UNIVERSIDADE ESTADUAL DO OESTE DO PARANÁ
CAMPUS DE FOZ DO IGUAÇU
PROGRAMA DE PÓS-GRADUAÇÃO EM
ENGENHARIA ELÉTRICA E COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

MÉTODO HÍBRIDO DE DETECÇÃO DE INTRUSÃO APLICANDO INTELIGÊNCIA ARTIFICIAL

CRISTIANO ANTONIO DE SOUZA

FOZ DO IGUAÇU
2018

Cristiano Antonio de Souza

Método Híbrido de Detecção de Intrusão Aplicando Inteligência Artificial

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Computação como parte dos requisitos para obtenção do título de Mestre em Engenharia Elétrica e Computação. Área de concentração: Sistemas Dinâmicos e Energéticos.

Orientador: Prof. Dr. Renato Bobsin Machado

Foz do Iguaçu
2018

Método Híbrido de Detecção de Intrusão Aplicando Inteligência Artificial

Cristiano Antonio de Souza

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Computação e aprovada pela Banca Examinadora:
Data da defesa pública: 09/02/2018.

Prof. Dr. **Prof. Dr. Renato Bobsin Machado** - (Orientador)
Universidade Estadual do Oeste do Paraná - UNIOESTE

Prof. Dr. **João Bosco Mangueira Sobral**
Universidade Federal de Santa Catarina - UFSC

Prof. Dr. **Romeu Reginatto**
Universidade Estadual do Oeste do Paraná - UNIOESTE

Resumo

As últimas décadas têm sido marcadas pelo rápido desenvolvimento tecnológico, o qual foi acelerado pela criação das redes de computadores, e enfaticamente pela disseminação e crescimento da Internet. Como consequência deste contexto, dados privados e sigilosos das mais diversas áreas passaram a ser tratados e armazenados em ambientes distribuídos, tornando-se vital a segurança dos mesmos. Decorrente ao fato, observa-se um crescimento na quantidade e variedade de ataques a sistemas computacionais, principalmente pela exploração de vulnerabilidades. Em função desse contexto, a área de pesquisa em detecção de intrusão tem ganhado notoriedade, e os métodos híbridos de detecção utilizando técnicas de Inteligência Artificial vêm alcançando resultados mais satisfatórios do que a utilização de tais abordagens de modo individual. Este trabalho consiste em um método Híbrido de detecção de intrusão combinando as técnicas Redes Neurais Artificiais (RNA) e *K-Nearest Neighbors* (KNN). A avaliação do método Híbrido proposto e a comparação com as técnicas de RNA e KNN isoladamente foram desenvolvidas de acordo com as etapas do processo de *Knowledge Discovery in Databases* (KDD) . Para a realização dos experimentos selecionou-se a base de dados pública NSL-KDD, sendo que com o processo de seleção de atributos derivou-se cinco sub-bases. Os resultados experimentais comprovaram que o método Híbrido teve melhor acurácia em relação a RNA em todas as configurações, ao passo que em relação ao KNN, alcançou acurácia equivalente e apresentou relevante redução no tempo de processamento. Por fim, cabe ressaltar que dentre as configurações híbridas avaliadas quantitativa e estatisticamente, os melhores desempenhos em termos de acurácia e tempo de classificação foram obtidos pelas abordagens híbridas HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20.

Palavras-chave: Redes Neurais Artificiais, K-Nearest Neighbor, Segurança Computacional .

Abstract

The last decades have been marked by rapid technological development, which was accelerated by the creation of computer networks, and emphatically by the spread and growth of the Internet. As a consequence of this context, private and confidential data of the most diverse areas began to be treated and stored in distributed environments, making vital the security of this data. Due to this fact, the number and variety of attacks on computer systems increased, mainly due to the exploitation of vulnerabilities. Thence, the area of intrusion detection research has gained notoriety, and hybrid detection methods using Artificial Intelligence techniques have been achieving more satisfactory results than the use of such approaches individually. This work consists of a Hybrid method of intrusion detection combining Artificial Neural Network (ANN) and *K-Nearest Neighbors* KNN techniques. The evaluation of the proposed Hybrid method and the comparison with ANN and KNN techniques individually were developed according to the steps of the *Knowledge Discovery in Databases* process. For the realization of the experiments, the NSL-KDD public database was selected and, with the attribute selection task, five sub-bases were derived. The experimental results showed that the Hybrid method had better accuracy in relation to ANN in all configurations, whereas in relation to KNN, it reached equivalent accuracy and showed a significant reduction in processing time. Finally, it should be emphasized that among the hybrid configurations evaluated quantitatively and statistically, the best performances in terms of accuracy and classification time were obtained by the hybrid approaches HIB(P25-N75)-C, HIB(P25-N75)-30 and HIB(P25-N75)-20.

Keywords: Artificial Neural Networks, K-Nearest Neighbor, Network Security.

Agradecimentos

Agradeço primeiramente a minha família, meus pais Valderi Francisco de Souza e Maria Tereza Rama de Souza, e a meu irmão Cristian Robson de Souza, pelo amor e carinho demonstrado, e pela dedicação em fornecer todas as condições para alcançar meus objetivos, por isso e por muito mais, sou eternamente grato.

Agradeço ao professor, orientador e amigo Renato Bobsin Machado pelas suas orientações e seu apoio, durante toda a elaboração deste trabalho.

Agradeço ao grande amigo e colega de estudos Gustavo dos Santos Vieira, por todos estes anos de convivência e companheirismo na universidade e no laboratório. Além disso agradeço ao Gustavo e aos amigos Gustavo Johann e Jean Douglas Valêncio por toda companhia e parceria fora da universidade, todos os jogos de futebol assistidos, todas as partidas de futebol jogadas e a todos os chopps brindados.

Agradeço à grande amiga e colega de mestrado Mailla Spricigo, por todo companheirismo e ajuda durante o desenvolvimento deste trabalho.

Agradeço também aos amigos Thiago Toledo e Matheus Bainy pela amizade e companheirismo cultivado durante o mestrado e as rodadas de chopp.

