ANÁLISE DE ALGORITMO PARA DETECÇÃO DE PONTOS DE MUDANÇA EM SÉRIES TEMPORAIS VISANDO A DETECÇÃO DE ATAQUES DDOS

Ana Paula Rocha Soares Passos

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica da Universidade Federal do Rio de Janeiro como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Orientador: Rosa Maria Meri Leão

Rio de Janeiro

Março de 2019

ANÁLISE DE ALGORITMO PARA DETECÇÃO DE PONTOS DE MUDANÇA EM SÉRIES TEMPORAIS VISANDO A DETECÇÃO DE ATAQUES DDOS

Ana Paula Rocha Soares Passos

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinadores:

_____

Profa. Rosa Maria Meri Leão, Dr.

_____

Prof. Edmundo Albuquerque de Souza e Silva, Ph.D.

_____

Prof. Daniel Sadoc Menasche, Ph.D.

RIO DE JANEIRO, RJ - BRASIL

MARÇO DE 2019

Rocha Soares Passos, Ana Paula

Análise de Algoritmo para Detecção de Pontos de Mudança em Séries Temporais Visando a Detecção de Ataques DDoS/Ana Paula Rocha Soares Passos. – Rio de Janeiro: UFRJ/POLI – COPPE, 2019.

ix, 33 p.: il.; 29, 7cm.

Orientador: Rosa Maria Meri Leão

Projeto (graduação) – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2019.

Referências Bibliográficas: p. 32 – 33.

1. Detecção de Pontos de Mudanças. 2. Ataques DDoS. 3. Modelo de Auto Regressão. 4. *Likelihood Ratio*. 5. Séries Temporais. I. Maria Meri Leão, Rosa. II. Universidade Federal do Rio de Janeiro, Escola Politécnica/ Curso de Engenharia de Computação e Informação. III. Título

Agradecimentos

Aos meus pais, Patrícia e Eduardo, pelo amor incondicional, pelas lições de vida e por estarem presentes em todas as minhas conquistas, serei eternamente grata por ter vocês como meus pais.

Ao meu esposo e melhor amigo, Victor, por ter sido o maior apoiador desse projeto de vida de me tornar Engenheira.

Aos meus avós, Francisco e Cleide, por todas as orações, carinho e zelo, vocês são como anjos da guarda na minha vida.

Aos meus sogros, Josué e Selma, por terem sido minha segunda família por todos esses anos e aos quais eu tenho tanto respeito e admiração.

Às minhas irmãs, Priscilla, Suyan, Taynar e Maria Eduarda, por serem minha maior fonte de recordações, vocês me motivam, inconscientemente, a buscar pelo melhor que posso me tornar como ser humano.

Às minhas tias, Helda e Tereza, por terem abraçado esse sonho junto comigo e terem me oferecido tamanha ajuda por vários anos, essa conquista também é de vocês.

Obrigada família, sem vocês eu não teria chegado até aqui.

Ao meu amigo Anderson, por toda ajuda e parceria nos trabalhos.

À minha orientadora, Rosa Leão, pelo acompanhamento e colaboração semanal através de sugestões de literatura, comparação de resultados e análise dos problemas discutidos.

À Universidade Federal do Rio de Janeiro, por ter sido uma verdadeira escola da vida, que me proporcionou tanto crescimento pessoal e profissional.

Resumo do Projeto de Graduação apresentado à Escola Politécnica/COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

ANÁLISE DE ALGORITMO PARA DETECÇÃO DE PONTOS DE MUDANÇA EM SÉRIES TEMPORAIS VISANDO A DETECÇÃO DE ATAQUES DDOS

Ana Paula Rocha Soares Passos

Março/2019

Orientador(a): Rosa Maria Meri Leão
Curso: Engenharia de Computação e Informação

Séries temporais são dados especialmente importantes pois são uma coleção de observações feitas sequencialmente ao longo do tempo e cujas observações vizinhas são dependentes entre si, fazendo-se necessário analisar e modelar essa dependência. Este trabalho apresenta uma análise algorítmica dos resultados encontrados ao submeter séries temporais não estacionárias a uma formulação estatística típica de detecção de ponto de mudança e analisar as distribuições de probabilidade de dados antes e depois de um ponto de mudança candidato e identificar o candidato como um ponto de mudança se as duas distribuições forem significativamente diferentes. O método ajusta um modelo de regressão automática aos dados para representar o comportamento estatístico da série temporal e atualiza as estimativas dos parâmetros de forma incremental, de modo que o efeito dos exemplos passados seja gradualmente descontado. Uma forma de avaliar a probabilidade de um determinado dado da sequência ser um dado "normal" ou um dado suspeito. Por fim, a avaliação dos resultados por métricas de desempenho, como Precisão e Sensibilidade, utilizadas para avaliar o algoritmo na detecção de pontos de mudança dos dados que sofreram ataques de negação de serviço distribuído.

Palavras-Chave: Séries Temporais, Detecção de ataques DDoS, Pontos de Mudança, *Outliers*, modelo auto regressivo, *Score*.

Abstract of the Undergraduate Project presented to Poli/COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

ALGORITHM ANALYSIS FOR CHANGE POINTS DETECTION IN TEMPORARY SERIES FOR DDOS ATTACKS

Ana Paula Rocha Soares Passos

Março/2019

Advisor: Rosa Maria Meri Leão

Course: Computer and Information Engineering

Time series are especially important because they are a collection of observations made sequentially over time and whose neighboring observations are dependent on each other, making it necessary to analyze and model this dependency. This paper presents an algorithmic analysis of the results found by subjecting non-stationary time series to a typical statistical point-of-change detection formulation and analyzing the probability distributions of data before and after a candidate change point and identifying the candidate as a point change if the two distributions are significantly different. The method adjusts an automatic regression model to the data to represent the statistical behavior of the time series and updates the parameter estimates incrementally, so that the effect of the past examples is gradually discounted. One way of evaluating the probability that a given sequence data is a "normal" die or a suspect die. Finally, the evaluation of the results by performance metrics, such as Precision and Sensitivity, used to evaluate the algorithm in the detection of data change points that suffered denied distributed service attacks.

Keywords: Time Series, DDoS attacks detection, Change points, Outliers, Auto regression model, Score.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1 – Introdução

1.1 – Motivação

O processo de minerar dados para descobrir conexões escondidas e prever tendências futuras tem uma longa história. Por vezes chamado de "descoberta de conhecimento em bancos de dados" [4]. Sua base compreende três disciplinas científicas entrelaçadas: Estatística, ciência que se utiliza das teorias probabilísticas para explicar a frequência da ocorrência de eventos, análise e interpretação de dados [15], Inteligência Artificial, inteligência similar à humana exibida por *softwares* e/ou máquinas [16] e Aprendizado de Máquina, estudo de reconhecimento de padrões e da teoria do aprendizado computacional, que são algoritmos que podem aprender com dados para realizar previsões [17].

As áreas de atuação da mineração são as mais diversas possíveis, indo desde o estudo de regiões de alteração genética em pesquisas sobre o câncer, pontos de mudança do número de ciclones tropicais anuais até monitoramento de falhas na rede.

O aumento exponencial do número de dispositivos IoT (*Internet of Things*), as redes sociais, o *Youtube* (banco de dados de vídeos), Wikipédia (maior enciclopédia da internet), como também os registros de compra e venda, os canais de interação não digital (*telemarketing* e *call center*) etc., tudo contribuiu para o surgimento do processo de análise e interpretação de todo o volume de dados gerados e armazenados remotamente, ou seja, o surgimento do *Big Data* (BD).

Minerar dados nada mais é que o processo de explorar grandes quantidades de dados (BD) à procura de padrões consistentes, como regras de associação ou sequências temporais, para detectar relacionamentos sistemáticos entre variáveis, detectando assim novos subconjuntos de dados [14].

De posse dos dados e do processo de como trabalhar, é evidente reconhecer a importância de saber lidar de maneira científica com as informações coletadas, a fim de garantir resultados confiáveis e que auxiliem na tomada de decisões com maior precisão, que é a finalidade deste trabalho.

1.2 – Objetivo

O objetivo do trabalho é detectar um ataque DDoS através da análise de dados de tráfego, perda e latência coletados nos roteadores residenciais dos usuários.

Usaremos um algoritmo da literatura cujo objetivo é a detecção de pontos de mudança em séries temporais não estacionárias.

Este trabalho, faz parte do projeto INSaNE (*Improving Network Security at the Network Edge*), projeto da UFRJ em parceria com o provedor de internet Gigalink, com a startup Anlix, com a Universidade Federal de Belém e com a University of Massachussets at Amherst.

Também fará parte do trabalho a completa implementação do algoritmo SDAR (*Sequentially Discounting Auto Regression model learning*), que foi dividia em duas partes: a parte de implementação do modelo auto regressivo para representar o comportamento estatístico dos dados das séries temporais e a parte de *Scoring*, também chamado de perda logarítmica.

Para a análise, o objetivo foi usar dados de usuários domésticos com detecção de pontos de mudança na borda, usando apenas informações de contagens de bytes, cujas métricas da rede analisadas foram as de tráfego de Upload, Latência e Perda, para 3 diferentes *datasets* de diferentes usuários do provedor de internet.

A análise qualitativa foi desenvolvida por métricas de desempenho importantes empregadas para avaliar algoritmos de detecção de pontos de mudança, que são as métricas de Precisão e Sensibilidade, com o objetivo de obter valores maiores ou iguais a 80% de ambas as métricas.

Por fim, os parâmetros presentes no algoritmo foram amplamente modificados para cada *dataset*, cada métrica e para cada tipo de ataque específico em que o objetivo de desempenho não foi o desejado com a finalidade de alcança-lo sem provocar overfitting.

1.3 – Metodologia

Para o desenvolvimento deste trabalho foi necessária a realização de uma pesquisa bibliográfica para a escolha de algoritmos que atendessem às necessidades da proposta final do trabalho, que é a de detectar e analisar pontos de mudança em séries temporais não estacionárias.

O algoritmo escolhido foi implementado na linguagem de programação Python, usando algumas bibliotecas básicas no desenvolvimento do código tais como: NumPy e SciPy e, para os códigos secundários de plotagem dos gráficos, manipulação e análise dos dados, foram necessárias as bibliotecas: Matplotlib e Pandas.

