uma função de custo $C(h(x_i), y_i)$, é possível avaliar o desempenho da classificação e atualizar θ. Por exemplo, pode-se utilizar a técnica de otimização *gradient descent* como função de custo. Sem perda de generalidade, para exemplificar esse processo, considere o problema de classificação de e-mail como spam ou não. Nosso *training set* é composto de m registros, cada registro é da forma (x_i, y_i) de $i = 1$ até m. Essa notação pode ser simplificada ao escrever na forma matricial, assim a matriz de atributos X onde cada linha representa uma instância e cada coluna um atributo e Y é um vetor que contém os rótulos de cada instância. Uma hipótese muito utilizada em problemas de classificação binário é a função sigmóide, que tem como imagem o intervalo $[0, 1]$. Dessa maneira, pode-se definir um limite, por exemplo de 0.7. Se o resultado da função sigmóide aplicada à uma instância for superior ou igual a esse limite ele é classificado como *spam*, e como normal caso contrário. O fluxo de execução tem os seguintes passos e pode ser repetido inúmeras vezes visando encontrar os valores de θ que minimizam a função $J(\theta)$ e consequentemente o erro.

1. Aplicação das funções de hipótese e de custo para cada instância do *training set*

$$h(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2.1}$$

$$C(h(x), y) = \begin{cases} -log(h(x)) & \text{se } y = 1 \\ -log(1 - h(x)) & \text{se } y = 0 \end{cases} \tag{2.2}$$

2. Cálculo da função $J(\theta)$

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} C(x_i, y_i) \tag{2.3}$$

3. Atualização dos parâmetros θ

Esse fluxo de execução pode ser representado de forma algorítmica e, sem perda de generalidade, a simulação será limitada à z iterações.

Dados: instâncias, rótulos

Entrada: X, Y

```
Inicialização do classificador;
Melhor Resultado = Infinito;
para k=1 até z faça
    para i = 1 até m faça
        Aplicar a função de hipótese e predizer a classe ŷ_i baseado em x_i;
        Receber a classe y_i verdadeira;
        Calcular a função de custo C(ŷi, y_i);
    fim
    Calcula função J(θ);
    se J(θ) < Melhor Resultado então
        Melhor Resultado = J(θ);
        Atualiza θ;
    fim
fim
```

Algoritmo 1: Algoritmo genérico de aprendizado em *batch* (lote) para problemas de classificação

Ao final das z iterações, $J(\theta)$ será mínimo se o classificador convergir, ou seja, os valores ótimos de θ serão encontrados. Quando esses valores forem encontrados para aquele conjunto de instâncias, o classificador pode ser validado com o *test set*. Em um problema binário de classificação, existe uma classe negativa e uma positiva, que geralmente é a classe de interesse. Existem quatro situações possíveis ao rotular uma instância:

- *True positive(TP)*: caso em que o classificador atribui o rótulo positivo corretamente;
- *False positive(FP)*: caso em que o classificador atribui o rótulo positivo incorretamente;
- *True negative(TN)*: caso em que o classificador atribui o rótulo negativo corretamente;
- *False negative(FN)*: caso em que o classificador erroneamente atribui o rótulo negativo.

Nessa validação, o desempenho dos classificadores é realizado através de métricas que informam a corretude dos classificadores [61], as métricas utilizadas foram as seguintes: precisão, corresponde à razão entre o número de positivos corretamente classificados e o total de exemplos rotulados como positivo (Eq. 2.4); *recall*, equivale à taxa de positivos corretamente classificados e o número total de exemplos positivos (Eq. 2.5); taxa de falso positivo, é o número de exemplos negativos classificados como positivos (Eq. 2.6); *F-measure* ou *F1 score*, é uma métrica que relaciona a precisão e o *recall*, sua imagem varia no intervalo fechado $[0, 1]$ e quanto mais próximo de 1 melhor o resultado, tipicamente, o parâmetro β é igual a 1 (Eq. 2.7).

$$\text{Precisão} = \frac{TP}{TP + FP} \tag{2.4}$$

$$\textit{Recall} = \frac{TP}{TP + FN} \tag{2.5}$$

$$\text{Taxa de falso positivo} = \frac{FP}{TN + FP} \tag{2.6}$$

$$\textit{F-measure} = \frac{(\beta^2 + 1)\text{Precisão} \times \text{Recall}}{\beta^2\text{Precisão} + \text{Recall}} \tag{2.7}$$

2.2 Abordagem de Aprendizado Online

A abordagem clássica do aprendizado de máquina é chamada de aprendizado em *batch* ou em lote. Nessa configuração, todas as instâncias do conjunto de treinamento devem ser analisadas para que os parâmetros do classificador possam ser atualizados. O aprendizado *online* difere do aprendizado de máquina em lote porque, nessa abordagem, o classificador pode ser atualizado a cada dado de entrada e não depois do acesso a todo *training set*. Um dos motivos é porque os dados estão disponíveis em tempo real, e devem ser prontamente utilizados. Essa abordagem é útil em cenários onde não é possível acessar todo *training set*, seja por questões de validade dos dados, ou seja, existe uma dependência temporal ou até mesmo inviabilidade de armazenamento, devido ao grande fluxo de informações num curto período de tempo. Utilizando o exemplo genérico anterior representado no algoritmo 1, o vetor θ pode ser atualizado a cada registro diferente do que ocorria na abordagem clássica (lote), onde a atualização dos parâmetros só poderia ocorrer após o processamento de todas as instâncias do conjunto de treinamento.

No aprendizado *online*, quando há um *streaming* dos dados, não existe uma distinção clara das fases de treinamento e teste. Uma abordagem para validar o classificador consiste em utilizar a instância atual para teste e posteriormente essa mesma instância será utilizada para treinar o classificador. Apesar

de não requerer o retreino quando uma nova instância é apresentada, o aprendizado *online* não é capaz de aprender uma classe que não estava presente na inicialização do classificador. Para lidar com isso, a publicação de Venkatesan e Joo Er [65] apresenta o paradigma de aprendizado progressivo, onde o classificador é remodelado automaticamente ao encontrar uma nova classe.

Com o crescente aumento no volume e na geração de dados, é imprescindível que os classificadores apresentem um desempenho satisfatório e que sejam extremamente eficientes, para que seja viável a utilização do aprendizado de máquina para aplicações de tempo real. Em comparação com a abordagem clássica, o aprendizado *online* é computacionalmente menos custoso em termos de processamento e também no armazenamento. No algoritmo 2 são exibidos os passos de um classificador genérico de aprendizado *online* [69]. Para cada instância, o classificador atribui um rótulo e compara com a classe verdadeira. Se a predição foi incorreta seus parâmetros são atualizados para classificação da próxima instância.

Dados: instâncias, rótulos

Entrada: X, Y

```
Inicialização do classificador;
para i = 1 até m faça
    Predizer a classe ŷ_i baseado em x_i;
    Receber a classe y_i verdadeira;
    Calcular a função de custo C(ŷi, y_i);
    se C(ŷ_i, y_i) > threshold então
        Atualiza θ;
    fim
fim
```

Algoritmo 2: Algoritmo genérico de aprendizado *online* para problemas de classificação

No exemplo do algoritmo *online* de classificação, ficou implícito a forma como a atualização dos parâmetros ocorre. Agora, será exibido o método de otimização utilizado por alguns algoritmos presentes na LIBSOL, biblioteca utilizada nos experimentos, chamado de *gradient descent*. O gradiente de uma função F (∇F) é definido como um vetor de derivadas parciais onde cada componente possui a derivada parcial em relação à uma variável. Em um ponto P qualquer, o gradiente da função F aplicado ao ponto P, matematicamente temos $\nabla F(P)$, indica a direção em que a função F cresce, dessa maneira se "caminharmos" na direção oposta, a função F será minimizada e convergirá para a solução ideal, essa é a metodologia básica do algoritmo de otimização *gradient descent*.

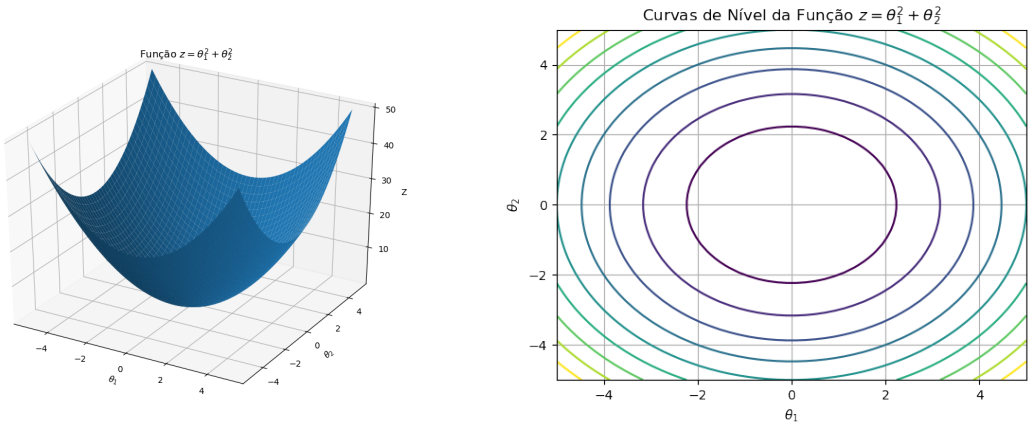
Ilustrando o funcionamento do algoritmo considere, sem perda de generalidade, uma função $F(\theta_1, \theta_2) = \theta_1^2 + \theta_2^2$ que queremos minimizar. Antes de iniciar o passo a passo do algoritmo, para ter uma representação bidimensional e simplificar a análise, observe o gráfico da função e suas curvas de nível na Figura 2.1. Do cálculo sabe-se que, com os testes da primeira e da segunda derivada é possível descobrir a declividade e a concavidade da função respectivamente em um determinado ponto. Assim ao fazer $\nabla F = 0$ será encontrado os pontos críticos da função e determinaremos se trata de um ponto

máximo ou mínimo. Após realizar os testes verifica-se que o ponto $(0, 0)$ é um ponto crítico e trata-se do mínimo global, vale destacar que nem sempre é trivial executar esses testes e eles não são necessários para a utilização do método, eles foram citados para a comprovar o resultado gerado pelo *gradient descent*.

Figura 2.1: Gráfico do parabolóide

(a) Gráfico da função

(b) Curvas de nível



Fonte: Elaborada pelo autor

Agora que o comportamento da função, os valores de θ que minimizam a função são conhecidos, o algoritmo pode ser iniciado. Considere por exemplo, inicialmente $\theta_1 = \theta_2 = 4$ e $\eta = 0.2$. O primeiro passo é calcular o gradiente da função F:

$$\begin{aligned} F(\theta_1, \theta_2) &= \theta_1^2 + \theta_2^2 \\ \nabla F &= \left\langle \frac{\partial F}{\partial \theta_1}, \frac{\partial F}{\partial \theta_2} \right\rangle = \langle 2\theta_1, 2\theta_2 \rangle \end{aligned} \tag{2.8}$$

Calculando o gradiente no ponto inicial, observa-se que $\nabla F(\theta_1, \theta_2) = (8, 8)$ como $\frac{\partial F}{\partial \theta_1}$ e $\frac{\partial F}{\partial \theta_2}$ são maiores que 0, pelo teste da primeira derivada que a função é crescente logo como deseja-se minimizar a função, é necessário decrementar os valores de θ como pode ser visto na Equação 2.9, por outro lado se nesse ponto a função fosse decrescente para θ_1 ou θ_2 implicaria que a derivada parcial de F em relação a θ_1 ou θ_2 seria negativa e o valor de θ deveria ser incrementado. Esse processo pode ser repetido infinitas vezes ou até que as derivadas parciais de F em relação a θ_1 e θ_2 fossem nulas.

$$\begin{aligned} \theta_1 &= \theta_1 - \eta \frac{\partial F}{\partial \theta_1} \\ \theta_2 &= \theta_2 - \eta \frac{\partial F}{\partial \theta_2} \end{aligned} \tag{2.9}$$

O parâmetro η é chamado de *learning rate* e define o quão drástica será a atualização dos parâmetros. Se η for muito pequeno os pesos podem ser mais precisos, porém o processo terá seu desempenho deteriorado e por outro lado se η for muito grande as atualizações dos pesos podem ser tão grandes e incorretas que os pesos ótimos podem não ser encontrados e o classificador pode não convergir e até

mesmo divergir [23]. A Tabela 2.1 contém os valores de θ bem como o valor de $F(\theta_1, \theta_2)$ numa execução limitada a 10 iterações. Percebe-se que θ_1 e θ_2 decrescem a cada iteração e, consequentemente o valor da função F também diminui e alcança o valor mínimo quando θ_1 e θ_2 se aproximam de zero.