Agradeço à amiga Pamela Ferreira, pela amizade e incentivo durante todo o período de realização deste trabalho.

Agradeço aos demais pesquisadores e amigos do Laboratório de Pesquisa em Segurança Computacional (LaPSeC) da Unioeste, Alisson Flecha, Lanna Schuster, Natan Paredes, Willian Antonio de Rosa, por toda a convivência e troca de conhecimento, que proporcionou um ambiente extremamente amigável e favorável para o desenvolvimento deste trabalho.

Também agradeço a Universidade Federal do Paraná (UFPR), que através do Centro de Computação Científica e Software Livre (C3SL) permitiu o uso do computador de alto desempenho (HPC), onde foram realizados todos os experimentos.

Agradeço à CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) pela concessão de bolsa durante o período de realização deste trabalho.

Agradeço ainda ao Programa de Pós-Graduação em Engenharia Elétrica e Computação, da Universidade Estadual do Oeste do Paraná, campus Foz do Iguaçu, pelo apoio prestado ao LaPSeC, fato que auxiliou em grande proporção na realização deste trabalho.

“A melhor maneira de prever o futuro é criá-lo.”
(Peter Drucker)

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

AES	Advanced Encryption Standard
AIS	Artificial Immune Systems
C3LS	Centro de Computação Científica e Software Livre
CANN	Abordagem combinando Cluster Centers and Nearest Neighbor
CBR	Case Based Reasoning
CERT.br	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CIDF	Common Intrusion Detection Framework
CISL	Common Intrusion Specification Language
CF	Collaborative filtering
DBSCAN	Density-basedspatial clustering of applications with noise
DDoS	Distributed Denial of Service
DES	Data Encryption Standart
DMC-NS	Distributed Mobile Computing and Network Security
DNN	Deep Neural Network
DoS	Denial of Service Attacks
FC-ANN	Abordagem combinando RNA e Clusterização Fuzzy
FPGA	Field Programmable Gate Array
GR	Gain Ratio
HG-GA	Abordagem de Algoritmos Genéticos baseada em Hipergrafos
HIB(P25-N1)-C	Abordagem de Híbrida (P25-N1) aplicada na base NSL-KDD completa
HIB(P25-N75)-C	Abordagem de Híbrida (P25-N75) aplicada na base NSL-KDD completa
HIB(P25-N1)-30	Abordagem de Híbrida (P25-N1) aplicada na base NSL-KDD com 30 atributos
HIB(P25-N75)-30	Abordagem de Híbrida (P25-N75) aplicada na base NSL-KDD com 30 atributos
HIB(P25-N1)-20	Abordagem de Híbrida (P25-N1) aplicada na base NSL-KDD com 20 atributos
HIB(P25-N75)-20	Abordagem de Híbrida (P25-N75) aplicada na base NSL-KDD com 20 atributos

HIB(P1-N1)-12	Abordagem de Híbrida (P25-N1) aplicada na base NSL-KDD com 12 atributos
HIB(P1-N75)-12	Abordagem de Híbrida (P1-N75) aplicada na base NSL-KDD com 12 atributos
HIB(P25-N1)-6	Abordagem de Híbrida (P25-N1) aplicada na base NSL-KDD com 6 atributos
HIB(P1-N75)-6	Abordagem de Híbrida (P1-N75) aplicada na base NSL-KDD com 6 atributos
HIDS	Host Based Intrusion Detection System
IA	Inteligência Artificial
IDS	Intrusion Detection System
IDWG	Intrusion Detection Working Group
IG	Information Gain
KDD	Knowledge Discovery in Databases
KNN	K-Nearest Neighbor
KNN-C	Abordagem de RNA aplicada na base NSL-KDD completa
KNN-30	Abordagem de RNA aplicada na base NSL-KDD com 30 atributos
KNN-20	Abordagem de RNA aplicada na base NSL-KDD com 20 atributos
KNN-12	Abordagem de RNA aplicada na base NSL-KDD com 12 atributos
KNN-6	Abordagem de RNA aplicada na base NSL-KDD com 6 atributos
LaPSeC	Laboratório de Pesquisa em Segurança Computacional
MD5	Message-Digest algorithm 5
MLP	MultiLayer Perceptron
NIDS	Network Based Intrusion Detection System
OS-ELM	Online Sequential Extreme Learning Machine
P1-N1	Abordagem para definir limites da faixa intermediária: 1º valor positivo a 1º valor negativo
P1-N25	Abordagem para definir limites da faixa intermediária: 1º valor positivo a 25º percentil negativo
P1-N50	Abordagem para definir limites da faixa intermediária: 1º valor positivo a 50º percentil negativo
P1-N75	Abordagem para definir limites da faixa intermediária: 1º valor positivo a 75º percentil negativo
P25-N1	Abordagem para definir limites da faixa intermediária: 25º percentil positivo a 1º valor negativo
P25-N25	Abordagem para definir limites da faixa intermediária: 25º percentil positivo a 25º percentil negativo
P25-N50	Abordagem para definir limites da faixa intermediária: 25º percentil positivo a 50º percentil negativo

P25-N75	Abordagem para definir limites da faixa intermediária: 25º percentil positivo a 75º percentil negativo
PBIL	Population-Based Incremental Learning
PNN	Probabilistic Neural Network
R2L	Remote to Local
RBF	Radial Basis Function
RNA	Redes Neurais Artificiais
RNA-C	Abordagem de RNA aplicada na base NSL-KDD completa
RNA-30	Abordagem de RNA aplicada na base NSL-KDD com 30 atributos
RNA-20	Abordagem de RNA aplicada na base NSL-KDD com 20 atributos
RNA-12	Abordagem de RNA aplicada na base NSL-KDD com 12 atributos
RNA-6	Abordagem de RNA aplicada na base NSL-KDD com 6 atributos
SHA-1	Secure Hash Algorithm 1
SC	Clustering Spectral
SCDNN	Abordagem combinando Agrupamento Espectral e Redes Neurais Profundas
SVM	Support Vector Machine
TANN	Abordagem de KNN baseado em áreas de triângulos
U2R	User to Root
UNIOESTE	Universidade Estadual do Oeste do Paraná

Capítulo 1

Introdução

As últimas décadas tem sido marcadas por um alto grau de desenvolvimento tecnológico, merecendo destaque a evolução das redes de computadores, suas aplicações, assim como a pesquisa e a criação de métodos direcionados a segurança destas redes e das informações envolvidas. É importante frisar que, neste contexto, a expansão da Internet trouxe muitas facilidades e permitiu o desenvolvimento de aplicações computacionais em diversas áreas de conhecimento, sejam de instituições públicas ou privadas (Tsai et al., 2009).