Em uma primeira etapa, o código desenvolvido foi testado usando dados artificiais propostos pelo próprio autor do algoritmo escolhido. A finalidade foi verificar se iríamos obter os mesmos resultados do autor do algoritmo, o que significa que a implementação estaria correta.

Em uma segunda etapa, usamos o algoritmo para analisar o tráfego de upload dos usuários com a finalidade de detectar possíveis ataques DDoS. Realizamos uma análise visual do *plot* de alguns tipos de gráficos visando avaliar a eficiência do algoritmo para dados reais. Calculamos também algumas métricas que permitiram realizar uma análise quantitativa do desempenho do algoritmo. As métricas obtidas foram Precisão e Sensibilidade.

1.4 – Organização do Trabalho

Além desta introdução, o trabalho é organizado como segue.

O Capítulo 2, apresenta alguns conceitos importantes e necessários que foram usados ao longo desse trabalho.

O Capítulo 3 apresenta uma breve descrição de todos os experimentos realizados, o detalhamento do Algoritmo usado e uma breve descrição das métricas escolhidas para a avaliação dos resultados encontrados.

O Capítulo 4 apresenta os resultados preliminares e a análise do Algoritmo.

O Capítulo 5 apresenta algumas limitações encontradas, assim como as considerações finais.

Capítulo 2 – Revisão da Literatura

2.1 – Séries Temporais Não Estacionárias

Uma série temporal é uma coleção de observações feitas sequencialmente ao longo do tempo [12]. E para ser classificada como estacionária, é necessário que a média seja constante ao longo do tempo, refletindo alguma forma de equilíbrio estável. As séries temporais não estacionárias, em contrapartida, não possuem uma média constante ao longo do tempo. Assim, uma série não estacionária é aquela cujas propriedades estatísticas mudam com o tempo.

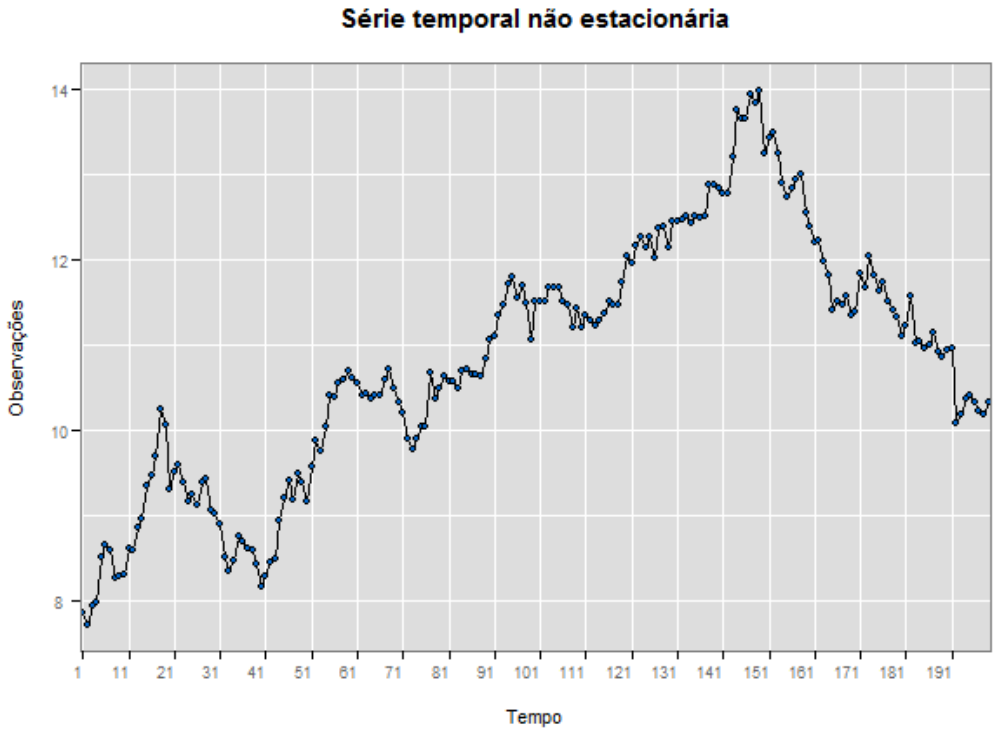

Figura 1 - Exemplo de série temporal não estacionária. Fonte: [11]

O efeito não estacionário pode ser produzido em séries temporais por tendências, sazonalidade, assim como correlações entre séries. Muitas vezes é interessante "converter" uma série temporal não estacionária em uma série estacionária (por exemplo, por remoção de tendência). Um exemplo onde é interessante remover a tendência é no caso onde a média aumenta consistentemente ao longo do tempo. Nesse

caso, a média e a variância das amostras aumentarão com o número de amostras, e sempre subestimarão a média e a variância em períodos futuros.

A maioria dos métodos de previsão estatística baseia-se no pressuposto de que as séries temporais são aproximadamente estacionárias [13]. Estudar séries não estacionárias é especialmente importante pois muitas séries temporais econômicas e comerciais são não estacionárias. Neste trabalho consideramos séries temporais não estacionárias que representam o tráfego de upload do usuário. A série temporal possui intervalos onde são coletadas somente amostras do tráfego de upload normal do usuário e intervalos que representam amostras de tráfego de upload normal somado ao tráfego de ataque DDoS.

2.2 – Pontos de Mudança

A detecção de ponto de mudança (CPD) é o problema de encontrar mudanças nos dados quando uma propriedade estatística da série temporal muda [9]. Segmentação, detecção de bordas, detecção de eventos e detecção de anomalias são alguns conceitos similares que costumam ser aplicados de forma ocasional, bem como a detecção de pontos de mudança.

A detecção do ponto de mudança está intimamente relacionada ao problema de estimativa de ponto de mudança ou mineração de ponto de mudança. Diferentemente do CPD, no entanto, a estimativa de ponto de mudança tenta modelar e interpretar mudanças conhecidas em séries temporais em vez de identificar que uma mudança ocorreu [1]. O foco das estimativas de ponto de mudança é descrever a natureza e o grau da mudança conhecida. O CPD tem sido estudado nas últimas décadas nas áreas de mineração de dados, estatística e ciência da computação. E esse problema abrange uma ampla gama de problemas do mundo real.

2.3 – Métodos Não Supervisionados para detecção de pontos de mudança

Em métodos não supervisionados, nenhum tipo de etiqueta é dado ao algoritmo de aprendizado, ficando sob responsabilidade do próprio algoritmo encontrar alguma estrutura ou padrões de comportamento através das entradas não rotuladas fornecidas, e somente depois, o usuário pode avaliar e tentar atribuir um rótulo aos padrões

encontrados. Dessa forma, é possível descobrir similaridades e diferenças entre os padrões encontrados, como também derivar conclusões úteis a respeito desses dados. O maior desafio de todo o processo é que rotular um grande conjunto de dados pode custar muito tempo, esforço computacional e dinheiro.

No contexto da detecção de pontos de mudança, tais algoritmos podem ser usados para segmentar dados de séries temporais, encontrando assim pontos de mudança com base em características estatísticas dos dados. A segmentação não supervisionada é atraente porque pode lidar com uma variedade de situações diferentes sem exigir treinamento prévio para cada situação. Na literatura, existem diversos métodos não supervisionados que são usados para detecção de pontos de mudança, tais como o Likelihood Ratio, Probabilistic Methods, Kernel Based Methods, Graph Based Methods e Clustering.

Entre os métodos citados, a razão de verossimilhança (*Likelihood Ratio*) é aplicada com base na observação de que a densidade de probabilidade de dois intervalos consecutivos é a mesma se eles pertencem ao mesmo estado [1].

Os métodos probabilísticos (*Probabilistic Methods*) estimam distribuições de probabilidade do novo intervalo com base nos dados que foram observados desde o ponto de mudança candidato anterior [1].

Em contraste, os métodos kernel (*Kernel Based Methods*) são uma classe de algoritmos para análise de padrões, cujo membro mais conhecido é a máquina de vetores de suporte (SVM). A tarefa geral da análise de padrões é encontrar e estudar tipos gerais de relações (por exemplo, clusters, classificações, componentes principais, correlações, classificações) em conjuntos de dados [19]. Esses métodos mapeiam as observações em um espaço de características de dimensão mais alta e detectam pontos de mudança comparando a homogeneidade de cada subsequência [1].

A técnica baseada em grafos (Graph Based Methods) é um método recém-introduzido que representa observações de séries temporais como um grafo e aplica testes estatísticos para detectar pontos de mudança com base nessa representação.

Por fim, o método de agrupamento (*Clustering)* é a tarefa de agrupar um conjunto de objetos de forma que os objetos no mesmo grupo (chamados de cluster) sejam mais semelhantes (em algum sentido) uns aos outros do que àqueles em outros clusters. É uma tarefa importante de mineração de dados exploratória e uma técnica comum para análise de dados estatísticos. Esses métodos agrupam os dados das séries temporais em

seus respectivos estados e localizam as alterações identificando as diferenças entre os recursos dos estados [1].

Dentre as várias opções de métodos não supervisionados para detecção de change points, este trabalho entrará em detalhes somente em relação ao método *Likelihood Ratio*, que foi o método escolhido para aplicação.

2.3.1 - *Likelihood Ratio*

Uma forma de detectar um ponto de mudança é analisar as distribuições de probabilidade de dados antes e depois de um ponto de mudança candidato e identificar o candidato como um ponto de mudança se as duas distribuições forem significativamente diferentes. Nessas abordagens, o logaritmo da razão de verossimilhança entre dois intervalos consecutivos de uma série temporal é monitorado para detectar pontos de mudança. Esta estratégia requer dois passos. Primeiro, a densidade de probabilidade de dois intervalos consecutivos é calculada separadamente. Em segundo lugar, a razão dessas densidades de probabilidade é calculada. Um dos algoritmos de ponto de mudança mais usados é a soma cumulativa [5][6]. O algoritmo acumula desvios em relação a um alvo específico de medições de entrada e indica que existe um ponto de mudança quando a soma cumulativa excede um determinado limite [1].