Iteração	θ_1	θ_2	$F(\theta_1, \theta_2)$
0	4,000000	4,000000	32,000000
1	2,400000	2,400000	11,520000
2	1,440000	1,440000	4,147200
3	0,864000	0,864000	1,492992
4	0,518400	0,518400	0,537477
5	0,311040	0,311040	0,193492
6	0,186624	0,186624	0,069657
7	0,111974	0,111974	0,025077
8	0,067185	0,067185	0,009028
9	0,040311	0,040311	0,003250

Tabela 2.1: Primeiro exemplo de execução do *gradient descent*

Agora considere que o ponto inicial seja o ponto $\theta_1 = 4$ e $\theta_2 = -7$, ao executar o algoritmo os resultados da Tabela 2.2 seriam obtidos. Nesse exemplo, θ_1 segue o mesmo comportamento exibido no exemplo anterior e, o valor da função F se aproxima do valor mínimo quando θ_2 é incrementado e se aproxima de zero. Como visto anteriormente, o ponto (0,0) trata-se do mínimo global da função F.

Iteração	θ_1	θ_2	$F(\theta_1, \theta_2)$
0	4,000000	-7,000000	65,000000
1	2,400000	-4,200000	23,400000
2	1,440000	-2,520000	8,424000
3	0,864000	-1,512000	3,032640
4	0,518400	-0,907200	1,091750
5	0,311040	-0,544320	0,393030
6	0,186624	-0,326592	0,141491
7	0,111974	-0,195955	0,050937
8	0,067185	-0,117573	0,018337
9	0,040311	-0,070544	0,006601

Tabela 2.2: Segundo exemplo de execução do *gradient descent*

O *gradient descent* é um algoritmo de otimização relativamente simples e uma ótima abordagem inicial para começar a investigação de um problema de aprendizado de máquina. O método sempre converte para o mínimo local e no caso de funções convexas ele convergirá para a solução ótima nesse caso o mínimo global.

Um dois algoritmos mais antigos de aprendizado de máquina que utiliza a abordagem *online* é o Perceptron [54], que consiste nos primeiros experimentos e modelos de redes neurais artificias para desenvolver a visão computacional, sua representação pode ser encontrada na Figura 2.2. Para cada atributo, é definido inicialmente um peso θ, que deve ser multiplicado pela entrada. Em seguida, todas as entradas são somadas juntamente com um bias.

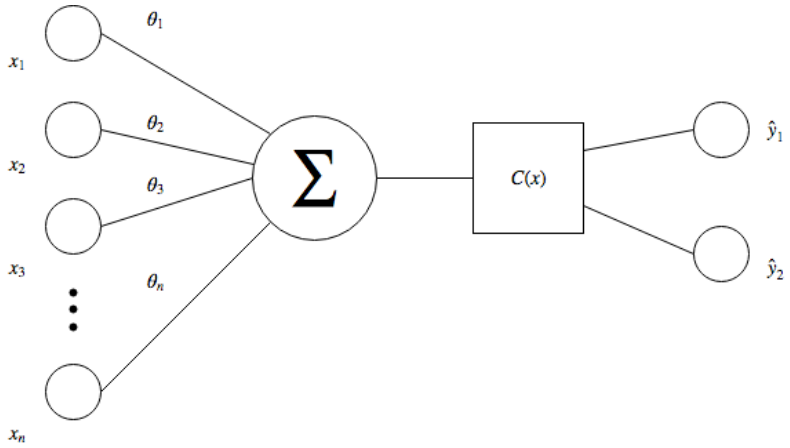
O perceptron é um classificador linear e utiliza a função sinal (Eq. 2.10) como hipótese, a função de custo (Eq. 2.11) ou como é comumente chamada no universo das redes neurais, a função de ativação que por sua vez é aplicada para que ocorra a atualização dos parâmetros (Eq. 2.12) que ocorre se $C(x) \neq 0$.

$$h(x) = \begin{cases} 1 & \text{se } \theta \cdot x + b > 0 \\ 0 & \text{caso contrário} \end{cases} \tag{2.10}$$

$$C(x) = \max(0, -y^i x^i \theta) \tag{2.11}$$

$$\theta = \theta + y_i x_i \tag{2.12}$$

Figura 2.2: Representação do Perceptron



Fonte: Elaborada pelo autor

2.2.1 Descrição da biblioteca LIBSOL

A LIBSOL [69] é uma biblioteca *open source* composta de algoritmos de aprendizado *online* escrita em C++ idealizada com foco na velocidade, escalabilidade, portabilidade e extensibilidade. O *design* modular utilizado permite que novos algoritmos ou funções de custo possam ser facilmente implementados utilizando um estilo de programação similar ao encontrado no MATLAB. Além da classificação binária também é oferecido suporte à problemas multi-classes, por meio da estratégia *"One-vs-all"*, onde um classificador binário é ajustado para cada classe. A instância recebe o rótulo da classe com melhor resultado comparando a saída de cada um dos classificadores gerados [25]. Os algoritmos implementados inicialmente na biblioteca são algoritmos lineares, ou seja, a hipótese é uma função polinomial de grau 1 na forma $f(x) = aX + b$. A modelagem de um problema definindo uma função de custo que deve ser

minimizada ou maximizada abre um leque de oportunidades para a utilização de diversas técnicas de otimização.

Resumindo o funcionamento dos algoritmos presentes na LIBSOL, a diferença entre eles se dá na forma como ocorre a atualização do classificador e quais estratégias de otimização são utilizadas, por exemplo utilizando o *gradient descent* e suas derivações, ou multiplicadores de Lagrange para encontrar os pontos extremos da função de custo. Por exemplo, o SOP [10] atualiza seus parâmetros quando erra a predição, já o AROW [16], também atualiza ao acertar sua predição de acordo com uma margem, ou seja, distância da instância atual para o limite de decisão, os algoritmos que possuem o prefixo "ADA", por sua vez, utilizam uma *learning rate* independente para cada parâmetro. Os algoritmos presentes na biblioteca podem ser separados em dois grupos: primeira ordem e segunda ordem, utilizando a matriz jacobiana ou matriz hessiana, respectivamente. Na Tabela 2.3 são listados os algoritmos disponibilizados pela biblioteca, separados pela ordem da matriz de derivadas parciais, ou seja, as matrizes jacobiana e hessiana. Todos os algoritmos disponibilizados na biblioteca foram utilizados para teste. O cálculo das métricas foi feito por meio da biblioteca Imbalanced-learn [38] que por sua vez utiliza as funcionalidades da Scikit-Learn [53], que é uma das mais populares bibliotecas voltadas ao aprendizado de máquina para a linguagem de programação Python [63].

Metodologia	**Nome**	**Descrição**
Primeira Ordem	Perceptron [54]	The Perceptron Algorithm
	OGD [71]	Online Gradient Descent
	PA [17]	Passive Agressive Algorithms
	ALMA [27]	Approximate Large Margin Algorithm
	RDA [70]	Regularized Dual Averaging
	STG [37]	Sparse Online Learning via Truncated Gradient
	FOBOS-L1 [21]	l1 Regularized Forward Backward Splitting
	RDA-L1 [70]	Mixed l_1/l_2^2 Regularized Dual Averaging
	ERDA-L1 [70]	Enhanced l_1/l_2^2 Regularized Dual Averaging
Segunda Ordem	SOP [10]	Second-Order Perceptron
	CW [19]	Confidence Weighted Learning
	ECCW [15]	Exactly Convex Confidence Weighted Learning
	AROW [16]	Adaptive Regularized Online Learning
	Ada-FOBOS [20]	Adaptive Gradient Descent
	Ada-RDA [70]	Adaptive Regularized Dual Averaging
	Ada-FOBOS-L1 [20]	Ada-FOBOS with l1 regularization
	Ada-RDA-L1 [20]	Ada-RDA with l1 regularization

Tabela 2.3: Algoritmos implementados na LIBSOL

2.3 Redes de Computadores

Redes de computadores tem uma definição bem intuitiva: dispositivos interligados, também chamados de *hosts* formam uma rede para o compartilhamento de dados ou recursos físicos [36]. O maior exemplo de uma rede de computadores é a Internet, que ao redor do mundo conecta milhões de dispositivos, desempenhando um papel sem precedentes para alcançar a globalização, e possibilitando a disseminação de conhecimento instantaneamente e sem grande esforço.

Para que os dispositivos dos mais variados tipos possam se comunicar perfeitamente, devem existir diretrizes para que todos participantes sigam e possam interagir [7]. Esse é o papel dos protocolos de rede, regras que definem como os participantes devem se comunicar definindo, por exemplo, a ordem e o formato das mensagens.

2.3.1 Pilha de protocolos TCP/IP

Dada a complexidade da comunicação de um sistema que conecta dispositivos não necessariamente com as mesmas especificações, as redes de computadores seguem um modelo estrutural para lidar com tal desafio. Esse modelo estrutural é chamado de arquitetura de camadas. A modularização permite atribuir funcionalidades específicas e bem definidas para cada camada [36]. A estrutura em camadas é uma organização hierárquica, ou seja, cada camada oferece serviços às camadas adjacentes. Um protocolo de rede pertence a uma camada de protocolos e pode ser implementado em *hardware*, *software* ou ambos.

Uma pilha de protocolos é um conjunto dos protocolos que pertencem a diferentes camadas. Um exemplo amplamente utilizado é o TCP/IP que é composto por cinco camadas, são elas: aplicação, transporte, rede, enlace e física. A altura de uma camada representa o quão perto ela se encontra dos usuários. Assim, a camada de aplicação no modelo TCP/IP e a camada de apresentação no modelo OSI são as camadas que de fato possuem interação com o usuário final. Alguns protocolos de relevância pertencentes a essas camadas são: HTTP, que é a base para comunicação na internet; FTP, que permite a transferência de arquivos; e IMAP, para recuperar e-mails de um servidor de e-mail.

2.3.2 Especificação IEEE 802.11

A especificação 802.11 [14] ou como é popularmente conhecido, Wi-Fi, contempla as duas camadas inferiores do modelo TCP/IP (enlace e física). Proporciona conexão sem fio para dispositivos relativamente próximos, oferecendo as mesmas funcionalidades básicas presentes na versão cabeada, com a adição da facilidade de instalação e associação à uma rede. Não está no escopo desse trabalho apresentar uma visão detalhada do funcionamento das rede sem fio, apenas abordar os assuntos essenciais que fundamentam a proposta do trabalho. Desse modo, por exemplo, não serão abordados tópicos como as diferenças entre as versões do protocolo 802.11, como ocorre a transmissão dos dados ou como funciona o protocolo de detecção de colisão CSMA/CA.

O conjunto básico de serviços de uma rede sem fio é composto por duas ou mais estações sem fio. Opcionalmente há uma estação base ou central (*access point* - AP). Isso é devido à existência de dois modos de organização definidos na especificação: o modo infraestrutura, onde existe uma estação central

que interconecta as estações e geralmente está interligado à estrutura cabeada, e a segunda organização, onde não existe de fato um dispositivo voltado a centralizar a comunicação e encaminhar para as estações destinatárias é chamado de *ad hoc*. Vale ressaltar que, nesse modo, uma das estações recebe algumas das responsabilidades de um AP.

Utilizando o modelo TCP/IP como exemplo, cada camada encapsula o conteúdo da camada anterior, adicionando seu cabeçalho. Assim, alguns campos são necessários para que a informação possa ser recuperada corretamente no destinatário. Na camada de enlace, a informação encapsulada junto com seu cabeçalho é chamada de quadro. São definidos três tipos principais de quadros e alguns sub-tipos no IEEE 802.11. Os quadros de administração ou *management frames* permitem que os *hosts* estabeleçam e mantenham a conexão com AP. Alguns sub-tipos são: *beacon*, que é enviado periodicamente para um AP anunciar sua presença; desautenticação, que é enviado por um *host* quando esse deseja encerrar a conexão com AP; dentre outros. Os quadros de controle servem para facilitar a troca de quadros entre os *hosts* e o AP. Por exemplo, o envio do quadro ACK (*Acknowledgement*) após o destinatário receber um quadro sem erro, ou os quadros RTS (*Request to Send*) e CTS (*Clear to Send*) para evitar a colisão de pacotes. O último tipo são os quadros que efetivamente carregam os dados. Todos os quadros compartilham a mesma estrutura,. Assim, todo quadro possui os campos dos tipos primários. Porém, somente os campos relativos ao tipo do quadro atual estão preenchidos.

2.3.3 Segurança em redes de computadores sem fio

Em momentos onde o vazamento de informações ganham os noticiários, alarmam os usuários e levantam questionamentos quanto à real capacidade dos sistemas e aplicações protegerem os dados contra acessos não autorizados. Garantir os princípios da segurança da informação é essencial para que informações valiosas, não importando o caráter, possam ser devidamente protegidas [51], [47] .