Como consequência deste cenário, informações das mais variadas naturezas são processadas a todo momento por sistemas que, geralmente, estão interligados através da Internet e, atualmente, em soluções distribuídas, tais como servidores remotos, computação em nuvem, entre outros. Grande parte desse grande volume de dados são confidenciais e privados, entre os quais pode-se enumerar documentos financeiros, bancários, senhas, informações pessoais, diagnósticos médicos. É notório, que uma quantidade cada vez maior de atividades essenciais são realizadas por meio da Internet e recursos tecnológicos, e seu funcionamento correto e confiável é de fundamental importância (Shon & Moon, 2007).

Em paralelo ao crescimento tecnológico, surgem também dificuldades para se manter a segurança das aplicações e infraestruturas computacionais, tendo em vista que conjuntamente com o aumento da quantidade de serviços disponíveis, cresce também a possibilidade de existirem vulnerabilidades. Isto faz com que técnicas especiais de segurança se tornem indispensáveis nos sistemas computacionais modernos (Lima, 2005)

Apesar de a ARPANET, predecessora da Internet, ter sido criada apenas nos anos no final dos anos 60, a preocupação com a segurança de dados existe a milênios. No período antes de Cristo o Imperador da Roma antiga, Júlio César, utilizava cifra de substituição baseada no alfabeto. Tal método consistia no uso de uma chave criptográfica por meio de deslocamento, sendo o alfabeto tratado de modo cíclico, e por conseguinte, a letra do texto natural era substituída pela terceira letra do alfabeto após a mesma (Goodrich & Tamassia, 2013). Muitas outras técnicas foram desenvolvidas ao longo do tempo, no entanto esse tema tornou-se primordial a partir do desenvolvimento dos computadores pessoais e dos ambientes em rede e distribuídos.

A partir dos anos 70 houve uma grande evolução no desenvolvimento de métodos de segurança computacional, merecendo destaque a implementação de alguns métodos criptográficos, entre os quais: *Data Encryption Standart* (DES) (Feistel, 1973), RSA (Rivest et al., 1978), *firewalls* (Tanenbaum et al., 2003), *Message-Digest algorithm 5* (MD5) (Rivest, 1992), *Secure Hash Algorithm 1* (SHA-1) (Preneel, 1999), *Advanced Encryption Standard* (AES) (Encryption, 2016). No entanto, o desenvolvimento de novas aplicações computacionais também permitiram a exploração de vulnerabilidades por meio de usuários maliciosos que usam tais brechas para práticas ilícitas, com o objetivo de obter conteúdo digital privilegiado, seja para benefício próprio ou mesmo para demonstrar suas habilidades e conhecimento técnico. Neste cenário, muitas técnicas intrusivas são disseminadas pela própria Internet, aumentando potencialmente tais ocorrências (Goodrich & Tamassia, 2013).

Neste contexto, os ataques às redes e serviços, visando a captura de informações privilegiadas, são comuns e a diversidade dos ataques gera a necessidade de investimento financeiro e intelectual na segurança de redes de computadores. Existem inúmeras ferramentas e técnicas desenvolvidas para prover segurança computacional. No entanto, o número, a variedade e a complexidade dos incidentes referentes a segurança têm crescido expressivamente, como pode ser observado na Figura 1.1, que apresenta os incidentes de segurança reportados ao Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.br).

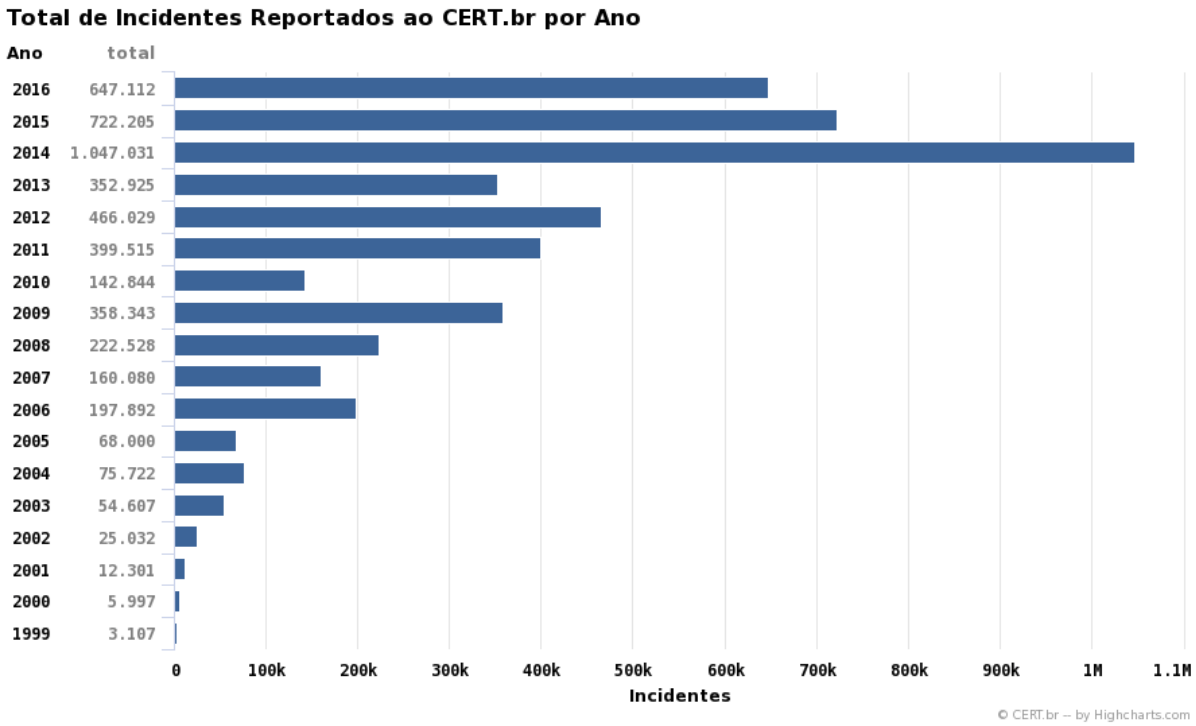


Figura 1.1: Incidentes de segurança reportados ao CERT.br por ano. Fonte: www.cert.br