O *Change Finder* [3] é outro método comumente utilizado que realiza a detecção de ponto de mudanças de forma simples e baseado em séries temporais não estacionárias. Esse método ajusta um modelo auto regressivo (AR model) aos dados para representar o comportamento estatístico da série temporal. Ele atualiza as estimativas dos parâmetros de forma incremental, de modo que o efeito das amostras em instantes de tempo anteriores a *t*, tenham menor importância à medida que o tempo *t* avança. Para uma série temporal x_t, é possível modelar a série temporal usando um modo AR de k-ésima ordem dado por:

$$x_t = \omega x_{t-k}^{t-1} + \mathcal{E}$$

onde $x_{t-k}^{t-1} = (x_{t-1}, x_{t-2}, \dots, x_{t-k})$ são observações anteriores, $\omega = (\omega_1, \dots, \omega_k) \in \mathbb{R}^k$ são constantes, e $\mathscr{E}$ é uma variável aleatória normal gerada de acordo com a distribuição Gaussiana com ruído branco. Ao atualizar os parâmetros do modelo,

a função de densidade de probabilidade no tempo t é calculada e temos uma sequência de densidades de probabilidade $\{p_t: t = 1,2,\dots\}$.

Em seguida, é gerada uma série temporal auxiliar y_t, dando um *score* a cada ponto de dados. Esta função de Score é definida como a média da log-verossimilhança, $Score\,(y_t) = -logp_{t-1}\,(y_t)$ ou desvio estatístico, $Score\,(y_t) = d\,(p_{t-1}\,, p_t)$, onde $d\,(*\,,*)$ é fornecido por qualquer uma das várias funções de distância da literatura, incluindo a distância de variação, a distância de Hellinger ou a distância quadrática. Os novos dados da série temporal representam a diferença entre cada par de intervalos consecutivos de séries temporais. Para detectar pontos de mudança, precisa saber se há mudanças abruptas entre duas diferenças consecutivas. Para fazer isso, mais um modelo AR é adequado para a série temporal baseada em diferenças e uma nova sequência de funções de densidade de probabilidade $\{q_t: t = 1,2,\dots\}$ é construída. O *Change-Point Score* é definido usando a função de $Score\,(y_t)$ mencionada. Um *score* maior indica uma maior possibilidade de ser um ponto de mudança.

2.4 – Métodos de Avaliação de Desempenho usados no Algoritmo de Detecção de Pontos de Mudança

Para avaliar o desempenho do algoritmo, iremos comparar os instantes de tempo que foram identificados pelo algoritmo como pontos de mudança com os instantes de tempo onde sabemos que ocorreu um ataque. No capítulo 3 iremos detalhar como as séries temporais de tráfego de upload com ataques foram geradas.

Usaremos a matriz de confusão para avaliar o algoritmo.

No campo da aprendizagem de máquina e especificamente em problemas de classificação estatística, uma matriz de confusão, também conhecida como matriz de erro (ou matriz de correspondência, como é chamado em aprendizagem não supervisionada) é um *layout* de tabela específico que permite a visualização do desempenho de um algoritmo. Cada linha da matriz de confusão representa as instâncias em uma classe prevista, enquanto cada coluna representa as instâncias em uma classe real (ou vice-versa) [7].

	Condição Positiva	Condição Negativa
Predição Condição Positiva	Verdadeiro Positivo (TP)	Falso Positivo (FP)
Predição Condição Negativa	Falso Negativo (FN)	Verdadeiro Negativo (TN)

Figura 2 - Representação de uma matriz de confusão

De posse dos números de TP, FP, FN e TN é possível uma análise mais detalhada do que uma mera proporção de classificações corretas de um conjunto de dados. Algumas das métricas de desempenho úteis que são importantes empregar para avaliar algoritmos usando análise supervisionada são descritas abaixo.

- Precisão, que é calculada como a proporção de pontos de dados positivos reais (pontos de mudança) para o total de pontos classificados como pontos de mudança. Ou seja, daqueles classificados como corretos, quantos deles são efetivamente corretos.

$$Precisão = \frac{TP}{TP + FP}$$

- Sensibilidade, também conhecida como recordação ou revocação ou taxa positiva verdadeira (Taxa TP), se refere à parte de uma classe de interesse (pontos de mudança) que foi reconhecida corretamente. Ou seja, significa a porcentagem que o classificador, o algoritmo, está identificando corretamente.

$$Sensibilidade = Recall = Taxa\,TP = \frac{TP}{TP + FN}$$

2.5 – Ataques DDoS

O ataque de negação de serviço distribuído (ou DDoS – *Distributed Denial of Service*) é uma tentativa mal-intencionada de interromper o tráfego normal de um servidor, serviço ou rede direcionados, sobrecarregando o alvo ou sua infraestrutura circundante com uma enxurrada de tráfego da Internet. Esses ataques atingem o seu objetivo, utilizando vários dispositivos conectados à Internet comprometidos como fontes de tráfego de ataque. Os dispositivos explorados podem incluir computadores, aparelhos telefônicos e outros recursos da rede, como dispositivos IoT (*Internet of Things*).

Um ataque DDoS requer que um invasor controle a rede de dispositivos on-line para realizar um ataque. Esses dispositivos são infectados com malware, transformando cada um em um *bot* (ou zumbi). O invasor, dessa forma, tem o controle remoto sobre o grupo de *bots*, que é chamado de *botnet*.

Uma vez que uma botnet tenha sido estabelecida, o atacante é capaz de direcionar as máquinas enviando instruções atualizadas para cada bot através de um método de controle remoto. Quando o endereço IP de uma vítima é visado pela botnet, cada bot responderá enviando solicitações ao destino, fazendo com que o servidor ou a rede segmentada atinja a capacidade máxima, resultando em negação de serviço para o tráfego normal. Como cada bot é um dispositivo de Internet legítimo, separar o tráfego de ataque do tráfego normal pode ser uma tarefa muito difícil.

Por esse motivo, os ataques DDoS têm sido motivo de muita preocupação entre usuários, empresas e governos. O aumento exponencial do números de dispositivos IoT, que são particularmente vulneráveis, uma vez que normalmente são instalados sem atualizações de *firmware* e com parâmetros *default* mantidos, sendo alvo fácil de *backdoors* e *exploits* com o objetivo de criar grandes *botnets* [10], tem propiciado o aumento da pesquisa nessa área. A grande quantidade desses tipos de ataques ultimamente, tem provocado desafios para a criação de novas abordagens em relação a detecção desses ataques e uma possível diminuição nas suas consequências.

2.6 – Overfitting

É um termo encontrado na literatura para descrever quando um modelo estatístico se ajusta muito bem ao conjunto de dados de treinamento, mas que afeta negativamente o desempenho do modelo em novos dados e, portanto, pode falhar em ajustar dados adicionais ou prever observações futuras de forma confiável. Isso significa

que o ruído ou as flutuações aleatórias nos dados de treinamento são captados e aprendidos como conceitos pelo modelo. O problema é que esses conceitos não se aplicam a novos dados e afetam negativamente a capacidade de generalização dos modelos.

O *overfitting* é mais provável com modelos não paramétricos e não lineares que têm mais flexibilidade ao aprender uma função alvo. Como tal, muitos algoritmos de aprendizado de máquina não paramétricos também incluem parâmetros ou técnicas para limitar e restringir a quantidade de detalhes que o modelo aprende. Por exemplo, as árvores de decisão são um algoritmo de aprendizado de máquina não paramétrico que é muito flexível e está sujeito a sobrecarregar os dados de treinamento. Esse problema pode ser resolvido podando-se uma árvore depois de ela ter aprendido para remover alguns dos detalhes que ela pegou.

A essência do *overfitting* é ter extraído inconscientemente alguma da variação residual (ou seja, o ruído) como se essa variação representasse a estrutura do modelo subjacente, o que pode levar a um desempenho pobre do modelo.

O *overfitting* é um problema porque a avaliação de algoritmos de aprendizado de máquina em dados de treinamento é diferente da avaliação que mais interessa, que é saber quão bem o algoritmo se comporta em dados não vistos.
Existem duas técnicas importantes que podem ser usadas para avaliar os algoritmos de aprendizado de máquina para limitar esse problema:

- Técnica de reamostragem para estimar a precisão do modelo
- Reter um conjunto de dados de validação

A técnica de reamostragem mais popular é a validação cruzada k-*fold*. Essa técnica permite que haja treinamento do modelo k vezes em diferentes subconjuntos de dados de treinamento e crie uma estimativa do desempenho de um modelo de aprendizado de máquina em dados não vistos. Um conjunto de dados de validação é simplesmente um subconjunto dos dados de treinamento que foi retido do algoritmo de aprendizado de máquina até o final do projeto. Depois de selecionar e ajustar o algoritmo de aprendizado de máquina em um determinado conjunto de dados de treinamento, é possível avaliar os modelos aprendidos no conjunto de dados de validação para obter uma ideia objetiva final de como os modelos podem executar em dados não vistos.

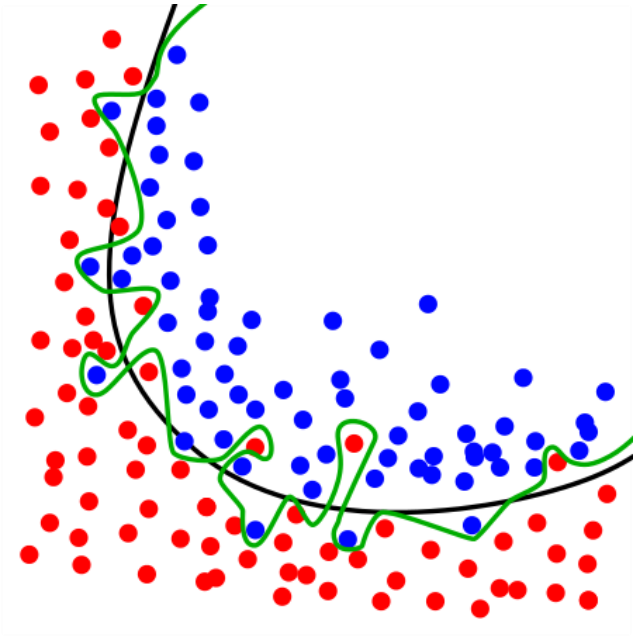

Figura 3 - A linha verde representa um modelo *overfitted* e a linha preta representa um modelo regularizado. Embora a linha verde siga melhor os dados de treinamento, ela é muito dependente desses dados e provavelmente terá uma taxa de erros maior em novos dados não vistos, em comparação à linha preta. Fonte: [8]

Capítulo 3 – Método

3.1 - Experimentos usados para emulação de ataques em ambiente real

A emulação dos ataques foi feita pelos autores do artigo "Uma abordagem leve para detecção de DDoS a partir de roteadores domésticos" [2]. Nesta seção, será explicado de forma resumida como foi feita a emulação dos ataques em ambiente real.
residências colocando um *raspberry pi* conectado ao roteador doméstico do usuário através do *Wifi*. Foram instalados no *raspberry pi*, os softwares Mirai e BASHLITE para gerar ataques em janelas de tempo aleatórias. Dessa forma, as medidas de latência, perda e o tráfego de upload normal do usuário juntamente com o tráfego de ataques são coletados no roteador doméstico. Com essa metodologia para emulação de ataques, é possível identificar as janelas de tempo onde ocorreu o ataque e comparar com os instantes de tempo indicados pelo algoritmo como ponto de mudança.