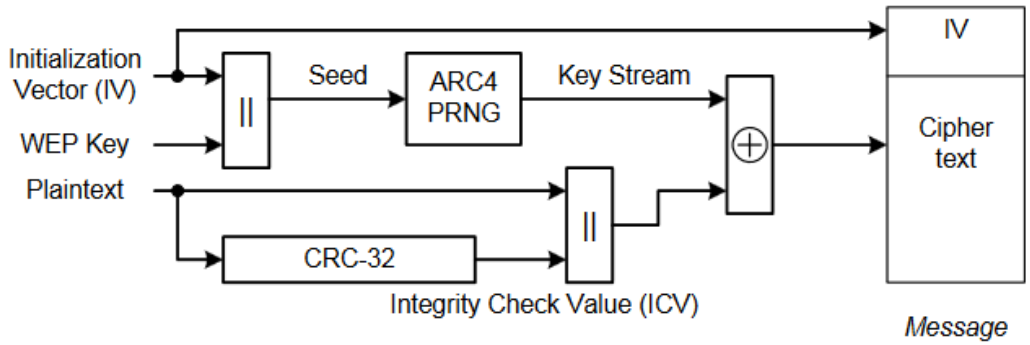
A criação das redes sem fio trouxe consigo uma série de novos riscos associados a não existência de uma conexão física, limitar o acesso a rede já não garante que a rede esteja protegida, a comodidade é inversamente proporcional a severidade das ameças que podem comprometer a rede. Pelo fato da informação estar no ar e ser facilmente capturada apenas modificando o modo de operação de um adaptador de rede, a segurança da rede é ainda mais crítica nesse cenário onde as redes sem fio são utilizadas.

Todo sistema deve prover mecanismos de segurança para que seus usuários e as informações nele presentes estejam protegidas contra o acesso de pessoas não autorizadas. Com as redes sem fio não é diferente. O WEP (*Wired Equivalent Protocol*) foi o primeiro mecanismo de segurança empregado nas redes sem fio, quando a especificação foi lançada, e tinha como objetivo oferecer o mesmo nível de segurança oferecido pelas redes cabeadas, apesar de suas particularidades. Dentre suas atribuições estão prover autenticação e criptografia dos dados utilizando criptografia de chave simétrica. Nesse caso, a mesma chave é utilizada para encriptação e desencriptação. Assim ela deve ser conhecida pelo remetente e o destinatário. Na fase de autenticação, a estação sem fio já recebeu quadros de sinalização de todos APs, cuja área de cobertura abranja a posição da estação referida. No quadro de sinalização o AP, envia seu endereço MAC e o SSID da rede, que é o identificador da rede, assim no momento da autenticação a estação sem fio sabe a quem deve direcionar a solicitação. Ao receber essa solicitação com o quadro

Authentication Request, o AP responde com um valor arbitrário chamado de *nonce*, em sequência a estação sem fio criptografa o *nonce* com sua chave e envia ao AP. Se o valor recebido for igual ao enviado, o cliente é autenticado recebendo o quadro *Authentication Response*.

A encriptação por meio do WEP exibida na Figura 2.3 funciona da seguinte maneira: primeiramente o IV (*Initialization Vector*), um número de 24 *bits* gerado a cada quadro é concatenado com a PSK (*Pre-Shared Key*), a senha definida no AP pelo administrador da rede para formar a chave de encriptação/desencriptação, essa chave resultante pode ter 64 ou 128 *bits* para o WEP-40 ou WEP-104, respectivamente. Em seguida essa chave serve de *seed* para a cifra de fluxo RC4 gerando o *key stream*, na sequência é adicionada à carga útil do quadro (*payload*) o ICV (*Integrity Check Value*), que é a saída do algoritmo de integridade CRC-32, que tem como entrada a própria carga útil. Finalmente, o texto cifrado é obtido, concatenando o IV e o resultado da operação lógica XOR entre o ICV e o *key stream*.

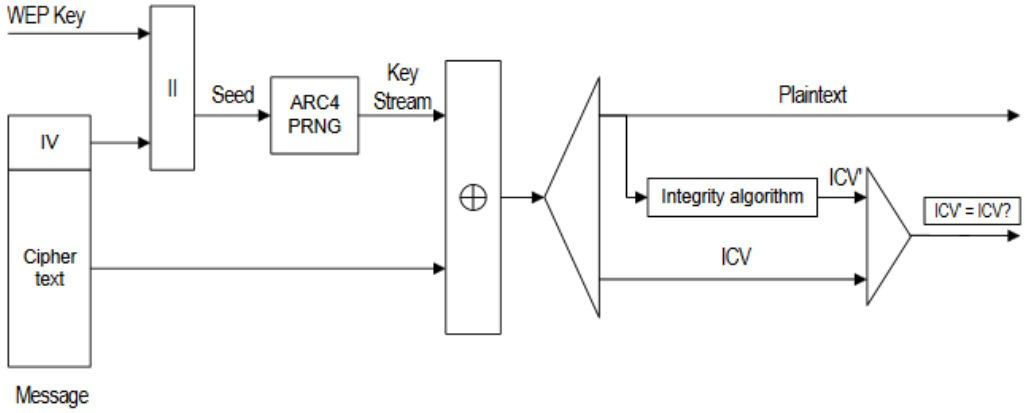
Figura 2.3: Fluxo de encriptação no WEP



Fonte: IEEE 802.11 [14]

Assim como acontece na etapa de encriptação, a PSK é concatenada com o IV para servir de *seed* para o algoritmo RC4 gerando o *key stream* que na sequência sofrerá o XOR com o texto cifrado. A saída dessa operação é o texto simples e o ICV. Um novo ICV (ICV') é gerado a partir da execução do algoritmo de integridade para realizar a comparação com o ICV obtido após a execução da operação lógica XOR. Esse processo de desencriptação pode ser visto de maneira sistemática na Figura 2.4.

Figura 2.4: Fluxo de desencriptação no WEP



Fonte: IEEE 802.11 [14]

O WEP deixou de ser o mecanismo de referência das redes sem fio após ataques publicados em fórum *online* pelo pseudônimo de Korek relatados na publicação de Chaabouni [11]. O trabalho de Fluhrer, Mantin e Shamir [24] mostrou que seu objetivo de garantir a confidencialidade da comunicação não foi alcançado. Apesar de substitutos como o 802.11i ou WPA (Wi-Fi Protected Access) e sua versão posterior WPA2 terem sido lançados corrigindo as vulnerabilidades encontradas através de mecanismos mais robustos, o WEP continua apresentando uma utilização considerável, seu uso pode ser consultado no serviço Wigle [67], que reune informações de redes sem fio enviadas pelos usuários.

2.4 IDS e WIDS

Manter um sistema seguro exige um monitoramento contínuo para que suas vulnerabilidades sejam encontradas, analisadas e devidamente corrigidas, antes que sejam exploradas por terceiros. Uma ameaça pode ser o acesso à informação, sua modificação ou mesmo a indisponibilidade do sistema provocada por um acesso indevido. A exploração de uma brecha de segurança pode afetar permanentemente a atividade de uma empresa ou até mesmo seus *stakeholders*. Um componente importante que compõe o leque de ferramentas utilizadas em ambientes empresariais são os sistemas de detecção de invasão, ou *Intrusion Detection Systems* (IDS), que geram alertas ao detectar ações potencialmente prejudiciais ao sistema [1], como por exemplo consumo de CPU e memória elevados e diferente da média de utilização, número de *sockets* abertos, número de tentativas falhas de *login* e tráfego de rede anormal. Quanto à sua classificação em relação ao modo de operação, os IDS são classificados em: baseado em assinatura e baseado em anomalia.

Nos sistemas de detecção de invasão baseados em assinatura, as decisões são extremamente dependentes da compreensão das atividades presentes num ataque, para criação de uma base de conhecimento que será consultada posteriormente para categorizar os eventos como normais ou invasão [40]. As principais vantagens dessa técnica são o baixo custo computacional e a baixa taxa de falsos positivos, isto é, categorizar erroneamente como invasão um evento que é legítimo. Por outro lado, como essa abordagem

busca por padrões específicos, não é possível identificar novos ataques. Mudanças sutis no modo de operação de um ataque podem burlar a detecção do IDS fazendo necessária a constante atualização e adição das assinaturas.

A detecção de invasão em um IDS utilizando a técnica de detecção de anomalia é baseada na premissa que o comportamento durante uma invasão difere do comportamento do uso regular. Nessa técnica, são criados perfis de uso representando o comportamento normal, qualquer atividade que diferencie da normalidade ultrapassando o limite estabelecido é considerada uma invasão [52]. A capacidade de reconhecer novos ataques assim que eles são executados é a principal vantagem da detecção de anomalia, em contrapartida ela apresenta uma alta taxa de falsos positivo. Dentre alguns métodos usados para a detecção de anomalia, destacam-se os estatísticos, aprendizado de máquina e mineração de dados [26], [9].

Sistemas de detecção de invasão de redes ou NIDS analisam os pacotes que trafegam nas redes na tentativa de identificar ameaças. Na teoria, todo pacote deveria ser verificado para que, mesmo no cenário de uma rede congestionada, o nível de detecção não deteriore, no entanto, na prática alguns pacotes são ignorados por questões de desempenho. Os *Wireless Intrusion Detection Systems* ou WIDS são uma especialização dos sistemas detecção de invasão que atuam de forma passiva capturando e analisando informações específicas das redes sem fio através de sensores distribuídos pelo ambiente. Eles devem ser planejadamente instalados, para que suas áreas de cobertura não se sobreponham e acabem produzindo registros inválidos.

2.5 Revisão da Literatura

O objetivo dessa seção é listar as publicações que ajudaram a construir o conhecimento necessário para o desenvolvimento deste trabalho. A publicação mais importante é o estudo de Kolias et al. [34] que, além de descrever a arquitetura das redes IEEE 802.11 ou sem fio, disponibiliza uma base de dados do tráfego em uma rede de computadores *wireless*. Eles apresentam os ataques mais frequentes em redes sem fio que seguem o padrão IEEE 802.11 em diversas versões dos mecanismo de segurança (isto é WEP, WPA, WPA2). Eles também apresentam o resultado da avaliação após a redução da dimensionalidade de alguns algoritmos baseados em regras, árvores de decisão, como: J48, Naive Bayes, One R dentre outros e podem ser consultados na Tabela 2.4. O algoritmo J48 apresentou o melhor resultado com uma *recall* de 96% e uma taxa de falsos positivos de 43%. No trabalho, não são informados quais atributos do quadro da especificação sem fio foram selecionados para os experimentos. Eles citam apenas que fizeram essa redução da dimensionalidade manualmente tendo como ponto de partida o trabalho de Neelakantan, Tech e Nagesh [48].

Algoritmo	Precisão	Recall	Taxa de falso positivo (FPR)	F-measure	Tempo (s)
AdaBoost	0,85	0,922	0,922	0,885	161,12
Hyperpipes	0,879	0,922	0,919	0,885	3,52
J48	0,962	0,963	0,436	0,948	568,92
Naive Bayes	0,917	0,906	0,399	0,909	29,67
OneR	0,9	0,946	0,642	0,922	156,98
Random Forest	0,959	0,958	0,493	0,944	739,78
Random Tree	0,959	0,962	0,438	0,948	49,3
ZeroR	0,85	0,922	0,922	0,885	3,65

Tabela 2.4: Resultados obtidos após o conjunto de dados ser reduzido no estudo de Kolias et al. [34]

O trabalho de Thanthrige, Samarabandu e Wang [64] utiliza a base de dados gerada na publicação de Kolias et al. [34], com a adição de técnicas de aprendizado de máquina chamada *feature reduction* ou redução de características, como *information gain* ou ganho de informação e *Chi-Squared statistics* ou estatística qui-quadrado. Eles experimentaram os mesmos algoritmos utilizados na publicação de Kolias et al. e constataram que, além da diminuição do tempo de processamento, a aplicação dessas técnicas causou um leve aumento na precisão de alguns algoritmos como o *Random Tree* que alcançou a precisão de 95,12%, um acréscimo de 2,4% em relação ao primeiro experimento que realizaram. Também observaram que a redução exagerada da dimensionalidade resulta no deterioramento da performance do classificador.

No trabalho de Cannady e Harrell [8] são apresentados alguns conceitos gerais presentes em IDS, como métricas, modelos e perfis. Também são apresentadas algumas abordagens para detecção de invasão como uso indevido (*misuse detection*), detecção de anomalia (*anomaly detection*) e uma abordagem híbrida. Há também o destaque de como a inteligência artificial e o aprendizado de máquina podem ser viáveis em IDSs. Na produção de García-Teodoro e et al. [26] há o destaque das três principais categorias de classificação de anomalias em um IDS baseada em estatísticas, onde o tráfego da rede é capturado e um perfil de comportamento é criado, baseado em conhecimento no qual especialistas auditam os dados baseados em algumas regras e aprendizado de máquina, onde um modelo permite a análise e identificação de padrões e consequentemente sua categorização.