Este cenário tem motivado a pesquisa e o desenvolvimento de trabalhos aplicando técnicas de inteligência artificial na segurança computacional, com o objetivo de melhorar os métodos convencionais de detecção e contornar as suas limitações. Na sequência apresentam-se alguns

trabalhos que contribuíram para o delineamento do presente trabalho.

Em 2010, Wang et al. (2010) propuseram uma abordagem chamada FC-ANN, que aplica Redes Neurais Artificiais (RNA) e clusterização *fuzzy* para detecção de intrusões. Primeiramente a técnica de agrupamento *fuzzy* foi utilizada para gerar vários sub-conjuntos de treinamento, e na sequência diferentes modelos de RNA foram treinados com base nestes dados para gerar diferentes modelos base. Posteriormente o módulo de agregação *fuzzy* foi empregado parar agrupar os resultados. Os experimentos foram realizados sobre a base **KDD CUP 99 Data** e demonstraram que a abordagem FC-ANN supera os métodos bem conhecidos como árvores de decisão e *Naive Bayes* quanto a precisão de detecção.

O trabalho de Govindarajan & Chandrasekaran (2011) apresenta dois métodos de classificação tradicionais, o primeiro uma rede neural *Multi-Layer Perceptron* (MLP), e o segundo uma rede neural *Radial Basis Function* (RBF) (Russell & Norvig, 2010). A proposta caracteriza um método de classificação híbrido, utilizando redes neurais MLP e RBF. A saída de cada classificador é dada com um peso, na escala (0-1), dependendo do desempenho de generalização. Se ambos os classificadores produzirem a mesma saída, ela é aceita, caso exista conflito, o classificador com maior peso determina a saída. Os resultados do trabalho mostraram que o método híbrido proposto teve uma uma precisão significativamente maior que a utilização única dos métodos.

Em 2015, Lin et al. (2015) publicaram um trabalho que consistia em uma abordagem híbrida para detecção de intrusão utilizando a técnica *k-Nearest Neighbors* (KNN) juntamente com a clusterização. Tal método utiliza um valor baseado na soma de duas distâncias para representar cada amostra em um classificador KNN, onde a primeira consiste na distância entre cada amostra e o centro de seu *cluster*, e a segunda é a distância entre cada amostra e o seu vizinho mais próximo no mesmo *cluster*. Utilizando a base de dados **KDD CUP 99 Data** como conjunto de dados, os autores obtiveram resultados melhores ou semelhantes a abordagem KNN individual em relação a precisão de detecção, e além disso, com um menor tempo de processamento.

Também é digno de nota que diversos trabalhos vêm sendo realizados em parceria entre o Laboratório de Segurança Computacional (LaPSeC) da Universidade Estadual do Oeste do Paraná (UNIOESTE) e o laboratório DMC-NS (*Distributed Mobile Computing and Network Security*), situado no Departamento de Informática e Estatística da Universidade Federal de Santa Catarina (UFSC). A colaboração entre as instituições citadas acontece desde 2005 e desde então vem trazendo avanços na área de atuação, como uma proposta de abordagem utilizando sistemas imunológicos artificiais para a identificação de ataques à redes, desenvolvida em um programa de mestrado unindo esforços das duas instituições (Machado, 2005b).

Por meio dessa breve contextualização sobre a evolução das técnicas de segurança computacional e a importância de ser ter um bom mecanismo de segurança, é possível observar que novas abordagens de detecção de intrusão devem ser desenvolvidas em busca de melhorias na

segurança de redes.

Motivado por dar continuidade a esta linha de pesquisa, o trabalho ora proposto situa-se na área de detecção de intrusões com o emprego de métodos de inteligência artificial. Assim, propõe-se neste trabalho, um método de detecção de intrusão híbrido, utilizando Redes Neurais Artificiais e também o algoritmo *k-Nearest Neighbors*, visando alcançar melhores índices de detecção de ataques em relação as estas mesmas técnicas abordadas de maneira individual.

1.1 Proposta desta Dissertação

Os trabalhos analisados, sobre o estado da arte, apresentam várias técnicas e abordagens que foram pesquisadas e propostas. Os métodos híbridos de detecção propostos atualmente, criados pela combinação ou integração de múltiplas técnicas de classificação, estão obtendo resultados mais satisfatórios que a utilização das técnicas de maneira individual.

Com o objetivo de propor um método híbrido, observou-se que o método KNN possui uma alta taxa de detecção tanto para eventos intrusivos quanto para eventos não intrusivos e que, no entanto, possui uma alta taxa de processamento e de tempo de execução durante o processo de classificação. Isso deve-se ao fato de que o KNN realiza o cálculo de distância comparando a instância a ser classificada com todos os exemplos presentes na base, e ainda usando todos os seus atributos. Por outro lado, um modelo neural (RNA) já treinada, possui um tempo de execução muito pequeno para classificação, pois é um modelo matemático em que cada amostra é aplicada, no entanto sua taxa de detecção é inferior ao método KNN.

Desse modo, o método híbrido proposto consiste em uma abordagem composta por uma RNA e pelo algoritmo KNN. O objetivo é melhorar a taxa de detecção da técnica RNA, de modo que alcance essa melhoria com menos processamento para a classificação dos exemplos do que a técnica KNN.