Os dados de tráfego, latência e perda obtidos, foram coletados de roteadores domésticos de 3 residências. As datas selecionadas para serem usadas neste trabalho foram de 13 de outubro a 26 de novembro de 2018. Cabe ressaltar que nenhuma informação do cabeçalho ou *payload* dos pacotes foi coletada, preservando a privacidade do usuário. A única informação coletada foi o número de bytes de upload transmitidos pelo roteador do usuário em cada minuto.

Os dados foram coletados a partir de roteadores *wireless* que servem como *gateways* domésticos para o provedor de Internet. Os roteadores executam *OpenWrt*, usando *software* de código aberto para coletar e enviar informações para um servidor.

3.2 - Algoritmo usado para detecção dos ataques

- Modelo de Séries Temporais

O Algoritmo usado foi retirado do artigo "A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Time Series Data" [3].

O *framework* possui 2 objetivos principais: o de lidar com dados de séries temporais e o de detectar pontos de mudança em um fluxo de dados. Para o primeiro objetivo, foi proposto um algoritmo para desconto de aprendizagem on-line do modelo AR (auto regressão). Em seguida, foi empregado esse algoritmo para detectar valores

discrepantes de dados de séries temporais. Para o segundo objetivo, foi criado um novo *framework* para detecção de ponto de mudança, conectando-se a detecção de outliers de séries temporais.

O artigo abordou dois tipos de modelos: o modelo onde os dados são desenhados independentemente a cada instante de tempo e o modelo em que uma sequência de dados forma uma série temporal. Porém, este trabalho só abordará e detalhará o segundo modelo, uma vez que os dados utilizados são séries temporais.

O autor emprega o modelo AR para representar o comportamento estatístico dos dados de séries temporais. O modelo AR é um dos modelos mais comuns e mais amplamente explorados em estatística para a representação de séries temporais, inclusive pelo próprio autor. Nele, uma suposta série temporal que, para um valor inicial, sua média é zero, será denotada por $\{z_t: t = 1,2, \dots\}$. O modo AR de ordem *k* pode ser representado por:

$$z_t = \omega z_{t-k}^{t-1} + \mathcal{E}$$

Onde $z_{t-k}^{t-1} = (z_{t-1},\ z_{t-2}, \dots, z_{t-k})$, $\omega = (\omega_1, \dots, \omega_k)\ \epsilon\ \mathbb{R}^k$, onde $\mathcal{E}$ é uma variável aleatória normal gerada pela distribuição Gaussiana de média 0 e matriz de covariância Σ: $\mathcal{N}(0, \Sigma)$. Dada uma série temporal $\{x_t: t = 1, 2, \dots\}$, tal que:

$$x_t = z_t + \mu$$

Onde $x_{t-k}^{t-1} = (x_{t-1}, \dots, x_{t-k})$.

A função densidade de probabilidade de x_t representada pelo modelo AR é dado por:

$$p(x_t | x_{t-k}^{t-1}: \theta) = \frac{1}{(2\Pi)^{\frac{k}{2}} |\Sigma|^{\frac{1}{2}}} \exp(-\frac{1}{2}(x_t - \omega)^T \Sigma^{-1}(x_t - \omega)) \quad (1)$$

Onde $\omega = w(x_{t-k}^{t-1} - \mu) + \mu$. Nós setamos $\theta = (\omega_1, \dots, \omega_k, \mu, \Sigma)$. Antes de detalharmos o algoritmo, se faz necessário estimar os parâmetros do modelo AR:

$$\mu' = \frac{1}{t-k} \sum_{i=k+1}^{t} x_i \quad (2)$$

$$C_j = \frac{1}{t-k}\sum_{i=k+1}^{t}(x_i - \mu)(x_{i-j} - \mu)^T \quad (3)$$

Onde a equação 2 representa uma estimativa de μ e a equação 3 representa uma estimativa de função de covariância de $x_1, \ldots, x_t$. Os coeficientes $\omega_1, \ldots, \omega_k$ do modelo AR são calculados ao solucionar a seguinte equação linear:

$$C_j = \sum_{i=1}^{k} \omega_i C_{j-i} \qquad (j = 1, \ldots, k) \quad (4)$$

Uma vez que a solução da equação acima seja $\omega_1, \ldots, \omega_k$, uma estimativa de Σ é calculada por:

$$\Sigma = C_0 - \sum_{i=1}^{k} \omega_i C_i \quad (5)$$

Para que o algoritmo possa assumir dados não estacionários, que é o caso dos dados coletados neste trabalho, se faz necessário fazer a introdução do SDAR (*sequentially discounting* AR *model learning*). E, para isso, é preciso fazer uma modificação no algoritmo acima, que é a introdução da *discounting property*. A *discounting property* nada mais é que a introdução de um parâmetro *r*, que faz as estatísticas calculadas a partir da série temporal decaírem exponencialmente com um fator multiplicativo (1 - *r*) à medida que o tempo passa.

Pseudocódigo do Algoritmo SDAR (0 < *r* < 1; dado)

Passo 1. Inicialização

Setar $\mu_0', C_j, \omega_j' \ (j = 1, \ldots, k), \Sigma'$.

Passo 2. Atualização dos parâmetros

Para cada intervalo de tempo *t* (= 1, 2, ...), lê x_t, prosseguir:

$$\mu' = (1 - r)\mu' + r x_t$$

$$C_j = (1 - r)C_j + r(x_t - \mu')(x_{t-j} - \mu')^T$$

Resolver a equação:

$$C_j = \sum_{i=1}^{k} \omega_i C_{j-i} \qquad (j = 1, \dots, k) \tag{6}$$

Com os valores da solução da equação acima $(\omega_1, \dots, \omega_k)$, calcular:

$$x_t{}' = \omega'(x_{t-k}^{t-1} - \mu') + \mu'$$

$$\Sigma' = (1 - r)\Sigma + r(x_t - x_t{}')(x_{t-j} - x_t{}')^T$$

Para cada intervalo de tempo t, o algoritmo SDAR atualiza a estimativa dos parâmetros com uma média ponderada, dependendo do parâmetro de desconto r. Quanto menor o valor de r, maior a importância das estatísticas calculadas nos instantes anteriores. A função densidade de probabilidade denotada por p_t é especificada pelas atualizações de parâmetros do algoritmo SDAR no tempo *t*. Então é possível obter uma sequência de densidades de probabilidade: $\{p_t : t = 1, 2, \dots\}$.

- *Scoring*

  Para o cálculo dos *scores*, que é, basicamente, uma forma de avaliar a probabilidade de uma determinada amostra da sequência ser um amostra de tráfego "normal" ou ser uma amostra de tráfego com ataque, o autor apresenta dois tipos de detecção, interligados entre si, que são:

  - Detecção de *Outliers*

    Para cada dado de entrada x_t, calcula-se o *score* de x_t pela fórmula:

$$Score(x_t) = -log p_{t-1}(x_t) \tag{7}$$

    Onde o $Score(x_t)$ é chamado de perda logarítmica. A perda logarítmica pode ser considerada como o comprimento de código necessário para codificar x_t em uma sequência binária sob a suposição de que uma amostra é gerada de acordo com a densidade de probabilidade p_{t-1}.

    Portanto, o *Score* mede o quanto a função densidade de probabilidade p_t mudou com relação a p_{t-1} depois de aprender com x_t. Com isso, é

possível notar que um maior $Score(x_t)$ indica que x_t é um *outlier* com alta probabilidade.

- Detecção de Pontos de Mudança

Dada uma constante positiva *T* e $\{x_t\}$ uma sequência de dados, y_t é definido como o T – pontuação média ao londo de $\{Score(x_i)\text{: i} = \text{t} - \text{T} + 1, \dots, \text{t}\}$, calcula-se por:

$$y_t = \frac{1}{T}\sum_{i=t-T+1}^{t} Score(x_i) \quad (8)$$

Obtém-se uma nova série temporal $\{y_t : t = 1, 2, \dots\}$, que é uma soma de *Scores*. Em seguida, o autor usa o modelo AR para representar a série temporal y_t e emprega o algoritmo SDAR novamente para construir uma sequência de funções de densidade de probabilidade determinadas pelos modelos AR, que será denotada como $\{q_t : t = 1, 2, \dots\}$. Dada uma nova constante positiva *T'*, definida como *T'* – pontuação média no tempo *t*, é possível calcular o *Score* por:

$$Score(t) = \frac{1}{T'}\sum_{i=t-T'+1}^{t}(-lnq_{i-1}(y_i)) \quad (9)$$

Essa nova abordagem reduz o problema de detecção de ponto de mudança para a detecção de valores discrepantes nas séries temporais de *Scores* de média *T*. Com isso, é possível notar que uma pontuação alta de $Score(t)$ indica que o dado no tempo *t* é um ponto de mudança com grande probabilidade.

Para o caso em que *T* é pequeno, *outliers* e pontos de mudança podem ser detectados imediatamente depois de aparecerem, mas podem ser difíceis de discriminar um do outro.

Já para o caso em que *T* é grande, leva um maior tempo de leitura para detectar pontos de mudança. No entanto, os *outliers* são filtrados e somente os pontos de mudança são detectados com precisão.