As publicações de Juma et al. [33], Shahe e Khiyal [56] descrevem o levantamento dos trabalhos que estudaram como a composição de diferentes técnicas de aprendizado de máquina pode ajudar melhorar o desempenho dos classificadores. São relatados classificadores simples e classificadores híbridos onde são utilizados algoritmos de classificação e clusterização. Em Juma et al. é citado o trabalho de Abadeh et al. [3] que ao propor um algoritmo genético alcançou a taxa de detecção de 96,3% e 0,29% na taxa de falsos positivos. As publicações também relatam que geralmente IDS baseados em detecção de anomalia apresentam uma alta taxa de alarmes falso positivos. Um sistema de detecção de invasão que apresente esse comportamento, causa uma experiência frustante ao constantemente gerar alertas quando na verdade, a rede não está sofrendo ataque.

A produção de Singh e Nene [60] descreve as principais técnicas de aprendizado de máquina para classificação do comportamento como normal ou invasão. As técnicas estudadas e apresentadas no artigo são: redes neurais, máquinas de vetor suporte (*Support Vector Machine* - SVM), algoritmos genéticos, lógica fuzzy, redes bayesianas e árvores de decisão.

A publicação de Patcha e Park [52] faz um levantamento das principais técnicas em sistemas de detecção de invasão que utilizam a detecção de anomalia para identificar atividades potencialmente perigosas decorrentes da utilização de computadores, as técnicas expostas são: baseada em estatísticas, mineração de dados (*data mining*) e aprendizado de máquina. O estudo de Tao e Ruighaver [62] mostra as principais dificuldades encontradas no desenvolvimento WIDS em relação aos IDS, tais dificuldades são: interferência no sinal, a utilização de múltiplos canais e a utilização de múltiplos protocolos de rede.

Em Mitchell e Chen [45] é apresentado um estudo extensivo sobre WIDS cobrindo diferentes configurações de redes como WLAN, WPAN, WSN dentre outras especificações de redes. Além disso é feita a classificação dos IDS utilizando alguns critérios como: sistema alvo, técnica de detecção, processo de coleta, análise e resposta.

Capítulo 3

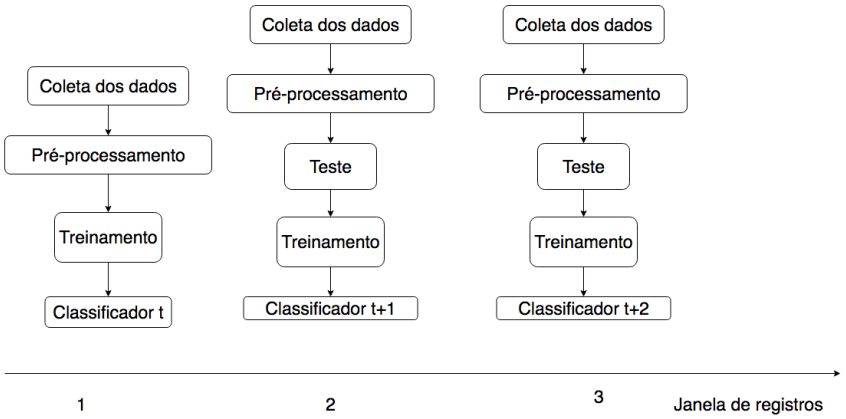
Metodologia de avaliação para detecção de anomalia em IDS/WIDS usando aprendizado *online*

3.1 Passos da Metodologia

A cada ano que passa, o número de pessoas conectadas à internet aumenta consideravelmente. Com o amadurecimento da internet das coisas (IoT), o tráfego nas redes também acompanhará esse crescimento [59], [49]. Essa demanda exige soluções cada vez mais eficientes, especialmente com relação à segurança, já que esses dispositivos podem ser alvos de ataques, e se comprometidos, podem causar perdas inestimáveis dado o nível de integração e controle que eles proporcionam. Diversas iniciativas de cibersegurança estão explorando a flexibilidade do aprendizado de máquina para lidar com as ameaças e desenvolver mecanismos de segurança mais eficientes [43], [32] e [4].

Além de seguir a metodologia tradicional de utilizar dois conjuntos de dados, um para treino e o outro para teste, também foi simulado o comportamento *online* partindo do ponto de vista dos dados e não do modelo, como pode ser visto na Figura 3.1. A simulação foi feita a partir da criação de janelas de registros, ou seja, a partir do conjunto de dados de treinamento foram definidos conjuntos de dados com k registros que foram utilizados para treinar os modelos e atualizar o classificador à medida que cada lote era processado. Na execução, há um parâmetro que indica qual a porcentagem do conjunto de dados deve ser utilizada. A seleção das janelas de registros ocorreu de forma pseudo aleatória, já que para os testes foi informado um valor de semente para o algoritmo de geração de um número aleatório.

Figura 3.1: Fluxo de execução do modo online



Fonte: Elaborada pelo autor

Desse modo, o objetivo desse trabalho é avaliar o uso do aprendizado *online* pode ser utilizado em WIDS que utilizem a técnica de detecção por anomalia para monitoramento e identificação de um ataque. Em termos práticos, deseja-se alcançar uma boa taxa de detecção com o mínimo de falsos positivos. A expectativa é que, como geralmente os algoritmos *online* são mais rápidos, a fase de treinamento seja consideravelmente menos custosa, e o modelo possa ser atualizado conforme os dados sejam disponibilizados.

3.2 Análise Experimental

3.2.1 Descrição da base de dados AWID

A Aegean WiFi Intrusion Dataset ou AWID como é chamada na publicação [34] é uma extensa base de dados composta de tráfego normal e ataques em redes sem fio. O tráfego foi coletado de uma rede montada especialmente para o estudo e tinha como dispositivos participantes computadores, celulares e smart TVs executando ações rotineiras da internet atual como navegação e *download* de arquivos. São apresentados dois conjuntos de dados: CLS e ATK. A diferença entre eles é como os registros são rotulados. O nível de detalhamento das classes no ATK é maior e, consequentemente, existem mais classes nele. Cada sessão para capturar o tráfego durou uma hora, sendo 25 minutos destinados aos ataques que foram realizados com as ferramentas aircrack-ng [5], MDK3 [41], metasploit [44] além de códigos próprios. A captura foi realizada utilizando a popular ferramenta de monitoramento Wireshark [68]. O conjunto de dados escolhido para o trabalho foi a versão reduzida do CLS, suas principais características podem ser consultadas na Tabela 3.1. Essa tabela compila as informações dos conjuntos de treinamento e de teste utilizados nos experimentos, contabiliza a quantidade de registros e a distribuição entre as classes. É fácil perceber como os conjuntos de dados estão desbalanceados, já que os registros rotulados como normal representam cerca de 90% de todos os registros. Essa base possui quatro classes sendo três delas de ataque, que são:

- *injection*: o objetivo desse ataque é gerar tráfego na rede normalmente utilizando pequenos quadros que servirão de base para os outros ataques;
- *impersonation*: esse é um ataque *man in the middle* onde o invasor introduz um falso AP (roteador) ou se passa por um cliente autorizado para interceptar, alterar e até interromper o tráfego e com as informações capturadas ele pode descobrir a senha da rede;
- *flooding*: nesse ataque a rede é bombardeada por quadros de controle, que normalmente são enviados pelo AP. Ele pode ser usado, por exemplo, para executar um ataque de negação de serviço (*DoS*), e desconectar todos os dispositivos da rede.

	Treino		Teste	
Classe	Quantidade	Proporção	Quantidade	Proporção
flooding	48484	0,027	8097	0,014
impersonation	48522	0,027	20079	0,035
injection	65379	0,036	16682	0,029
normal	1633190	0,91	530785	0,922
Total	1795575	1	575643	1

Tabela 3.1: Estatísticas do conjunto de dados utilizado

Os atributos presentes na AWID são os campos presentes em um quadro, segundo a especificação IEEE 802.11, a lista completa desses atributos podem ser encontrados no Apêndice A. A lista e a descrição dos atributos selecionados pode ser consultada na Tabela 3.2. A redução da dimensionalidade foi feita manualmente com base em Neelakantan, Tech e Nagesh [48], na publicação criadora da base de dados. Também foi realizada, uma extensiva busca na especificação IEEE 802.11, no fórum da ferramenta Wireshark e em diversos grupos de discussão para avaliar a relevância de cada atributo para o problema estudado. Vale salientar que a carga útil dos quadros não é analisada, somente os campos de um quadro de uma rede sem fio.

Complementado a redução da dimensionalidade e as transformações realizadas nos atributos como conversão de categórico/textual para numérico, foi necessário executar ações para tratar atributos ausentes e o desbalanceamento entre as classes. A primeira tentativa para o preenchimento dos atributos ausentes foi utilizar o algoritmo KNN para identificar as k instâncias mais próximas, e simplesmente preencher o atributo faltante com a média dos valores dessas k instâncias. Essa abordagem além de ser muita custosa não apresentou resultados satisfatórios. A abordagem escolhida para o preenchimento dos atributos foi para cada atributo em cada classe utilizar a moda, ou seja, o valor que mais se repete, garantindo assim a similaridade, pelo menos em relação aos atributos faltantes, das instâncias pertencentes ao mesmo tipo de ataque.

O desbalanceamento das classes é um problema recorrente no aprendizado de máquina e bastante estudado. Há algoritmos que não são sensíveis ao custo atribuído às classes e, nesses casos, o balanceamento das classes é uma prática recomendada para tentar melhorar a performance dos algoritmos.

Existem basicamente duas técnicas para lidar com o desbalanceamento, *oversampling* e *undersampling* [28], [35]. No *oversampling*, as instâncias das classes minoritárias são replicadas ou geradas artificialmente, e no *undersampling* os registros da classe majoritária são removidos para igualar a distribuição das instâncias entre as classes, o que pode causar perda de informação relevante. No modo *online*, essas técnicas não foram aplicadas para avaliar o comportamento dos classificadores numa situação mais próxima ao cenário comum, onde a maioria dos registros pertencem à classe normal.

Atributo	**Descrição**
wlan.fc.type	tipo do quadro (administração, controle, dados)
wlan.fc.type_subtype	subtipo do quadro de administração (autenticação, beacon e etc)
wlan.fc.subtype	subtipo do quadro de controle (RTC, CTS, ACK e etc)
wlan.fc.ds	composição dos campos ToDS e FromDS que indicam se o quadro foi enviado de uma estação para um AP, de uma rede cabeada para uma estação wireless dentre outros
wlan.fc.frag	indica se o quadro atual é o último fragmento (quando o quadro original é dividido em múltiplas partes)
wlan.fc.retry	indica se o quadro atual é uma retransmissão
wlan.fc.pwrmgt	flag indicando se o modo de economia está ativo
wlan.fc.moredata	flag indicando se há mais quadros para a estação de destino
wlan.fc.protected	flag indicando se a carga útil do quadro foi criptografada
wlan.wep.iv	vetor de inicialização
wlan.wep.icv	verificação de integridade do quadro
wlan.wep.key	índice da senha WEP (um AP pode ser configurado com até quatro senhas porém só pode utilizar uma por vez)
wlan.duration	tempo de duração estimado em microssegundos
wlan.bssid	endereço MAC do AP
wlan.ra	endereço MAC da estação intermediária que está recebendo o quadro
wlan.da	endereço MAC de destino
wlan.ta	endereço MAC da estação intermediária que está transmitindo o quadro
wlan.sa	endereço MAC de origem
wlan.seq	número do quadro
wlan.fcs_good	flag indicando se algum erro foi detectado
frame.len	tamanho do quadro em bytes
data.len	tamanho da carga em bytes
class	rótulo da instância

Tabela 3.2: Atributos selecionados após análise dos atributos de um quadro IEEE 802.11

3.2.2 Análise dos Resultados

Todas atividades como pré-processamento e testes foram realizadas em uma máquina virtual hospedada na Google Cloud Platform nomeada de *n1-highcpu-16* que possui 16 CPUs virtuais e 14,4 GB de memória ram.

Utilizando como referência os resultados da publicação que disponibilizou a base de dados [34] presentes na Tabela 2.4. Os resultados obtidos são ligeiramente inferiores, porém com média de tempo de treinamento muito inferior, cerca de 1 segundo contra 214 segundos. Os resultados experimentais podem ser encontrados na Tabela 3.3. São exibidas na tabela, além do tempo de execução as quatro métricas citadas anteriormente. Os algoritmos CW e ECCW tiveram um desempenho insatisfatório e apresentaram uma taxa de detecção em torno de 3%. Os algoritmos que demonstraram os melhores resultados foram: SOP e AROW, visto que deseja-se alcançar a maior taxa de detecção e a menor taxa de falsos positivos. Eles estão praticamente empatados, com médias de: precisão de 90%, recall de 92,5%, taxa de falsos positivos de 48,5%, *F-measure* de 91% e média de tempo de 0,86 segundos.