Foram selecionadas para compor o método híbrido proposto uma RNA *Feedfoward* do tipo *Multilayer Perceptrons*, devido a capacidade de resolver problemas não linearmente separáveis.

A segunda técnica utilizada no método proposto é o algoritmo KNN. O objetivo do algoritmo KNN é determinar a classe de uma amostra baseado nos exemplos, considerando os K vizinhos mais próximos. Inicialmente definimos um valor de K igual a 1 para a abordagem proposta.

O protocolo proposto inicia com o estabelecimento dos modelos neural e KNN. A instância a ser classificada é primeiramente apresentada a RNA, gerando uma saída entre -1 e 1 para cada amostra, onde os exemplos mais próximos de -1 se assemelham ao comportamento não intrusivo, e os exemplos mais próximos de 1 se assemelham ao comportamento intrusivo.

Neste contexto, os exemplos que obtiveram valores intermediários da saída do modelo neural, passam por um segundo método de classificação, que é o KNN. A definição da faixa intermediária, assim como a geração dos modelos e sua avaliação foi realizada aplicando a base de dados NLS-KDD (Tavallaee et al., 2009).

1.2 Estrutura do Trabalho

No Capítulo 2 é apresentado o problema da segurança em redes de computadores. Também são abordados conceitos relacionados a detecção de intrusão, que envolvem a detecção de intrusão, como padronização, métodos de detecção e variações arquiteturais.

O Capítulo 3 é destinado a apresentar conceitos relativos à Inteligência Artificial (IA), com ênfase em métodos tais como aprendizado de máquina, RNA algoritmo KNN.

No Capítulo 4 apresentam-se alguns trabalhos científicos, considerando-se o estado da arte em detecção de intrusão. Nas abordagens selecionadas destacam-se os métodos e recursos computacionais aplicados.

No Capítulo 5 é definido o método proposto, a base utilizada para experimentação, além de toda a metodologia definida para a realização dos experimentos.

No Capítulo 6 são apresentados as taxas de acurácia, taxas de erro, além dos valores de verdadeiros positivos, verdadeiros negativos, falsos positivos e falsos negativos obtido por cada um dos métodos. Neste capitulo também são discutidos os resultados obtidos, considerando-se todas as experimentações realizadas, assim como uma análise estatística dos modelos avaliados..

No Capítulo 7 são apresentadas as conclusões, contribuições e propostas para trabalhos futuros.

Para a elaboração do texto aplicou-se algumas padronizações, termos em inglês são apresentados em *itálico* e ferramentas e técnicas computacionais são destacadas com fonte *emphasise*.

Capítulo 2

Segurança de Redes e Detecção de Intrusão

2.1 Considerações Iniciais

Este capítulo é dedicado a abordar conceitos de segurança de redes de computadores e métodos utilizados para identificar ataques e violações de segurança. Para isso, inicialmente serão descritas as principais ameaças à segurança computacional, classes e características das técnicas intrusivas existentes. Com foco na detecção e solução de tais problemas também serão abordados conceitos, funcionalidades, classificação e padronização referentes aos Sistemas de Detecção de Intrusão (*Intrusion Detection System* - IDS).

2.2 Segurança Computacional

Com o desenvolvimento das redes computacionais, e enfaticamente com a popularização da Internet, observou-se um crescimento expressivo de aplicações computacionais, de modo que o valor dos dados e a sua privacidade passaram a ser de grande importância. Diante deste novo contexto tecnológico, surgiram dificuldades para se manter a segurança de aplicações e dos dados, tendo em vista que as técnicas para explorar vulnerabilidades desta infraestruturas computacionais são constantemente aperfeiçoadas, com o objetivo de adquirir acesso a sistemas, obter e fazer uso indevido de informações sensíveis.

As vulnerabilidades de sistemas computacionais são exploradas por usuários maliciosos para realizar atividades ilícitas, motivado por distintos motivos, entre os quais obter conteúdo digital privilegiado que possa trazer algum beneficio ao invasor e/ou causar algum dano significativo ao alvo dos ataques.

Em contrapartida, a área de segurança computacional preocupa-se em proteger sistemas e recursos contra invasões e acessos indevidos. Os princípios elementares da segurança com-

putacional estão centrados em três (3) características, expressas por meio do acrônimo C.I.D. (do inglês C.I.A., *confidentiality, integrity, availability*), que significa confidencialidade, integridade e disponibilidade (Goodrich & Tamassia, 2013). A seguir conceitua-se tais princípios (Goodrich & Tamassia, 2013):

- **Confidencialidade:** Consiste na proteção dos dados, de modo que o acesso aos mesmos somente ocorre mediante autorização. Baseia-se na premissa que somente pessoas autorizadas devem ter acesso a esses dados;
- **Integridade:** Tal princípio procura garantir que as informações não devem sofrer alterações indevidas, sejam elas feitas por softwares maliciosos ou por pessoas não autorizadas. Por exemplo, durante uma transmissão de informação, a mensagem enviada deve trafegar do emissor até o receptor sem sofrer alterações;
- **Disponibilidade:** Propriedade de que a informação deve estar sempre acessível e modificável àqueles que possuem autorização para o fazê-lo. Uma informação que não estiver disponível quando requisitada perde sua utilidade.

2.3 Tipos de Ameaças

Um ataque consiste em uma ação maliciosa, realizada com o intuito de comprometer e violar os princípios de segurança. A principal estratégia utilizada para alcançar tal objetivo é a exploração de vulnerabilidades dos sistemas. Os resultados de um ataque podem caracterizar uma invasão, acarretar indisponibilidade de serviços ou mesmo acesso indevido a informações confidenciais (Bace & Mell, 2001).

De acordo com Bace & Mell (2001), os tipos de ataques mais comumente reportados por IDS são: ataques de sondagem (*Scanning Attacks*), negação de serviço (*Denial of Service Attacks - DoS*) e ataques de penetração (*Penetration Attacks*).