A figura abaixo ilustra com mais clareza visual todo o processo de funcionamento do algoritmo:

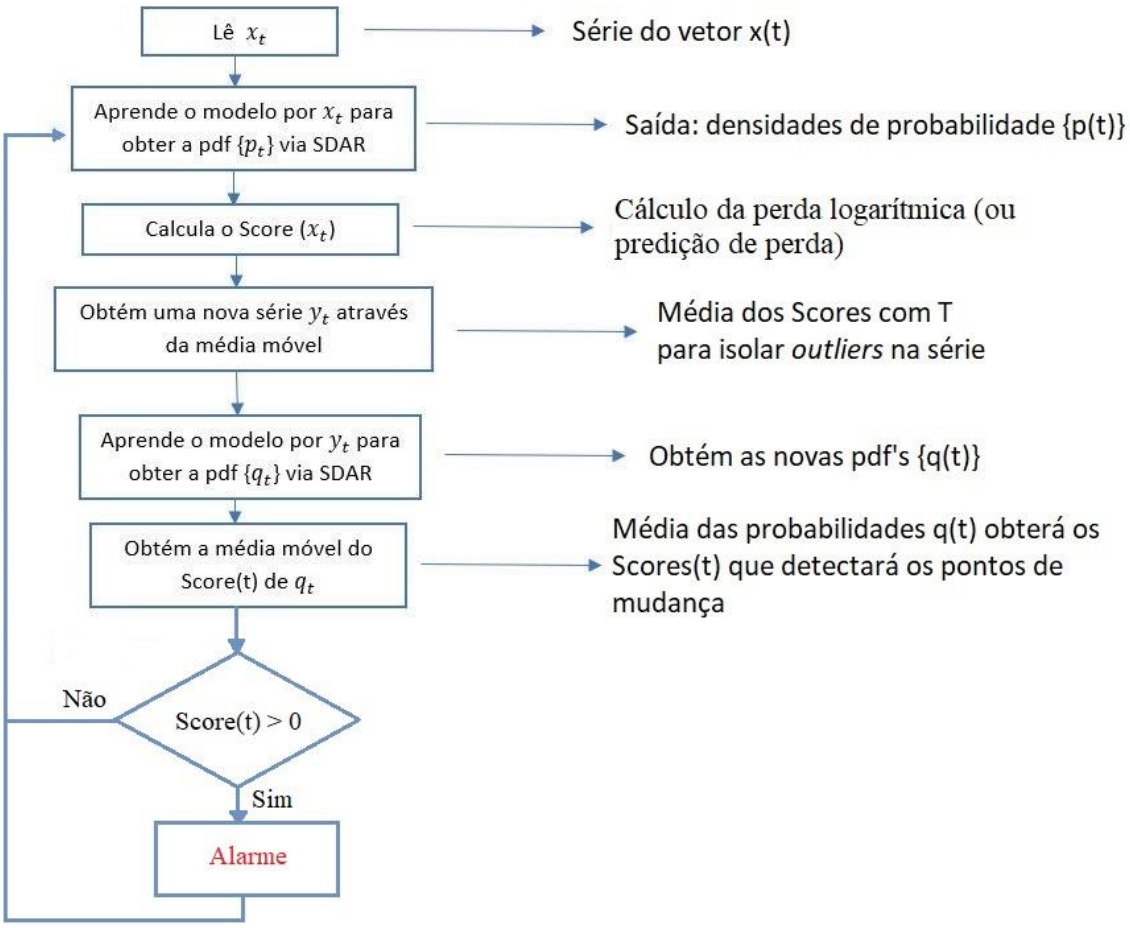

Figura 4 - Ilustração do funcionamento do algoritmo. Fonte: [18]

Capítulo 4 – Resultados

4.1 – Resultados Preliminares com Dados Sintéticos

O primeiro conjunto de dados tem por objetivo testar a implementação do algoritmo. A sequência de amostras foi gerada de acordo com o seguinte modelo AR:

$$x_t = 0.6x_{t-1} - 0.5x_{t-2} + \mathcal{E}_t$$

onde $\mathcal{E}_t$ é a variável aleatória gaussiana com média 0 e variância 1.

Este conjunto de dados consiste em 10.000 registros conforme ilustrado na Figura 4. Os pontos de mudança ocorrem no tempo x x 1.000 (t = 1, 2, ..., 9). Onde a diferença entre o valor do $(x-1)$-ésimo ponto de mudança e o x-ésimo ponto de mudança é $\Delta(x)$, que chamamos de tamanho de mudança em x. Neste caso, definimos $\Delta(x) = x$. O que torna mais fácil detectar pontos de mudança para x grande. Para o modelo e *dataset* escolhidos, foi empregado o modelo AR de ordem $k = 2$ para o cálculo dos *scores*. Aqui foi usada a perda logarítmica para calcular o *score* e o parâmetro de desconto usado no algoritmo SDAR foi $r = 0.005$. Para os demais parâmetros foram usados $T = 5$ e $T' = 5$ para calcular os *scores* correspondentes.

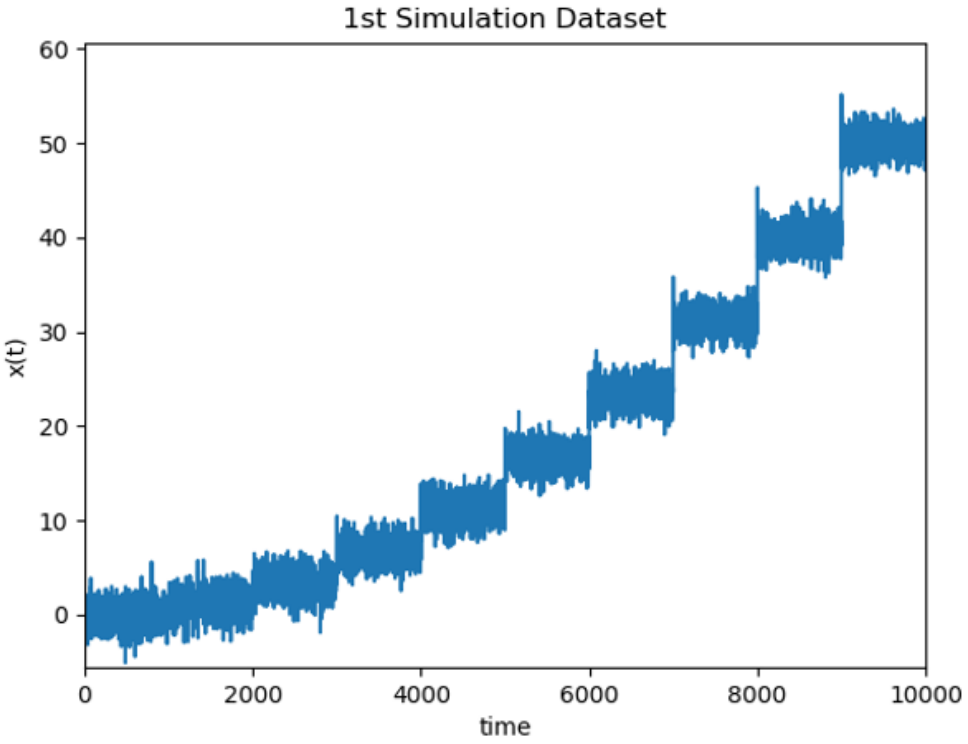

Figura 5 - *Dataset* para simulação

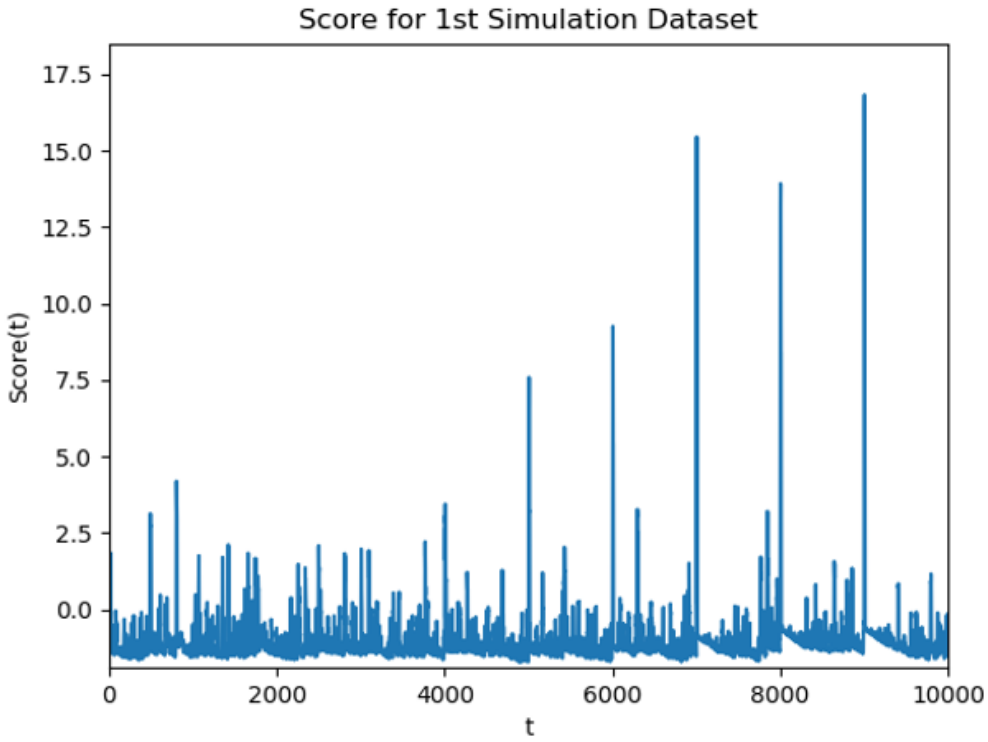

Figura 6 - Detecção de pontos de mudança pelo algoritmo SDAR

Na figura 6 acima, o eixo horizontal mostra o tempo, enquanto o eixo vertical mostra os *scores*. Podemos notar que nos instantes onde ocorreu uma mudança estatística na série temporal da Figura 5, o valor do *score* apresenta um valor bem superior quando comparado aos outros valores calculados para a série. Esse exemplo foi usado para testar a implementação do algoritmo. Verificamos que o resultado obtido com a nossa implementação é igual ao resultado obtido pelos autores do artigo.