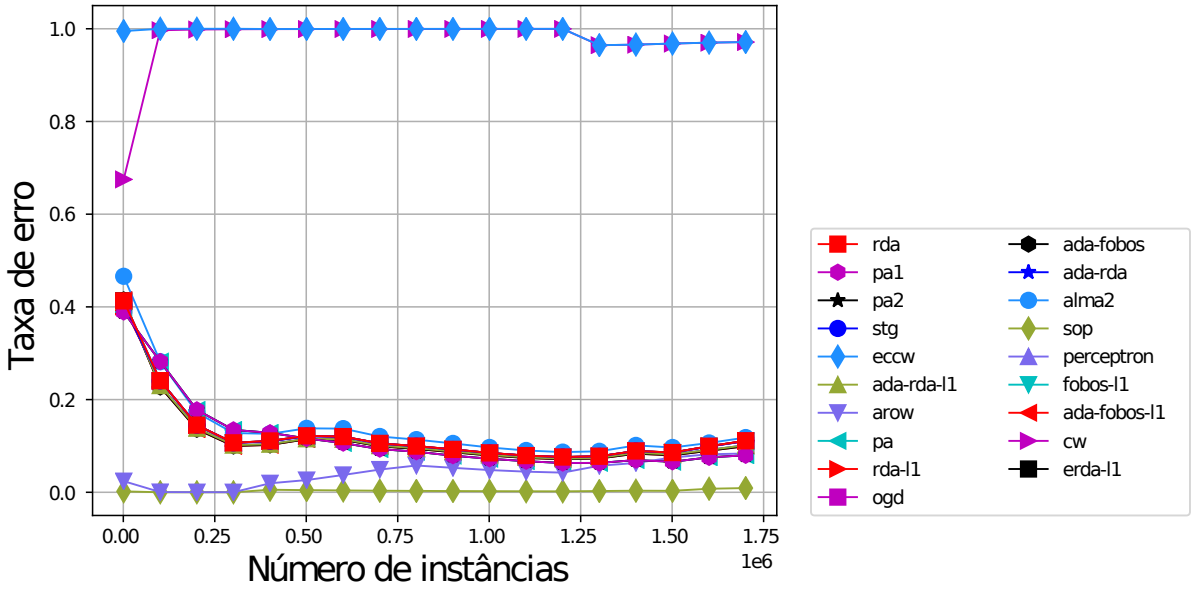
Algoritmo	**Precisão**	**Recall (TPR)**	**Taxa de falso positivo (FPR)**	**F-measure**	**Tempo (s)**
Perceptron	0,85	0,58	0,59	0,69	0,66
OGD	0,85	0,6	0,59	0,71	0,67
PA	0,86	0,81	0,75	0,83	0,68
ALMA	0,83	0,34	0,46	0,46	0,67
RDA	0,85	0,58	0,59	0,69	0,68
STG	0,85	0,6	0,59	0,71	0,68
FOBOS-L1	0,83	0,63	0,88	0,71	1,20
RDA-L1	0,85	0,58	0,59	0,69	0,2
ERDA-L1	0,85	0,58	0,59	0,69	0,83
SOP	0,9	0,93	0,5	0,91	0,9
CW	0	0,03	0,03	0	0,81
ECCW	0	0,03	0,03	0	0,68
AROW	0,9	0,92	0,47	0,91	0,82
Ada-FOBOS	0,83	0,63	0,88	0,71	0,65
Ada-RDA	0,82	0,58	0,91	0,68	0,71
Ada-FOBOS-L1	0,83	0,63	0,88	0,71	1,2
Ada-RDA-L1	0,82	0,58	0,91	0,68	1,05

Tabela 3.3: Resultados obtidos com todo conjunto de treinamento

Os gráficos na sequência mostram o comportamento dos classificadores à medida que as instâncias são apresentadas. Na Figura 3.2, pode ser observado como a taxa de erro (1 − precisão), diminui para a maioria dos algoritmos testados conforme as instâncias são consumidas. Note como a convergência é

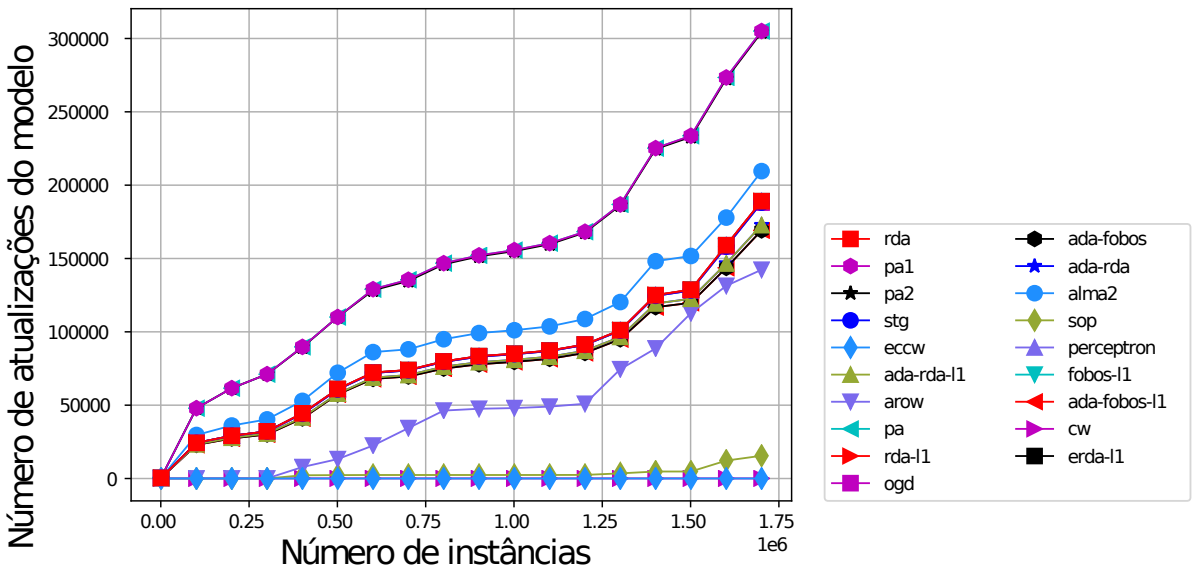
alcançada com cerca de 500 mil instâncias. Na Figura 3.3, pode-se notar como os classificadores sofrem atualizações constantes, com exceção dos algoritmos CW e ECCW que apresentaram um desempenho insatisfatório visto anteriormente.

Figura 3.2: Taxa de erro x Número de instâncias



Fonte: Elaborada pelo autor

Figura 3.3: Atualizações do modelo x Número de instâncias



Fonte: Elaborada pelo autor

Como foi citado anteriormente na apresentação das características dos conjuntos de dados, a base estava desbalanceada e pode causar a falsa impressão que bons resultados foram encontrados vide o recall obtido pelos algoritmos SOP, AROW e PA. Isso se dá pelo fato de rotular as instâncias de acordo com a classe majoritária, nesse caso a classe normal, está correto para a maioria dos casos. Pode ser observado na matriz de confusão normalizada do algoritmo SOP na Tabela 3.4, que a maioria dos registros pertencentes à classe normal foram rotulados corretamente, enquanto que para as outras classes os valores obtidos estão bem distantes de 1. Numa matriz de confusão normalizada de um classificador ideal, teríamos uma matriz identidade pelo fato do classificador ter acertado todas as predições realizadas.

Impersonation	Injection	Flooding	Normal	Classificado como:
0,00038	1	0	0,079	Impersonation
0	0,0085	0	1	Injection
0	0	0,34	1	Flooding
0,000043	0,00064	0	1	Normal

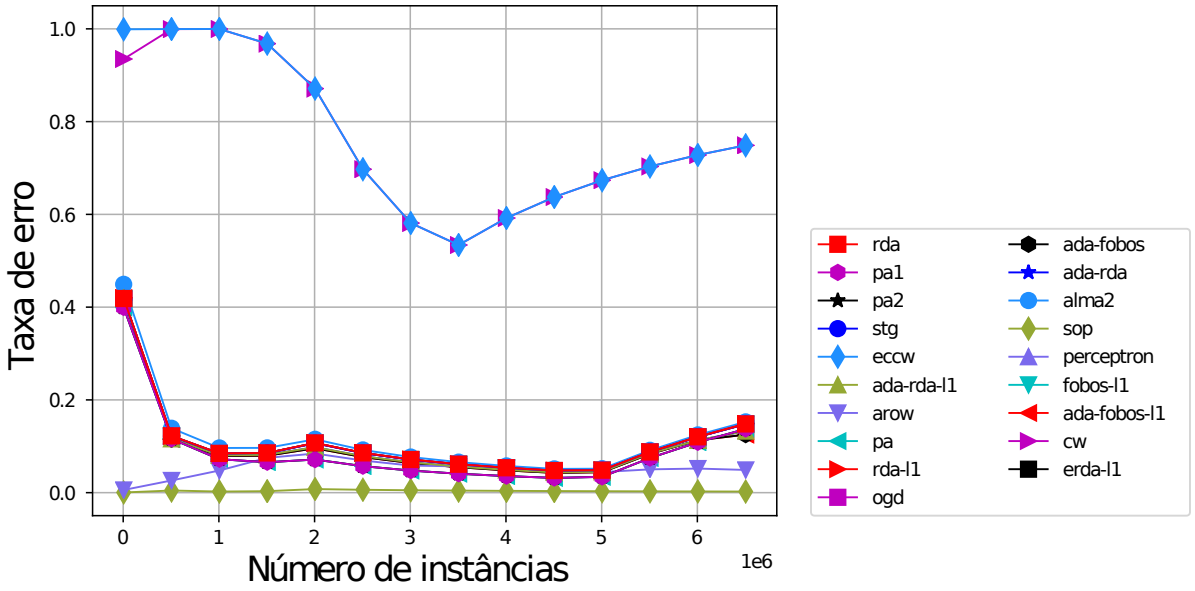
Tabela 3.4: Matriz de confusão do algoritmo SOP

Para obter resultados mais fidedignos foram aplicadas algumas técnicas de *oversampling* e *undersampling* presentes na biblioteca Imbalanced-learn [38] para investigar como o desbalanceamento das classes impacta na classificação dos algoritmos. Os resultados obtidos com o *oversampling* e *undersampling* da base podem ser consultados nas tabelas a seguir. A aplicação da técnica de *oversampling* como consta na Tabela 3.5, mostrou uma redução considerável na taxa de falsos positivos e uma leve queda na taxa de detecção e na *F-measure* para a maioria dos algoritmos. Agora, com a aplicação do algoritmo SMOTE [12], os melhores resultados foram obtidos pelos algoritmos Ada-FOBOS e Ada-FOBOS-L1 com 65% de precisão, 70% de recall, 10% de taxa de falsos positivos e 66% de *F-measure*. Com a aplicação do ADASYN [29], o algoritmo AROW apresentou o melhor resultado com 70% de precisão, 69% de recall, 10% de taxa de falsos positivos e 65% de *F-measure*. As Figuras 3.4 e 3.5 exibem o comportamento dos classificadores treinados com a base de dados que passou pelo processo de *oversampling*. Os resultados obtidos com ambas as técnicas (SMOTE e ADASYN) são praticamente idênticos.