2.3.1 Ataques de Sondagem

Os ataques de sondagem consistem no envio de pacotes de dados a fim de examinar uma rede ou *host* alvo. Usando as respostas recebidas do alvo, o invasor pode aprender características e vulnerabilidades do sistema. Esse tipo de ataque possibilita a extração de informações importantes, tais como: tipos de tráfego utilizados, endereços de servidores, serviços ativos, sistemas operacionais, versões dos serviços, entre outras. Ao obter essas informações o atacante pode utilizar ataques e ferramentas que explorem estas vulnerabilidades (Bace & Mell, 2001).

2.3.2 Negação de Serviço

Os ataques de negação de serviço são projetados para interromper temporariamente ou impedir completamente o acesso a serviços, recursos, *hosts* e redes. Neste tipo de ataque um intruso é capaz de monopolizar um recurso ao utilizá-lo de modo anormal, fazendo com que as solicitações feitas pelos demais usuários sejam negadas (Denning, 1987).

Existem dois tipos principais de ataques desta natureza:

- **Exploração de falhas:** Técnica que explora falhas no alvo para provocar o esgotamento de recursos;
- **Inundação:** Métodos que enviam pacotes a taxas que o alvo não tenha capacidade para processar.

2.3.3 Ataques de Penetração

Os ataques de penetração consistem na aquisição ou alteração não autorizada de privilégios. Esse tipo de ataque é mais perigoso se comparados a sondagem e de DoS, pois podem comprometer a disponibilidade, integridade e controle de sistemas. Ataques de penetração têm por objetivo assumir o controle de um sistema por meio da exploração de falhas e vulnerabilidades de *software*.

Os ataques de penetração variam quanto a seus detalhes e impactos. Os tipos mais comuns são:

- *User to Root*: Um usuário local em um *host* obtém controle completo do *host* alvo;
- *Remote to User*: Um invasor utiliza a rede e consegue uma conta de usuário no *host* de destino;
- *Remote to Root*: Um invasor, por meio da rede, obtém o controle completo do *host* alvo;
- *Remote Disk Read*: Um invasor na rede adquire a capacidade de ler arquivos de dados privados no *host* de destino sem a autorização do proprietário;
- *Remote Disk Write*: Um *cracker*, usuário da rede, angaria a capacidade de escrever em arquivos de dados privados no *host* de destino sem a autorização do proprietário.

2.4 Detecção de Intrusão

Uma intrusão, de acordo com Crosbie & Spafford (1995), pode ser definida como um conjunto de ações realizadas com o intuito de ultrapassar as barreiras de defesa de uma aplicação para comprometer a integridade, confidencialidade e disponibilidade de recursos.

A área de pesquisa de Detecção de Intrusão tem como objetivo propor e aplicar técnicas computacionais para identificar intrusões e implementar mecanismos de contramedidas (Machado, 2005b).

Já os sistemas de detecção de intrusão têm como objetivo reconhecer comportamentos intrusivos em uma rede e alertar os administradores ou realizar ações de contra medidas automaticamente (Bace & Mell, 2001).

Os IDS são inseridos como a última linha de defesa dentro de uma arquitetura computacional, o que os torna de importância vital, possibilitando inferir sobre a legitimidade de ações realizadas e possuindo comportamento pró-ativo em situações de ataque (Bernardes, 1999).

2.4.1 Estrutura e Padronização dos IDS

Atualmente existem vários métodos e ferramentas para a detecção de intrusão, as quais diferem no que concerte a importantes decisões de projeto, entre as quais: métodos de captura de eventos, mecanismos de classificação de eventos e técnicas para a realização de contramedidas.

Essa diversidade de abordagens dos métodos de detecção de intrusão, associados a implementação de ataques cada vez mais sofisticados, motivaram o estabelecimento de padronizações para a definição da arquitetura de um IDS. Deste modo contribui-se para a interação e integração entre ferramentas desse gênero.

Uma das padronizações mais aceitas é a *Common Intrusion Detection Framework* (CIDF) (Staniford-Chen et al., 1998) criada pelo *CIDF Work Group*. Entre os objetivos deste grupo está o desenvolvimento de protocolos e interfaces de programação, de modo que os variados projetos e pesquisas relacionadas a IDS pudessem partilhar informações e recursos (Staniford-Chen et al., 1998).

A padronização CIDF define uma arquitetura integrada por caixas (módulos) que interagem entre si, constituindo um modelo para IDS aplicando a *Common Intrusion Specification Language* (CISL) para a comunicação entre os componentes. Como pode ser observado na Figura 2.1, a padronização CIDF possui os seguintes módulos: E-Box (gerador de eventos), A-Box (analisador de eventos), D-Box (base de dados de eventos) e C-Box (contra medidas). Os componentes E-box e A-box são as mais utilizadas e estão presentes em todos os IDS.

A seguir são descritas as funcionalidades de cada um dos módulos de acordo com Staniford-Chen et al. (1998):

- A Caixa *E-box* trata dos métodos aplicáveis para capturar eventos gerados fora do IDS e padronizá-los. Esse tipo de componente não realiza processamento sobre os dados, apenas os captura e os envia para os componentes de análise e armazenamento. Os componentes do tipo *A-box* definem os mecanismos para análise dos eventos capturados pelos