4.2 – Análise gráfica dos resultados obtidos com dados coletados em ambiente real

Passaremos agora a analisar os dados coletados em ambiente real. Usamos dados coletados nos roteadores de usuários domésticos conforme descrito no Capítulo 3. Três séries temporais foram usadas no nosso trabalho: (1) o número de bytes de upload transmitidos em 1 minuto, (2) a Latência medida em 1 minuto e (3) a fração de pacotes perdidos em minuto.

Para um mesmo usuário doméstico aleatório (7C-8BCA-D0-6B-EA), foram feitas análises gráficas para comprovar a eficiência do algoritmo na detecção de pontos de mudança nos dados reais coletados.

As Figuras 7 e 8 apresentam o tráfego de upload e os scores estimados pelo algoritmo, respectivamente. Podemos notar que nos instantes que os scores apresentam valores altos, houve uma mudança significativa no tráfego de upload do usuário.

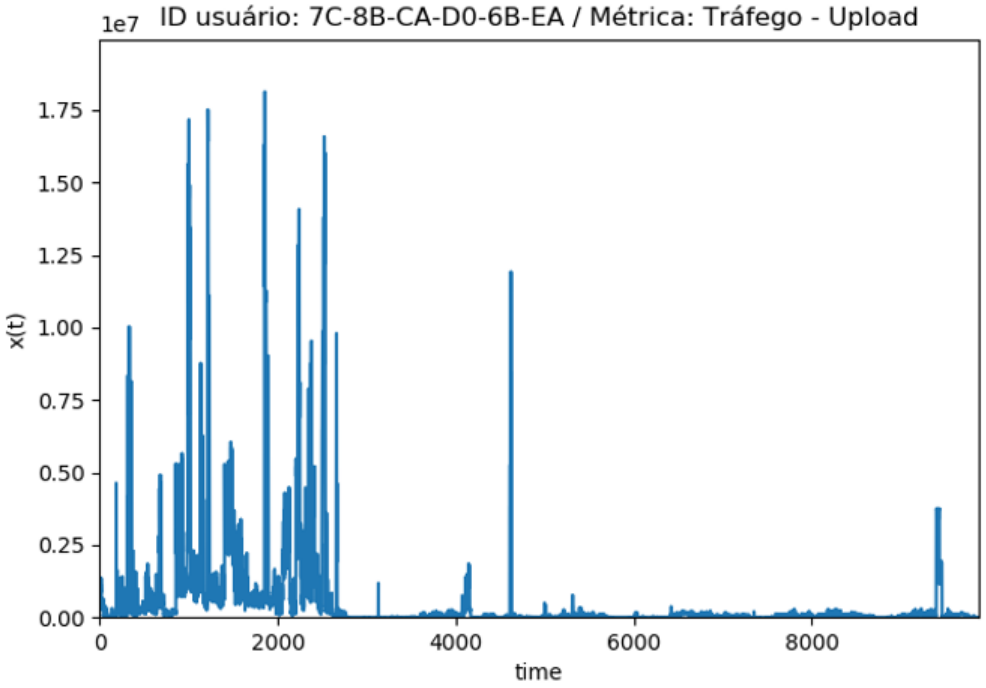

Figura 7 - Tráfego de *Upload* para o usuário aleatório que sofreu ataques DDoS durante a coleta

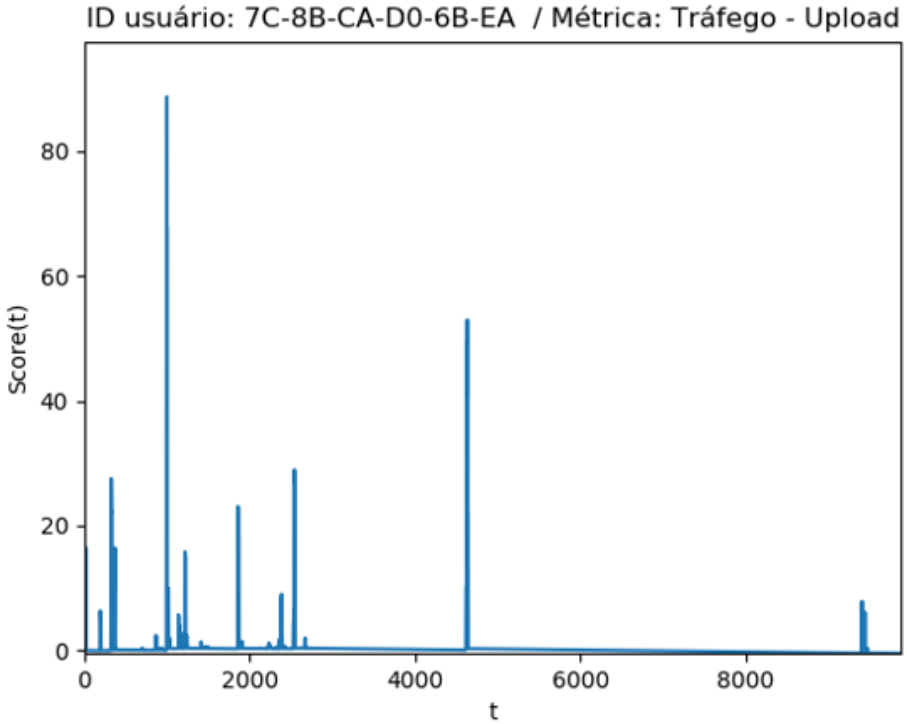

Figura 8 - *Scores* de identificação dos pontos de mudança para o tráfego de *Upload*

As Figuras 9 e 10 apresentam a latência medida e os *scores* calculados pelo algoritmo. É interessante notar que nas janelas de tempo que a latência apresenta valores bem superiores à média, os *scores* também apresentam valores altos.

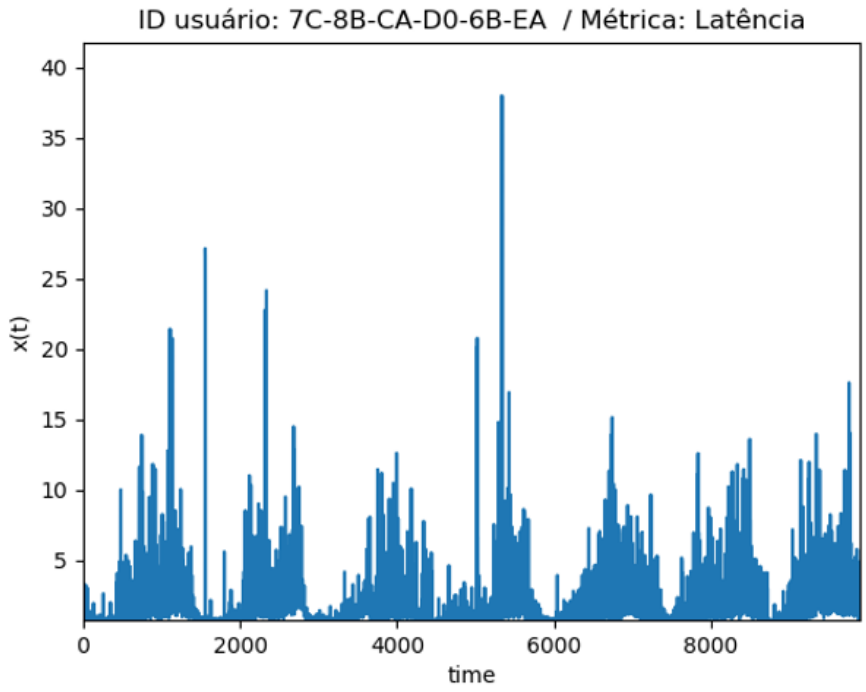

Figura 9 - Latência para o usuário aleatório 7C-8B-CA-D0-6B-EA que sofreu ataques DDoS durante a coleta

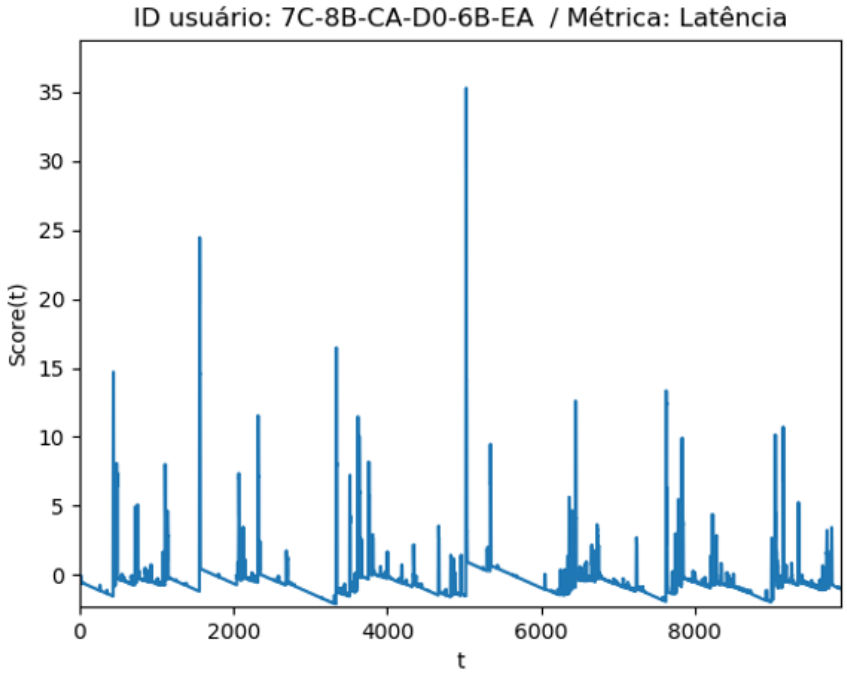

Figura 10 - *Scores* de identificação dos pontos de mudança para a latência

A perda e os respectivos *scores* apresentados nas Figuras 11 e 12 apresentam comportamento semelhante ao tráfego de upload e a latência.