	SMOTE					ADASYN				
Algoritmo	**Precisão**	**Recall (TPR)**	**Taxa de falso positivo (FPR)**	**F-measure**	**Tempo (s)**	**Precisão**	**Recall (TPR)**	**Taxa de falso positivo (FPR)**	**F-measure**	**Tempo (s)**
Perceptron	0,53	0,65	0,12	0,58	2,59	0,21	0,23	0,26	0,21	2,31
OGD	0,56	0,62	0,13	0,56	2,58	0,55	0,59	0,14	0,53	2,32
PA	0,51	0,48	0,17	0,47	2,61	0,57	0,54	0,14	0,53	2,30
ALMA	0,59	0,67	0,11	0,61	2,59	0,34	0,38	0,21	0,32	2,41
RDA	0,53	0,65	0,12	0,58	2,69	0,21	0,23	0,26	0,21	2,32
STG	0,56	0,62	0,13	0,56	2,65	0,55	0,59	0,14	0,53	2,43
FOBOS-L1	0,56	0,62	0,13	0,56	3,53	0,55	0,59	0,14	0,53	3,03
RDA-L1	0,44	0,52	0,16	0,42	2,97	0,28	0,4	0,2	0,31	2,68
ERDA-L1	0,53	0,65	0,12	0,58	3,04	0,21	0,23	0,26	0,21	2,60
SOP	0,54	0,52	0,16	0,46	3,30	0,55	0,57	0,14	0,48	2,89
CW	0,06	0,25	0,25	0,1	2,60	0,06	0, 25	0,25	0,1	2,30
ECCW	0,06	0,25	0,25	0,1	2,57	0,06	0,25	0,25	0,1	2,38
AROW	0,44	0,45	0,18	0,41	2,60	0,7	0,69	0,1	0,65	2,41
Ada-FOBOS	0,65	0,7	0,1	0,66	2,59	0,36	0,41	0,2	0,36	2,28
Ada-RDA	0,65	0,68	0,11	0,66	2,54	0,31	0,36	0,21	0,32	2,27
Ada-FOBOS-L1	0,65	0,7	0,1	0,66	4,22	0,36	0,41	0,2	0,36	3,77
Ada-RDA-L1	0,65	0,68	0,11	0,66	3,89	0,31	0,36	0,21	0,32	3,27

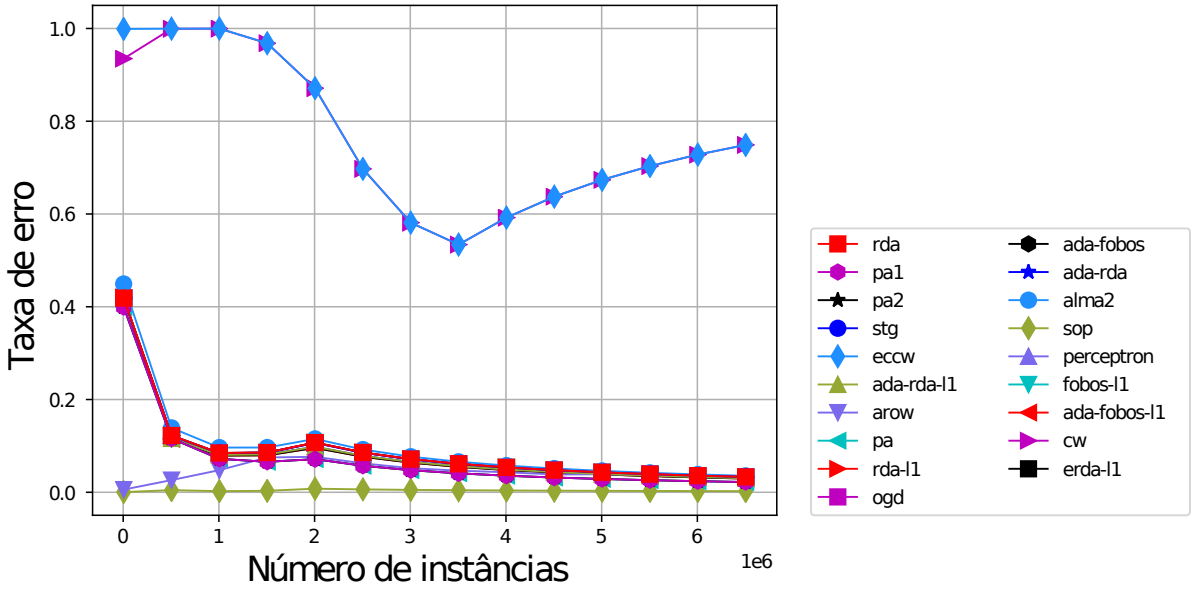
Tabela 3.5: Resultados obtidos após *oversampling*

Figura 3.4: Taxa de erro x Número de instâncias (*Oversampling* - SMOTE)



Fonte: Elaborada pelo autor

Figura 3.5: Taxa de erro x Número de instâncias (*Oversampling* - ADASYN)



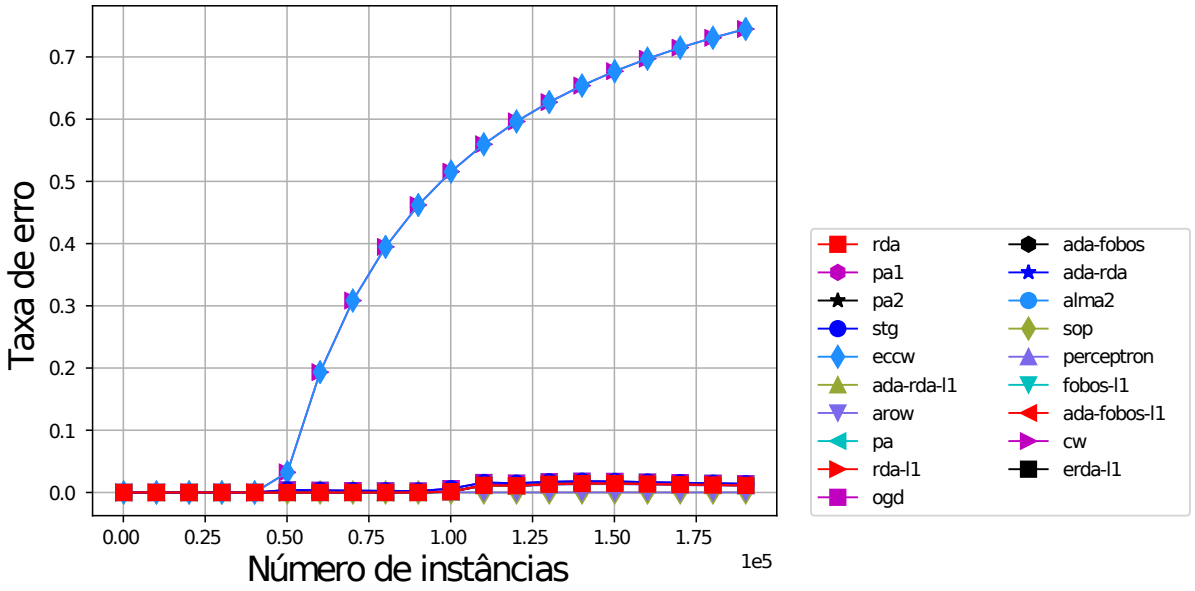
Fonte: Elaborada pelo autor

Como pode ser visto na Tabela 3.6, a aplicação das técnicas de *undersampling* não impactou positivamente no desempenho dos classificadores. A taxa de detecção caiu para 50% no algoritmo SOP e para 40% no STG com a utilização do *Near Miss* [42] e *Random Undersampling* [38], respectivamente. As Figuras 3.6 e 3.7 exibem o comportamento dos classificadores ao serem treinados com a base que passou pelo processo de *undersampling*. Como a aplicação do *undersampling* resulta na perda de informação, o comportamento observado nesse conjunto específico da base de dados não se assemelha ao observado anteriormente com a distribuição original das classes.

	Near Miss					Random Undersampling				
Algoritmo	**Precisão**	**Recall (TPR)**	**Taxa de falso positivo (FPR)**	**F-measure**	**Tempo (s)**	**Precisão**	**Recall (TPR)**	**Taxa de falso positivo (FPR)**	**F-measure**	**Tempo (s)**
Perceptron	0,51	0,27	0,24	0,26	0,07	0,2	0,37	0,21	0,26	0,08
OGD	0,49	0,29	0,24	0,0,3	0,07	0,45	0,4	0,2	0,3	0,08
PA	0,57	0,35	0,22	0,3	0,07	0,06	0,24	0,25	0,1	0,08
ALMA	0,26	0,27	0,24	0,26	0,08	0,15	0,18	0,27	0,1	0,08
RDA	0,49	0,29	0,24	0,3	0,08	0,28	0,17	0,28	0,11	0,08
STG	0,49	0,29	0,24	0,3	0,07	0,45	0,4	0,2	0,3	0,08
FOBOS-L1	0,49	0,29	0,24	0,3	0,09	0,45	0,4	0,2	0,3	0,10
RDA-L1	0,49	0,29	0,24	0,3	0,08	0,07	0,16	0,28	0,09	0,08
ERDA-L1	0,49	0,29	0,24	0,3	0,08	0,28	0,17	0,28	0,11	0,08
SOP	0,25	0,5	0,17	0,33	0,09	0,06	0,25	0,25	0,1	0,08
CW	0,06	0,25	0,25	0,1	0,08	0,06	0,25	0,25	0,1	0,08
ECCW	0,06	0,25	0,25	0,1	0,08	0,06	0,25	0,25	0,1	0,08
AROW	0,31	0,26	0,25	0,12	0,08	0,06	0,25	0,25	0,1	0,08
Ada-FOBOS	0,24	0,27	0,24	0,25	0,07	0,29	0,19	0,27	0,13	0,08
Ada-RDA	0,03	0,02	0,33	0,02	0,07	0,17	0,24	0,25	0,17	0,08
Ada-FOBOS-L1	0,24	0,27	0,24	0,25	0,12	0,29	0,19	0,27	0,13	0,12
Ada-RDA-L1	0,03	0,02	0,24	0,25	0,10	0,17	0,24	0,25	0,17	0,11

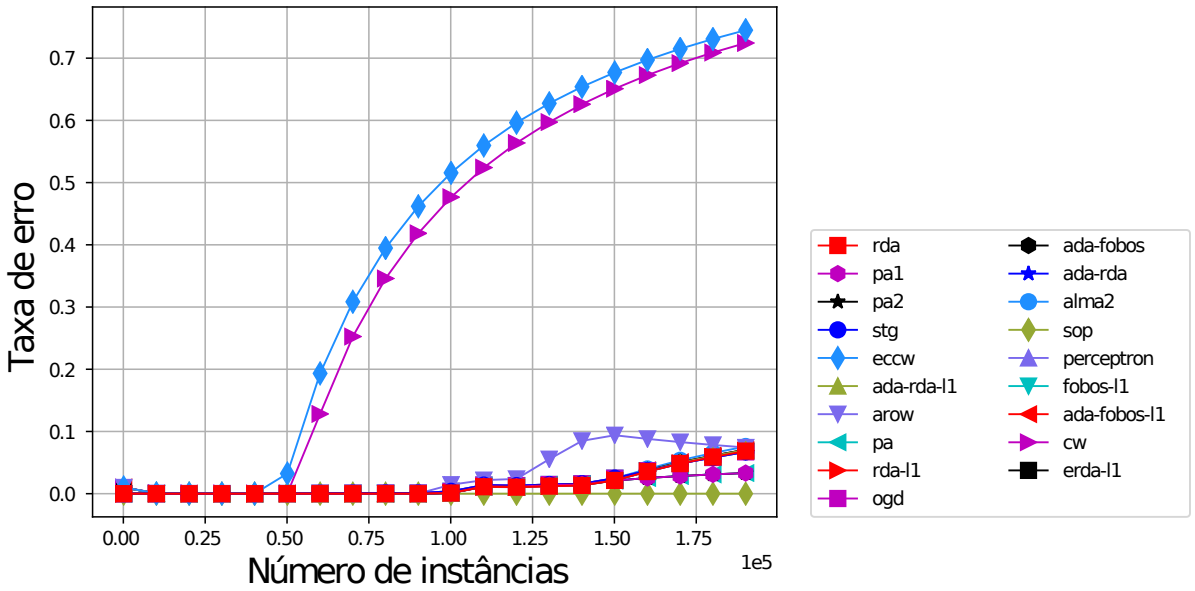
Tabela 3.6: Resultados obtidos após *undersampling*

Figura 3.6: Taxa de erro x Número de instâncias (*Undersampling - Near Miss*)



Fonte: Elaborada pelo autor

Figura 3.7: Taxa de erro x Número de instâncias (*Undersampling - Random Undersampling*)



Fonte: Elaborada pelo autor

Ao ajustar a distribuição das classes, os algoritmos não tendem a rotular as instâncias como a classe majoritária e como era de de esperar após a aplicação das técnicas a taxa de detecção (*recall*) dos ataques diminui e, por outro lado, a taxa de falsos positivos (FPR) também decresceu o que era uma das premissas do funcionamento de IDS baseado em anomalia.

Finalizando as abordagens analisadas, o modo online do ponto de vista dos dados foi testado com diversos valores de k e a melhor média de resultado foi obtido quando cada janela possuía 5000 registros

e pode ser observado na Tabela 3.7. A partir da segunda janela, cada conjunto de dados foi utilizado para teste e depois para treino do classificador.