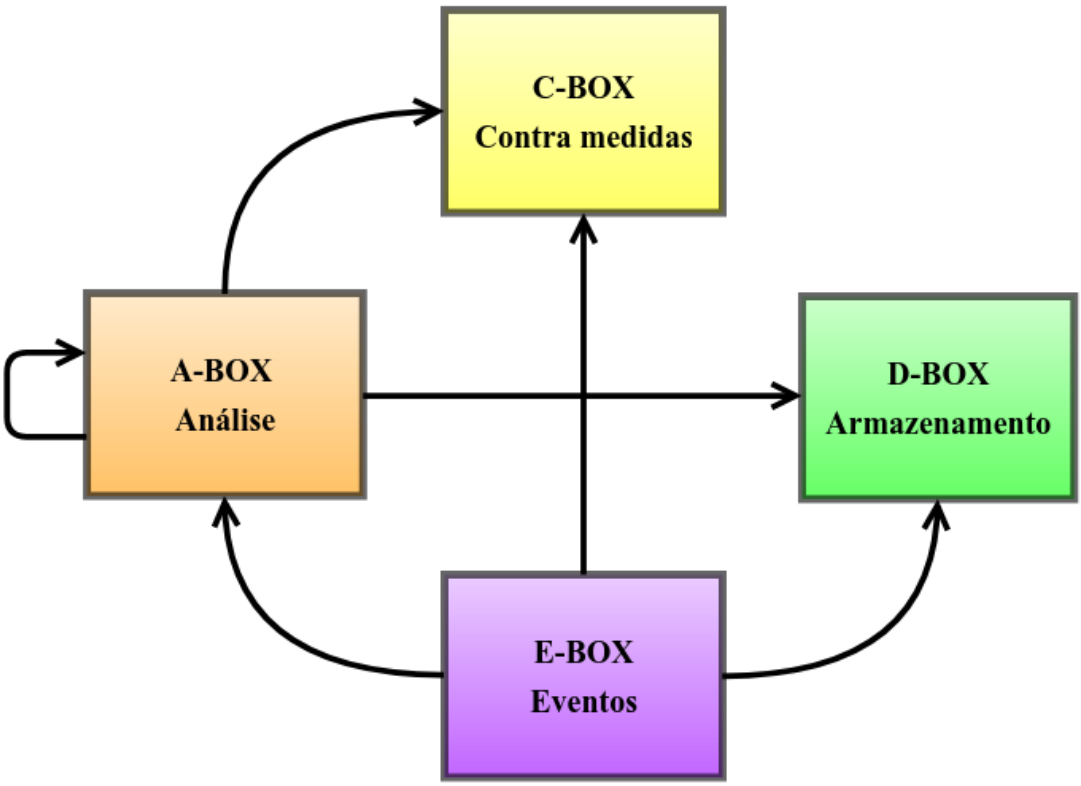


Figura 2.1: Relação de componentes no padrão CIDF. Adaptado de (Barbosa & Moraes, 2000).

componentes do tipo *E-box*.

- A caixa *A-box* recebe os eventos capturados pela *E-box*, busca por padrões que caracterizem um ataque e os envia para outros componentes, como a *D-box*, para armazenamento, e a *C-box*, para a execução de contramedidas. É o componente com o papel mais importante em um IDS, pois é nele que são realizadas as classificações dos eventos, identificando eventos intrusivos e eventos não intrusivos.
- As Bases de Dados de Eventos (*D-box*) foram projetadas para armazenar os eventos provenientes da *E-box* para futura análise, ou provenientes da *A-box*, neste caso, tratando-se de eventos já analisados. Esses registros formam a base de conhecimento do IDS.
- A última caixa é a *C-box*, conhecida como Unidade de Contramedidas. Este componente é formado por unidades de resposta que possuem o objetivo de realizar atividades para responder as intrusões detectadas. Tais contramedidas são executadas pela *C-box*, e são classificadas como passiva, através da emissão de relatórios para que haja interação humana, ou ativa, através de intervenções automáticas, como encerramento de processos e desligamento de servidores.

O *Intrusion Detection Working Group* (IDWG) é um grupo criado com o objetivo de definir formatos de dados e procedimentos para compartilhamento de informações entre IDS. A arquitetura desenvolvida pelo IDWG é mais detalhada em relação ao CIDF, tendo como componentes principais os sensores, o analisador e o gerente (Wood & Erlinger, 2007).

Os sensores são responsáveis por reunir os eventos suspeitos e repassá-los ao analisador. Os analisadores classificam os eventos de acordo com a política de segurança definida. Caso o evento seja classificado como intrusivo é gerado alerta para o gestor, o qual notifica o operador, que então tomará providências, sejam elas automáticas ou não. O gestor é manipulado pelo administrador da rede e possui funções como configurar os demais componentes, avaliar e mensurar os eventos notificados (Wood & Erlinger, 2007).

Existem outras abordagens que utilizam organizações diferentes, no entanto, em todas estão presentes os capturadores de eventos e algum mecanismo de análise.

2.4.2 Classificação de IDS

Os IDS são compostos por uma estrutura de *hardware* e *software* que tem por objetivo a detecção de eventos intrusivos em redes de computadores. Essa estrutura pode variar em relação à maneira que é implantada, pela frequência de operação, pelos dados que analisam ou pelas análises feitas sobre esses dados (Campello & Weber, 2001).

A Figura 2.2 apresenta uma representação geral das diferentes classificações de IDS, cujos métodos aplicáveis são descritos nas subseções seguintes.