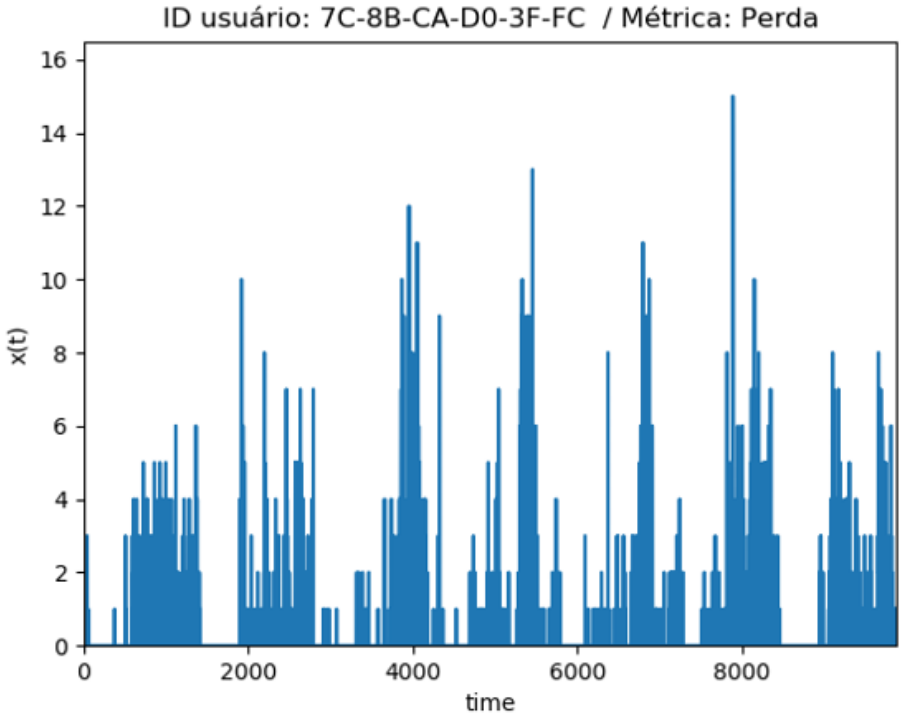

Figura 11 - Perda para o usuário aleatório 7C-8B-CA-D0-6B-EA que sofreu ataques DDoS durante a coleta

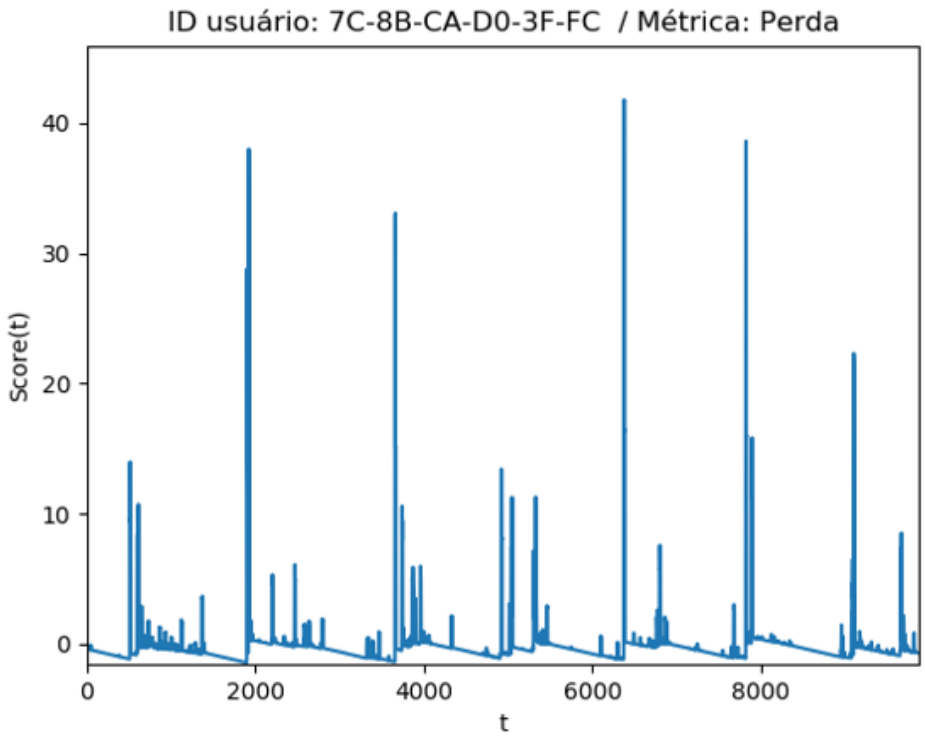

Figura 12 - *Scores* de identificação dos pontos de mudança para a Perda

4.3 – Análise de desempenho do algoritmo para detecção de ataques DDoS usando dados coletados em ambiente real

Para uma análise mais conclusiva e qualitativa do algoritmo em questão, foram usados 3 *datasets* diferentes. Dois desses 3 *datasets* foram de usuários diferentes, mas usando datas de coleta muito próximas uma da outra, e o terceiro foi referente a um mesmo usuário dos já citados, mas referente a datas distantes das já calculadas.

O modelo AR foi empregado para fazer esta análise, e, para isso, os parâmetros foram setados em $k = 4$ para a modelagem de dados e $k = 4$ para o cálculo dos *scores* para os 3 *datasets*. Para a perda logarítmica usada para o cálculo do *score*, os parâmetros de desconto foram setados como $r = 0.005$, $T = 5$ e $T' = 5$, parâmetros usados pelo autor do algoritmo em seu experimento com dados reais [3]. Conforme descrito no algoritmo (Figura 4), quando o valor do *Score(t)* é maior que zero, consideramos uma mudança na série temporal, portanto, um alarme é acionado.

Para a análise de desempenho, foram calculadas as métricas de Sensibilidade e Precisão, explicadas na seção 2.4 deste trabalho. Podemos notar observando as Tabelas 1, 2 e 3 que as métricas de Sensibilidade e Precisão, para alguns tipos de ataque, apresentam valores superiores a 90%, o que indica um bom resultado.

Tabela 1 - Análise qualitativa obtida dos dados do usuário A coletados entre os dias 13 e 23 de outubro de 2018.

	Tipos de Ataques	Sensibilidade	Precisão
Perda	Ataque0	36.00%	41.40%
	Ataque3	_	–
	Ataque4	33.33%	39.60%
	Ataque9	47.00%	48.00%
	Ataque10	_	–
	Ataque50	90.74%	64.08%
	Ataque53	78.60%	60.70%
	Ataque54	48.00%	48.56%
Latência	Ataque0	62.50%	53.40%
	Ataque3	48.65%	47.73%
	Ataque4	89.58%	62.71%
	Ataque9	90.35%	62.92%
	Ataque10	18.52%	25.80%
	Ataque50	90.74%	63.01%
	Ataque53	85.30%	61.56%
	Ataque54	90.00%	62.82%
Upload	Ataque0	_	–
	Ataque3	40.00%	48.86%
	Ataque4	88.89%	67.99%
	Ataque9	_	–
	Ataque10	89.00%	48%
	Ataque50	38.20%	47.70%
	Ataque53	86.80%	67.50%
	Ataque54	34.00%	44.80%

Pôde ser observado na Tabela 1 que, se o alarme for acionado sempre que for encontrado um ponto de mudança, a Sensibilidade será altíssima, mas haverá muitos alarmes falsos. Dessa forma, a precisão cairá muito. Ou seja, é importante encontrar um ponto de equilíbrio em relação ao que é considerado de fato um ponto de mudança.

Tabela 2 - Análise qualitativa obtida dos dados do usuário B coletados entre os dias 13 e 23 de outubro de 2018.

	Tipos de Ataques	Sensibilidade	Precisão
Perda	Ataque0	92.90%	56.00%
	Ataque3	45.40%	38.30%
	Ataque4	37.50%	33.90%
	Ataque9	96.00%	56.80%
	Ataque10	97.00%	57.00%
	Ataque50	96.40%	56.90%
	Ataque53	93.00%	59.80%
	Ataque54	31.70%	30.30%
Latência	Ataque0	93.00%	57.10%
	Ataque3	63.60%	47.70%
	Ataque4	50.00%	41.70%
	Ataque9	16.80%	19.40%
	Ataque10	96.90%	58.10%
	Ataque50	29.70%	29.80%
	Ataque53	28.20%	28.70%
	Ataque54	30.10%	30.10%
Upload	Ataque0	22.50%	35.70%
	Ataque3	84.80%	67.60%
	Ataque4	0%	0%
	Ataque9	31.20%	43.40%
	Ataque10	97.00%	70.50%
	Ataque50	21.00%	34.10%
	Ataque53	19.70%	32.70%

	Ataque54	14.30%	26.00%

Tabela 3 - Análise qualitativa obtida dos dados do usuário A coletados entre os dias 15 e 21 de novembro de 2018.

	Tipos de Ataques	Sensibilidade	Precisão
Perda	Ataque0	44.07%	36.78%
	Ataque3	92.54%	54.99%
	Ataque4	95.19%	55.69%
	Ataque9	28.91%	27.62%
	Ataque10	96.40%	56.00%
	Ataque50	45.77%	37.67%
	Ataque53	48.23%	38.91%
	Ataque54	95.33%	55.73%
Latência	Ataque0	76.12%	52.23%
	Ataque3	41.93%	37.60%
	Ataque4	8.10%	10.43%
	Ataque9	29.27%	29.60%
	Ataque10	24.46%	26.00%
	Ataque50	96.43%	58.07%
	Ataque53	52.94%	43.19%
	Ataque54	95.45%	57.83%
Upload	Ataque0	21.11%	28.08%
	Ataque3	16.45%	23.33%
	Ataque4	96.43%	64.08%
	Ataque9	96.48%	64.09%

	Ataque10	46.89%	46.45%
	Ataque50	95.80%	63.93%
	Ataque53	92.40%	63.09%
	Ataque54	31.49%	36.79%

Capítulo 5 – Conclusão

O algoritmo estudado é um *framework* para detectar *outliers* e pontos de mudança de séries temporais não estacionárias [3]. O algoritmo consiste de duas partes: modelagem de dados e pontuação. Na modelagem de dados, uma função de densidade de probabilidade de uma sequência de dados é aprendida incrementalmente. O algoritmo SDAR possui como propriedade a diminuição gradual da influência dos dados passados. Com isso, é possível lidar com fontes não estacionárias.

Na parte de pontuação, é atribuída uma pontuação para cada amostra ou cada ponto de tempo com base no modelo aprendido, onde foi reduzido o problema de detecção de pontos de mudança para detecção de outliers de séries temporais de pontuações médias móveis.