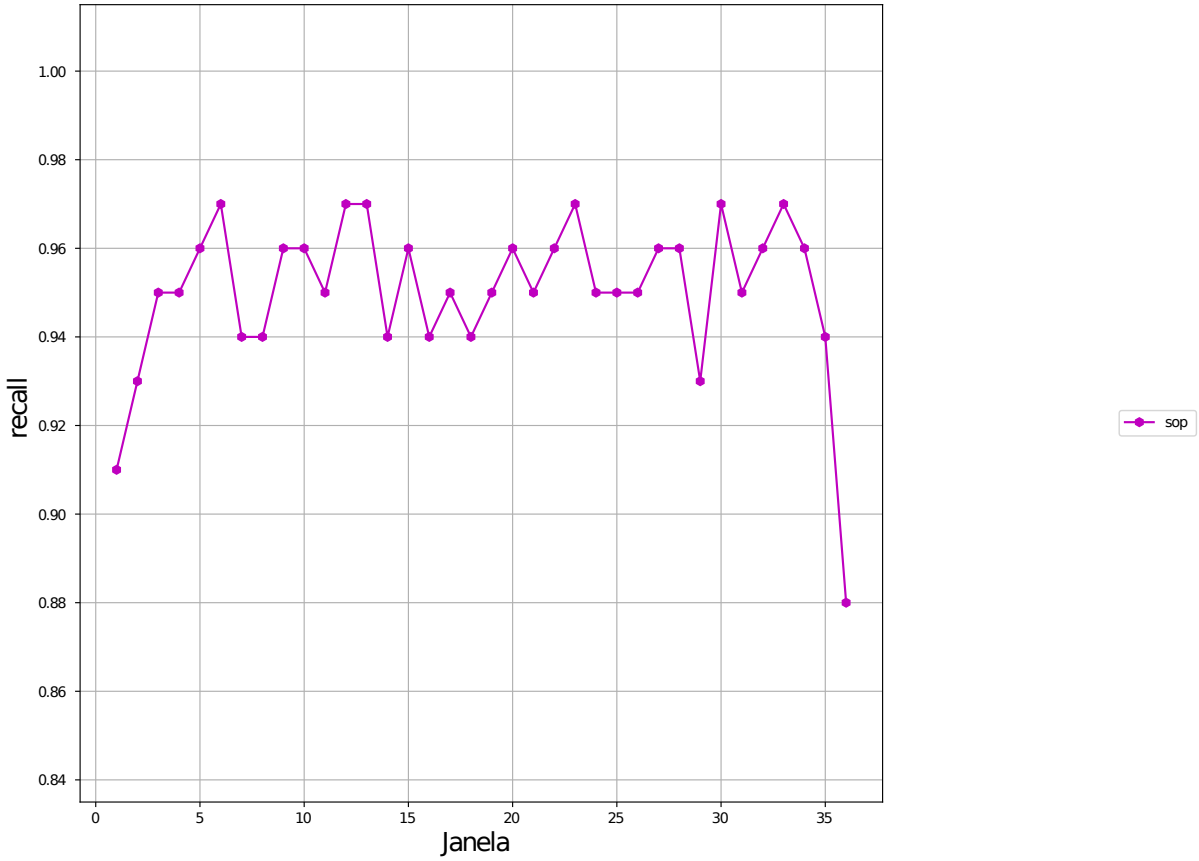
Ao compararmos com os resultados da primeira execução, os resultados obtidos nesse modo online indicam uma pequena variação na precisão, com exceção do algoritmo ECCW que nesse momento alcançou o valor de 38% contra uma precisão nula obtida anteriormente. Outro fato que chama a atenção é como a taxa de falsos positivos decresceu cerca de 40% nos algoritmos SOP e Ada-RDA-L1. O algoritmo SOP apresentou o melhor resultado com precisão, recall e *F-measure* acima de 90% e uma taxa de falsos positivos de 17%.

Algoritmo	**Precisão**	**Recall (TPR)**	**Taxa de falso positivo (FPR)**	**F-measure**	**Tempo (s)**
Perceptron	0,85	0,66	0,51	0,73	0,003
OGD	0,85	0,68	0,55	0,74	0,003
PA	0,89	0,80	0,43	0,83	0,003
ALMA	0,86	0,66	0,49	0,73	0,003
RDA	0,85	0,65	0,50	0,72	0,002
STG	0,85	0,68	0,55	0,74	0,003
FOBOS-L1	0,85	0,68	0,55	0,74	0,004
RDA-L1	0,86	0,67	0,50	0,73	0,004
ERDA-L1	0,85	0,65	0,50	0,72	0,003
SOP	0,96	0,95	0,17	0,95	0,003
CW	0,02	0,05	0,04	0,03	0,003
ECCW	0,38	0,32	0,23	0,32	0,002
AROW	0,92	0,90	0,49	0,91	0,003
Ada-FOBOS	0,86	0,68	0,48	0,75	0,002
Ada-RDA	0,86	0,66	0,50	0,73	0,003
Ada-FOBOS-L1	0,86	0,68	0,48	0,75	0,004
Ada-RDA-L1	0,86	0,66	0,50	0,73	0,004

Tabela 3.7: Resultados obtidos com a janela de 5000 registros

O gráfico da Figura 3.8 exibe a taxa de detecção do algoritmo que demonstrou o melhor resultado ao decorrer das k janelas de 5000 registros cada. O objetivo desse teste foi simular a análise em tempo real dos registros e verificar como a taxa de detecção e a taxa de falsos positivos variam conforme os dados são apresentados.

Figura 3.8: Recall ao decorrer das janelas de 5000 registros



Fonte: Elaborada pelo autor

Capítulo 4

Conclusões e Trabalhos Futuros

Nesse trabalho foi estudado como o aprendizado de máquina, em especial o aprendizado *online* pode contribuir para a construção de sistemas de detecção de invasão baseados em anomalia. A identificação de um registro originário do tráfego de uma rede sem fio, como potencialmente perigoso ou não foi modelado como um problema de classificação. Como o conjunto de dados estava rotulado, a abordagem supervisionada foi escolhida.

Dentre os esforços investidos neste trabalho, destacam-se: levantamento bibliográfico, realizado para entender o funcionamento de uma sistema de detecção de invasão e como aprendizado de máquina vem sendo aplicado em soluções de cibersegurança; seleção de atributos de um quadro IEEE 802.11, filtragem de atributos relevantes para o problema estudado; levantamento de ataques, identificação dos principais ataques contra redes sem fio; pré-processamento do conjunto de dados, procedimento de preparação dos dados para utilização nos classificadores; seleção das métricas, investigação sobre métricas apropriadas para mensurar o desempenho dos classificadores. Num primeiro momento, os experimentos foram realizados com a mínima intervenção possível no conjunto de dados. Posteriormente, foram aplicadas técnicas de balanceamento por causa da classe normal. A aplicação do *oversampling* resultou na diminuição da taxa de falsos positivos. Uma maior redução na taxa de falsos positivos é o objetivo a ser alcançado em trabalhos futuros, como por exemplo, o trabalho de Abadeh et al [3]. A solução apresentada demonstrou o potencial em conjuntos com outras técnicas construir um sistema de detecção de invasão eficiente. A utilização do conjunto de dados AWID e da biblioteca LIBSOL permitiu a construção e execução em tempo hábil de diversos experimentos. Como foi exibido na seção anterior, os parâmetros que caracterizam um sistema de detecção de invasão viável que utilize a técnica de detecção de anomalia, isto é, alta taxa de detecção e um baixo de número de falsos positivos foram alcançados e há diversas possibilidades de aperfeiçoamento.

O primeiro ponto que pode ajudar melhorar a performance é a aquisição de mais dados, nesse caso seria possível utilizar a versão estendida do conjunto de dados. Um ponto que pode ser explorado no futuro é a composição de diferentes algoritmos explorando o fato de alguns apresentarem uma melhor performance em determinada classe. Uma outra tentativa é a combinação de diferentes técnicas de aprendizado de máquina, por exemplo, combinando algoritmos de clusterização e posteriormente classificação

ou vice-versa [33], [56]. Uma outra abordagem que pode melhorar a solução proposta e se não contribuir positivamente ajudará a confirmar que a escolha por algoritmos lineares foi correta, é estudar a utilização de algoritmos não lineares com os propostos em [58] e [13] já que iniciamos o estudo com o palpite das classes serem linearmente separáveis, pelo fato de não ser possível visualizar esse comportamento em problemas multidimensionais.

Referências Bibliográficas

[1] J. P. ANDERSON. "Computer security threat monitoring and surveillance". Em: *Technical Report, James P. Anderson Company* (1980). URL: https://ci.nii.ac.jp/naid/10014688737/en/.

[2] Wil van der Aalst. "Process Mining: Overview and Opportunities". Em: *ACM Trans. Manage. Inf. Syst.* 3.2 (jul. de 2012), 7:1–7:17. ISSN: 2158-656X. DOI: 10.1145/2229156.2229157. URL: http://doi.acm.org/10.1145/2229156.2229157.

[3] Mohammad Saniee Abadeh et al. "A parallel genetic local search algorithm for intrusion detection in computer networks". Em: *Engineering Applications of Artificial Intelligence* 20.8 (2007), pp. 1058–1069.

[4] *Advanced Analytics and Machine Learning: A Prescriptive and Proactive Approach to Security.* https://www.mcafee.com/enterprise/pt-br/forms/gated-form.html?docID=cf0969890a0d463e425c853894e26e87&aType=white_paper. Acesso em: 19/07/2018.

[5] *Aircrack-ng.* http://www.aircrack-ng.org/. Acesso em: 25/05/2018.

[6] Taiwo Oladipupo Ayodele. "Types of machine learning algorithms". Em: *New advances in machine learning.* InTech, 2010.

[7] John L Berg e Harald Schumny. *An analysis of the information technology standardization process.* Elsevier, 2012.

[8] James Cannady e Jay Harrell. "A comparative analysis of current intrusion detection technologies". Em: *In Proc. of the Fourth Technology for Information Security Conference'96 (TISC'96.* 1996.

[9] Carlos A Catania e Carlos GarcíA Garino. "Automatic network intrusion detection: Current techniques and open issues". Em: *Computers & Electrical Engineering* 38.5 (2012), pp. 1062–1072.

[10] Nicolo Cesa-Bianchi, Alex Conconi e Claudio Gentile. "A second-order perceptron algorithm". Em: *SIAM Journal on Computing* 34.3 (2005), pp. 640–668.

[11] Rafik Chaabouni. *Break WEP faster with statistical analysis.* Rel. téc. 2006.

[12] Nitesh V Chawla et al. "SMOTE: synthetic minority over-sampling technique". Em: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[13] Mingmin Chi, Huijun He e Wenqiang Zhang. "Nonlinear Online Classification Algorithm with Probability Margin". Em: *Asian Conference on Machine Learning.* 2011, pp. 33–46.

[14] IEEE Computer Society LAN MAN Standards Committee et al. "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications". Em: *IEEE Standard 802.11-1997* (1997).

[15] Koby Crammer, Mark Dredze e Fernando Pereira. "Exact convex confidence-weighted learning". Em: *Advances in Neural Information Processing Systems*. 2009, pp. 345–352.

[16] Koby Crammer, Alex Kulesza e Mark Dredze. "Adaptive regularization of weight vectors". Em: *Advances in neural information processing systems*. 2009, pp. 414–422.

[17] Koby Crammer et al. "Online passive-aggressive algorithms". Em: *Journal of Machine Learning Research* 7.Mar (2006), pp. 551–585.

[18] Pedro Domingos. "A few useful things to know about machine learning". Em: *Communications of the ACM* 55.10 (2012), pp. 78–87.

[19] Mark Dredze, Koby Crammer e Fernando Pereira. "Confidence-weighted linear classification". Em: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 264–271.

[20] John Duchi, Elad Hazan e Yoram Singer. "Adaptive subgradient methods for online learning and stochastic optimization". Em: *Journal of Machine Learning Research* 12.Jul (2011), pp. 2121–2159.

[21] John Duchi e Yoram Singer. "Efficient online and batch learning using forward backward splitting". Em: *Journal of Machine Learning Research* 10.Dec (2009), pp. 2899–2934.

[22] *Essentials of Machine Learning Algorithms (with Python and R Codes)*. `https://www.analyticsvidhya.com/blog/2017/09/common-machine-learning-algorithms/`. Acesso em: 30/06/2018.

[23] *Estimating an Optimal Learning Rate For a Deep Neural Network*.

[24] Scott Fluhrer, Itsik Mantin e Adi Shamir. "Weaknesses in the key scheduling algorithm of RC4". Em: *International Workshop on Selected Areas in Cryptography*. Springer. 2001, pp. 1–24.

[25] Mikel Galar et al. "An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes". Em: *Pattern Recognition* 44.8 (2011), pp. 1761–1776.

[26] Pedro Garcia-Teodoro et al. "Anomaly-based network intrusion detection: Techniques, systems and challenges". Em: *computers & security* 28.1 (2009), pp. 18–28.

[27] Claudio Gentile. "A new approximate maximal margin classification algorithm". Em: *Journal of Machine Learning Research* 2.Dec (2001), pp. 213–242.

[28] Haibo He e Edwardo A Garcia. "Learning from imbalanced data". Em: *IEEE Transactions on knowledge and data engineering* 21.9 (2009), pp. 1263–1284.

[29] Haibo He et al. "ADASYN: Adaptive synthetic sampling approach for imbalanced learning". Em: *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE. 2008, pp. 1322–1328.

[30] *How Much Training Data is Required for Machine Learning?* `https://machinelearningmastery.com/much-training-data-required-machine-learning/`. Acesso em: 01/07/2018.

[31] Jih-Jeng Huang, Gwo-Hshiung Tzeng e Chorng-Shyong Ong. "Marketing segmentation using support vector clustering". Em: *Expert systems with applications* 32.2 (2007), pp. 313–317.

[32] *Intelligent Security: Using Machine Learning to Help Detect Advanced Cyber Attacks*. `https://info.microsoft.com/intelligent-security-e-book-registration.html`. Acesso em: 19/07/2018.