Apesar do algoritmo apresentar bons resultados para alguns tipos de ataques, encontramos algumas limitações conforme descrevemos a seguir. Testamos o algoritmo com 3 *datasets* diferentes, no entanto, não obtivemos o mesmo desempenho para os três *datasets* e tipos de ataque. As tabelas a seguir ilustram três casos.

No primeiro caso, usando as séries temporais de Perda, referentes a ataques do tipo 50, obtivemos valores de Sensibilidade acima de 90% para os *datasets* 1 e 2, mas obtivemos um valor baixo para o *dataset* 3.

Tabela 4 - Tabela de precisão e sensibilidade de perda sob o ataque 50, referente aos 3 pacotes de dados.

Perda (Ataque 50)	Sensibilidade	Precisão
Dataset 1	90.74%	64.08%
Dataset 2	96.40%	56.90%
Dataset 3	45.77%	37.67%

Já para as séries de Latência, referentes a ataques do tipo 54, obtivemos valores altos de Sensibilidade para os *datasets* 1 e 3.

Tabela 5 - Tabela de precisão e sensibilidade de latência sob o ataque 54, referente aos 3 pacotes de dados.

Latência (Ataque 54)	Sensibilidade	Precisão
Dataset 1	90.00%	62.82%
Dataset 2	30.10%	30.10%
Dataset 3	95.45%	57.83%

Para os dados de tráfego de Upload, referentes à ataques do tipo 53, os melhores valores de Sensibilidade foram obtidos para os *datasets* 1 e 3.

Tabela 6 - Tabela de precisão e sensibilidade dos dados de tráfego de Upload sob o ataque 53, referente aos 3 pacotes de dados.

Upload (Ataque 53)	Sensibilidade	Precisão
Dataset 1	86.80%	67.50%
Dataset 2	19.70%	32.70%
Dataset 3	92.40%	63.09%

A partir da análise desses resultados foram levantadas algumas opções para melhorar o desempenho do algoritmo. Uma primeira opção é variar o parâmetro r. Adotamos o parâmetro $r = 0.005$ conforme adotado pelo autor, no entanto este é um parâmetro empírico. Fizemos alguns experimentos com $r = 0.35$, por exemplo, permanecendo com os valores dos demais parâmetros inalterados, e foi possível notar melhorias na detecção de alguns ataques.

Uma outra possibilidade é coletar mais dados para aumentar o número de *datasets* analisados e com isso ter mais confiança nos resultados obtidos.

Outra observação importante foi que no artigo [3], quando o autor testa a eficiência do algoritmo para dados reais, ele faz uso de séries temporais com pontos de mudança bastante significativos, onde os *scores* apresentados representam o valor total dos ativos na economia do Japão. Os scores de valores altos são de momentos em que ocorreram alterações bastante significativas dos índices. O que levanta a possibilidade de o algoritmo ser mais eficiente quando ocorrem diferenças muito significativas no índice, e não para alterações menos expressivas.

Outra possibilidade na tentativa de avaliar melhor o algoritmo, além dos já citados, é fazer testes com amostras que sofreram outros tipos de ataque que não os DDoS, para tirar melhores conclusões.

- Houve a ocorrência de *Overfitting* ao tentar ajustar os parâmetros do algoritmo com a finalidade de obter melhores valores de Sensibilidade e/ou Precisão.

  Ao reavaliar os dados de tráfego de Upload sob o ataque 53, por exemplo, referente aos 3 pacotes de dados, e modificando o parâmetro *r*, foi possível notar que apesar do valor de *r* = 0.005 obter o melhor resultado de Sensibilidade para os *datasets* 1 e 3, no *dataset* 2 o resultado se mostrou muito inferior, apenas 19.7%.

  Para o valor de *r* = 0.05, os 3 *datasets* apresentaram o melhor valor médio de Sensibilidade e Precisão, o que nos leva a concluir que não há um valor ideal de *r* para todo tipo de *dataset* e que é possível que o bom resultado esteja associado a um possível caso de *overfitting* para um *dataset* específico. Como pode ser observado na tabela abaixo:

Tabela 7 - Tabela de comparação do parâmetro *r* para os *datasets* analisados

Datasets	Valores de r	Sensibilidade	Precisão
Dataset 1	r = 0.005	86.8%	67.5%
	r = 0.05	84.2%	86.3%
	r = 0.5	26.3%	58.6%
Dataset 2	r = 0.005	19.7%	32.7%
	r = 0.05	74.6%	84.8%
	r = 0.5	18.3%	43.8%
Dataset 3	r = 0.005	92.4%	63.08%
	r = 0.05	67.1%	83.3%
	r = 0.5	17.7%	46.8%

Por fim, ressaltamos que os objetivos do trabalho foram cumpridos, já que foi possível fazer uma análise da aplicabilidade de um algoritmo para detectar pontos de mudança em séries não estacionárias na detecção de ataques DDoS.

Referências Bibliográficas

[1] AMINIKHANGHAHI, SAMANEH; COOK, DIANE J., (2016), “A survey of methods for time series change point detection”. Disponível em: < https://www.researchgate.net/publication/307947624_A_Survey_of_Methods_for_Time_Series_Change_Point_Detection>. Acessado em outubro de 2018.

[2] MENDONÇA, GABRIEL; SANTOS, GUSTAVO H. A.; DE SOUZA E SILVA. EDMUNDO; LEÃO, RPSA M. M.; MENASCHÉ, DANIEL S., (), “Uma abordagem leve para detecção de DDoS a partir de roteadores domésticos”. Disponível em: < https://mail.google.com/mail/u/1/#inbox/FMfcgxwBVWFzmdsHzNfBBFtxTfXwbSWV?projector=1&messagePartId=0.1>. Acessado em fevereiro de 2019.

[3] YAMANISHI, KENJI; TAKEUCHI, JUN-ICHI, (2002), “A Unifying Framework for Detecting Outliers and Change Points from Non-Stationary Times Series Data”. Disponível em: < http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.12.3469&rep=rep1&type=pdf >. Acessado em setembro de 2018.

[4] COSTA, CLAUDIO NAPOLIS; COUTINHO, JONATAS VIEIRA; MAGALHÃES, LÚCIA HELENA; ARBEX, MÁRCIO, AARESTRUP ARBEX, (), “Descoberta de conhecimento em base de dados”. Disponível em: < http://fsd.edu.br/revistaeletronica/arquivos/2Edicao/artigo9.pdf >. Acessado em janeiro fevereiro de 2019.

[5] BASSEVILLE, MICHÈLE; NIKIFOROV, IGOR V., (1993), “Detection of abrupt changes-theory and application”. Disponível em: < https://www.researchgate.net/publication/2406812_Detection_of_Abrupt_Change_Theory_and_Application >. Acessado em fevereiro de 2019.

[6] CHO, HAERAN; FRYZLEWICZ, PIOTR, (2015), “Multiple-change-point detection for high dimensional time series via sparsified binary segmentation”. Disponível em: < https://research-information.bristol.ac.uk/files/86585676/paper.pdf >. Acessado em fevereiro de 2019.

[7] POWERS, D.M.W., (2011), “Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation”. Disponível em: < https://bioinfopublication.org/files/articles/2_1_1_JMLT.pdf >. Acessado em fevereiro de 2019.

[8] WIKIPEDIA - THE FREE ENCYCLOPEDIA. “Overfitting”. Disponível em: < https://en.wikipedia.org/wiki/Overfitting >. Acessado em fevereiro de 2019.

[9] KAWAHARA, YOSHINOBU; SUGIYAMA, MASASHI, (2012), “Sequential change-point detection based on direct density-ratio estimation”. Disponível em: < http://www.ms.k.u-tokyo.ac.jp/2012/CDKLIEP.pdf >. Acessado em fevereiro de 2019.

[10] ANTONAKAKIS, MANOS et al., (2017), “Understanding the mirai botnet”. Disponível em: < https://www.usenix.org/system/files/conference/usenixsecurity17/sec17-antonakakis.pdf >. Acessado em fevereiro de 2019.

[11] “Estacionariedade”. Disponível em: < http://www.portalaction.com.br/series-temporais/11-estacionariedade >. Acessado em fevereiro de 2019.

[12] IMDADULLAH., (2013), "Time Series Analysis". Basic Statistics and Data Analysis. Disponível em: < http://itfeature.com/time-series-analysis-and-forecasting/time-series-analysis-forecasting >. Acessado em março de 2019.

[13] FOK, LUIS., (2017), “What is Stationary series and non-Stationary series”. Disponível em: < https://www.quora.com/What-is-Stationary-series-and-non-Stationary-series >. Acessado em fevereiro de 2019.

[14] CLIFTON, CHRISTOPHER., (2010), "Encyclopædia Britannica: Definition of Data Mining". Disponível em: < https://www.britannica.com/technology/data-mining >. Acessado em março de 2019.

[15] ROMIJN, JAN-WILLEM, (2014). "Philosophy of statistics". Stanford Encyclopedia of Philosophy. Disponível em: < https://plato.stanford.edu/entries/statistics/ >. Acessado em março de 2019.

[16] WIKIPEDIA - THE FREE ENCYCLOPEDIA. “Inteligência Artificial”. Disponível em: < https://pt.wikipedia.org/wiki/Intelig%C3%AAncia_artificial >. Acessado em fevereiro de 2019.

[17] BISHOP, CHRISTOPHER M., (2006), “Pattern Recognition and Machine Learning”. Disponível em: < http://users.isr.ist.utl.pt/~wurmd/Livros/school/Bishop%20-%20Pattern%20Recognition%20And%20Machine%20Learning%20-%20Springer%20%202006.pdf >. Acessado em setembro de 2018.

[18] TAKEUCHI, J; YAMANISHI, K., "A unifying framework for detecting outliers and change points from time series," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 4, pp. 482-492, April 2006. doi: 10.1109/TKDE.2006.1599387. Disponível em: < http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1599387&isnumber=33630 >. Acessado em março de 2019.

[19] WIKIPEDIA - THE FREE ENCYCLOPEDIA. “Kernel Based Methods”. Disponível em: < https://en.wikipedia.org/wiki/Kernel_method#cite_note-1 >. Acessado em março de 2019.