[33] SUNDUS JUMA et al. "MACHINE LEARNING TECHNIQUES FOR INTRUSION DETECTION SYSTEM: A REVIEW." Em: *Journal of Theoretical & Applied Information Technology* 72.3 (2015).

[34] Constantinos Kolias et al. "Intrusion detection in 802.11 networks: empirical evaluation of threats and a public dataset". Em: *IEEE Communications Surveys & Tutorials* 18.1 (2015), pp. 184–208.

[35] CV KrishnaVeni e T Sobha Rani. "On the classification of imbalanced datasets". Em: *IJCST* 2.SP1 (2011), pp. 145–148.

[36] James F. Kurose e Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. 6th. Pearson, 2012. ISBN: 0132856204, 9780132856201.

[37] John Langford, Lihong Li e Tong Zhang. "Sparse online learning via truncated gradient". Em: *Journal of Machine Learning Research* 10.Mar (2009), pp. 777–801.

[38] Guillaume Lemaître, Fernando Nogueira e Christos K. Aridas. "Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning". Em: *Journal of Machine Learning Research* 18.17 (2017), pp. 1–5. URL: `http://jmlr.org/papers/v18/16-365`.

[39] Lei-jun Li e Hong Peng. "A defense model study based on IDS and firewall linkage". Em: *Information Science and Management Engineering (ISME), 2010 International Conference of*. Vol. 2. IEEE. 2010, pp. 91–94.

[40] Hung-Jen Liao et al. "Intrusion detection system: A comprehensive review". Em: *Journal of Network and Computer Applications* 36.1 (2013), pp. 16–24.

[41] *MDK 3 - git.kali.org - packages/mdk3.git/summary*. `git://git.kali.org/packages/mdk3.git`. Acesso em: 25/05/2018.

[42] Inderjeet Mani e I Zhang. "kNN approach to unbalanced data distributions: a case study involving information extraction". Em: *Proceedings of workshop on learning from imbalanced datasets*. Vol. 126. 2003.

[43] *McAfee: Machine Learning Raises Security Teams to the Next Level | TechPapers*. `https://techpapers.co.uk/mcafee_105/`. Acesso em: 19/07/2018.

[44] *Metasploit | Penetration Testing Software, Pen Testing Security | Metasploit*. `https://www.metasploit.com/`. Acesso em: 25/05/2018.

[45] Robert Mitchell e Ray Chen. "A survey of intrusion detection in wireless network applications". Em: *Computer Communications* 42 (2014), pp. 1–23.

[46] Tom M Mitchell. "Machine learning. 1997". Em: *Burr Ridge, IL: McGraw Hill* 45.37 (1997), pp. 870–877.

[47] Haralambos Mouratidis, Paolo Giorgini e Gordon Manson. "When security meets software engineering: a case of modelling secure information systems". Em: *Information Systems* 30.8 (2005), pp. 609–629.

[48] M. Tech N. Pratik Neelakantan C. Nagesh. "Role of Feature Selection in Intrusion Detection Systems for 802.11 Networks". Em: *International Journal of Smart Sensors and Ad Hoc Networks (IJSSAN)* (2011).

[49] Andrew M Odlyzko. "Internet traffic growth: Sources and implications". Em: *Optical transmission systems and equipment for WDM networking II*. Vol. 5247. International Society for Optics e Photonics. 2003, pp. 1–16.

[50] *Overfitting in Machine Learning: What It Is and How to Prevent It*. `https://elitedatascience.com/overfitting-in-machine-learning`. Acesso em: 01/07/2018.

[51] Donn B Parker. "The strategic values of information security in business". Em: *Computers & Security* 16.7 (1997), pp. 572–582.

[52] Animesh Patcha e Jung-Min Park. "An overview of anomaly detection techniques: Existing solutions and latest technological trends". Em: *Computer networks* 51.12 (2007), pp. 3448–3470.

[53] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". Em: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[54] Frank Rosenblatt. "The perceptron: a probabilistic model for information storage and organization in the brain." Em: *Psychological review* 65.6 (1958), p. 386.

[55] AL Samuel. "Some Studies in Machine Learning Using the Game of Checkers". Em: *IBM Journal of Research and Development* 3.3 (1959), p. 210.

[56] Asghar Ali Shah, Malik Sikander Hayat Khiyal e Muhammad Daud Awan. "Analysis of Machine Learning Techniques for Intrusion Detection System: A Review". Em: *International Journal of Computer Applications* 119.3 (2015).

[57] Shai Shalev-Shwartz et al. "Online learning and online convex optimization". Em: *Foundations and Trends® in Machine Learning* 4.2 (2012), pp. 107–194.

[58] Si Si, Sanjiv Kumar e Yang Li. "Nonlinear Online Learning with Adaptive Nyström Approximation". Em: *CoRR* abs/1802.07887 (2018). arXiv: `1802.07887`. URL: `http://arxiv.org/abs/1802.07887`.

[59] Dhananjay Singh, Gaurav Tripathi e Antonio J Jara. "A survey of Internet-of-Things: Future vision, architecture, challenges and services". Em: *Internet of things (WF-IoT), 2014 IEEE world forum on*. IEEE. 2014, pp. 287–292.

[60] Jayveer Singh e Manisha J Nene. "A survey on machine learning techniques for intrusion detection systems". Em: *International Journal of Advanced Research in Computer and Communication Engineering* 2.11 (2013), pp. 4349–4355.

[61] Marina Sokolova e Guy Lapalme. "A systematic analysis of performance measures for classification tasks". Em: *Information Processing & Management* 45.4 (2009), pp. 427–437.

[62] Zhiqi Tao e AB Ruighaver. “Wireless intrusion detection: Not as easy as traditional network intrusion detection”. Em: *TENCON 2005-2005 IEEE Region 10 Conference*. IEEE. 2005, pp. 1–5.

[63] *Top 15 Frameworks for Machine Learning Experts*. `https://www.kdnuggets.com/2016/04/top-15-frameworks-machine-learning-experts.html`. Acesso em: 01/07/2018.

[64] Xianbin Wang Udaya Sampath K. Perera Miriya Thanthrige Jagath Samarabandu. “Machine Learning Techniques for Intrusion Detection on Public Dataset”. Em: *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)* (2016).

[65] Rajasekar Venkatesan e Meng Joo Er. “A novel progressive learning technique for multi-class classification”. Em: *Neurocomputing* 207 (2016), pp. 310–321.

[66] *Why are we experiencing such an explosive growth of machine learning and its applications today even though the space exists for more than 3-4 decades?* `https://www.quora.com/Why-are-we-experiencing-such-an-explosive-growth-of-machine-learning-and-its-applications-today-even-though-the-space-exists-for-more-than-3-4-decades`. Acesso em: 01/07/2018.

[67] *WiGLE Stats*. `https://wigle.net/stats`. Acesso em: 26/03/2018.

[68] *Wireshark · Go Deep*. `https://www.wireshark.org/`. Acesso em: 25/05/2018.

[69] Yue Wu, Steven C.H. Hoi e Nenghai Yu. “LIBSOL: A Library for Scalable Online Learning Algorithms”. Em: *SMU Technical Report (SMU-TR-2016-07-25)* (2016).

[70] Lin Xiao. “Dual averaging methods for regularized stochastic learning and online optimization”. Em: *Journal of Machine Learning Research* 11.Oct (2010), pp. 2543–2596.

[71] Martin Zinkevich. “Online convex programming and generalized infinitesimal gradient ascent”. Em: *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*. 2003, pp. 928–936.

Apêndice A - Lista de atributos de um quadro IEEE 802.11 capturado pelo Wireshark

frame.interface_id

frame.dlt

frame.offset_shift

frame.time_epoch

frame.time_delta

frame.time_delta_displayed

frame.time_relative

frame.len

frame.cap_len

frame.marked

frame.ignored

radiotap.version

radiotap.pad

radiotap.length

radiotap.present.tsft

radiotap.present.flags

radiotap.present.rate

radiotap.present.channel

radiotap.present.fhss

radiotap.present.dbm_antsignal

radiotap.present.dbm_antnoise

radiotap.present.lock_quality

radiotap.present.tx_attenuation

radiotap.present.db_tx_attenuation

radiotap.present.dbm_tx_power

radiotap.present.antenna

radiotap.present.db_antsignal

radiotap.present.db_antnoise

radiotap.present.rxflags

radiotap.present.xchannel

radiotap.present.mcs

radiotap.present.ampdu

radiotap.present.vht

radiotap.present.reserved

radiotap.present.rtap_ns

radiotap.present.vendor_ns

radiotap.present.ext

radiotap.mactime

radiotap.flags.cfp

radiotap.flags.preamble

radiotap.flags.wep

radiotap.flags.frag

radiotap.flags.fcs

radiotap.flags.datapad

radiotap.flags.badfcs

radiotap.flags.shortgi

radiotap.datarate

radiotap.channel.freq

radiotap.channel.type.turbo

radiotap.channel.type.cck

radiotap.channel.type.ofdm

radiotap.channel.type.2ghz

radiotap.channel.type.5ghz

radiotap.channel.type.passive

radiotap.channel.type.dynamic

radiotap.channel.type.gfsk

radiotap.channel.type.gsm

radiotap.channel.type.sturbo

radiotap.channel.type.half

radiotap.channel.type.quarter

radiotap.dbm_antsignal

radiotap.antenna

radiotap.rxflags.badplcp

wlan.fc.type_subtype

wlan.fc.version

wlan.fc.type

wlan.fc.subtype

wlan.fc.ds

wlan.fc.frag

wlan.fc.retry

wlan.fc.pwrmgt

wlan.fc.moredata

wlan.fc.protected

wlan.fc.order

wlan.duration

wlan.ra

wlan.da

wlan.ta

wlan.sa

wlan.bssid

wlan.frag

wlan.seq

wlan.bar.type

wlan.ba.control.ackpolicy

wlan.ba.control.multitid

wlan.ba.control.cbitmap

wlan.bar.compressed.tidinfo

wlan.ba.bm

wlan.fcs_good

wlan_mgt.fixed.capabilities.ess

wlan_mgt.fixed.capabilities.ibss

wlan_mgt.fixed.capabilities.cfpoll.ap

wlan_mgt.fixed.capabilities.privacy

wlan_mgt.fixed.capabilities.preamble

wlan_mgt.fixed.capabilities.pbcc

wlan_mgt.fixed.capabilities.agility

wlan_mgt.fixed.capabilities.spec_man

wlan_mgt.fixed.capabilities.short_slot_time

wlan_mgt.fixed.capabilities.apsd

wlan_mgt.fixed.capabilities.radio_measurement

wlan_mgt.fixed.capabilities.dsss_ofdm

wlan_mgt.fixed.capabilities.del_blk_ack

wlan_mgt.fixed.capabilities.imm_blk_ack

wlan_mgt.fixed.listen_ival

wlan_mgt.fixed.current_ap

wlan_mgt.fixed.status_code

wlan_mgt.fixed.timestamp

wlan_mgt.fixed.beacon

wlan_mgt.fixed.aid

wlan_mgt.fixed.reason_code

wlan_mgt.fixed.auth.alg

wlan_mgt.fixed.auth_seq

wlan_mgt.fixed.category_code

wlan_mgt.fixed.htact

wlan_mgt.fixed.chanwidth

wlan_mgt.fixed.fragment

wlan_mgt.fixed.sequence

wlan_mgt.tagged.all

wlan_mgt.ssid

wlan_mgt.ds.current_channel

wlan_mgt.tim.dtim_count

wlan_mgt.tim.dtim_period

wlan_mgt.tim.bmapctl.multicast

wlan_mgt.tim.bmapctl.offset

wlan_mgt.country_info.environment

wlan_mgt.rsn.version

wlan_mgt.rsn.gcs.type

wlan_mgt.rsn.pcs.count

wlan_mgt.rsn.akms.count

wlan_mgt.rsn.akms.type

wlan_mgt.rsn.capabilities.preauth

wlan_mgt.rsn.capabilities.no_pairwise

wlan_mgt.rsn.capabilities.ptksa_replay_counter

wlan_mgt.rsn.capabilities.gtksa_replay_counter

wlan_mgt.rsn.capabilities.mfpr

wlan_mgt.rsn.capabilities.mfpc

wlan_mgt.rsn.capabilities.peerkey

wlan_mgt.tcprep.trsmt_pow

wlan_mgt.tcprep.link_mrg

wlan.wep.iv

wlan.wep.key

wlan.wep.icv

wlan.tkip.extiv

wlan.ccmp.extiv

wlan.qos.tid

wlan.qos.priority

wlan.qos.eosp

wlan.qos.ack

wlan.qos.amsdupresent

wlan.qos.buf_state_indicated

wlan.qos.bit4	data.len
wlan.qos.txop_dur_req	
wlan.qos.buf_state_indicated	class