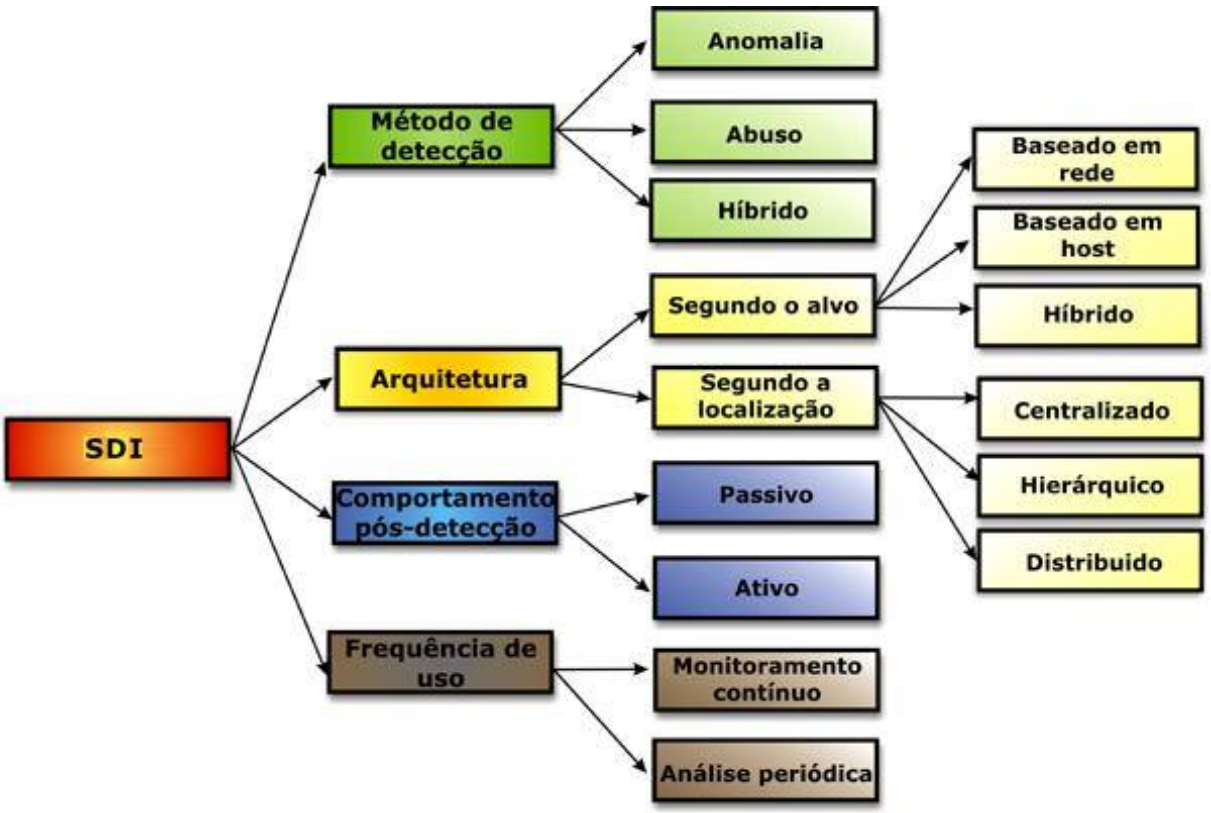


Figura 2.2: Classificação dos Sistemas de Detecção de Intrusão. Fonte: (Campello & Weber, 2001).

Método de Detecção

Os IDS podem ser classificados de acordo com os métodos de detecção empregados. As duas abordagens primárias amplamente utilizadas para analisar eventos e detectar ataques são: detecção por anomalia e detecção por abuso (Northcutt et al., 2001).

Na detecção por abuso, as ações monitoradas são comparadas com eventos intrusivos predefinidos, normalmente armazenadas em uma base de dados sequencias de eventos que caracterizam uma intrusão. Esses padrões previamente conhecidos são chamados de assinaturas e apesar de permitir uma rápida detecção e diminuir a ocorrência de alarmes falsos, possui a limitação de apenas detectar esses ataques conhecidos (Northcutt et al., 2001). Este método é utilizado pela maioria dos sistemas de antivírus comerciais (Bace & Mell, 2001).

A detecção por anomalia assume que toda atividade intrusa é necessariamente anômala, ou seja, toda atividade que não se enquadra nos modelos de comportamentos definidos como normais é considerada uma intrusão. A grande vantagem da técnica de detecção por anomalia é o fato de permitir detectar novos ataques e/ou variações de ataques conhecidos, uma vez que não é necessário conhecê-los anteriormente. No entanto, essa técnica tem maior tendência de classificar eventos erroneamente, pois nem toda atividade anômala é uma intrusão (Machado, 2005b).

Ambos os métodos supracitados possuem vantagens e desvantagens, cada um se torna mais adequado para detectar certo tipo de ataque. Devido a isso, muitos trabalhos estão propondo abordagens híbridas combinando detecção por anomalia e por abuso.

Arquitetura Segundo o Alvo

Uma outra classificação dos IDS é definida em função do alvo. Alguns IDS analisam os pacotes de rede para encontrar atacantes, enquanto outros analisam fontes de informação geradas pelo sistema operacional ou por demais *softwares* (Bace & Mell, 2001).

Em um IDS baseado em *host*, também conhecido como *Host Based Intrusion Detection System* (HIDS), todos os componentes, desde a coleta dos eventos até a classificação, estão localizados no mesmo *host* e os mecanismos de monitoramento de eventos e de análise dos eventos utilizam apenas informações do próprio computador. Os eventos podem ser originados a partir de *logs* do sistema, dados sobre usuários, serviços e processos. Essa abordagem possibilita a independência da rede e a detecção de ataques internos, mas possui como desvantagem a dificuldade em tratar ataques da rede e ataques ao próprio IDS (Bace & Mell, 2001).

Os IDS baseados em rede são conhecidos como *Network Based Intrusion Detection System* (NIDS). Nessa abordagem os eventos e atividades são obtidos capturando o tráfego de rede em modo promíscuo. Os IDS baseados em rede geralmente consistem em um conjunto de sensores colocados em vários pontos estratégicos da rede. Uma das dificuldades existentes neste

tipo de abordagem é determinar os melhores locais para posicionar os sensores. No entanto, esta estratégia possibilita a detecção de ataques externos e é mais independente de plataforma, quando comparado aos demais tipos de IDS (Mukherjee et al., 1994).

Arquitetura Segundo o Local

Os IDS também podem ser classificados de acordo com o modo pelo qual os seus componentes estão dispostos. Desse modo, os IDS podem ser classificados em centralizados, parcialmente distribuídos e distribuídos (Campello & Weber, 2001).

Os IDS centralizados possuem todos seus componentes funcionais em apenas um único ponto, desde a captura dos eventos até a configuração e gerência. Como vantagens, esse tipo de IDS apresenta facilidade no desenvolvimento, instalação e configuração. Como desvantagem, uma solução totalmente centralizada pode ser inviável em sistemas complexos (Porras & Neumann, 1997).

Na abordagem parcialmente distribuída, os componentes são dispostos de modo distribuído e são coordenados por módulos gerentes centralizados, obedecendo uma estrutura hierárquica. Assim, embora alguns componentes sejam distribuídos, os componentes de tomada de decisão ficam concentrados em um único ponto.

Os IDS que possuem seus componentes dispostos em pontos diversificados e se comunicam através de mensagens são chamados de IDS distribuídos. Esse tipo de IDS pode ter grupos de detectores e analisadores em locais distintos o que fornece maior abrangência de detecção, com módulos espalhados por diferentes pontos do sistema (Asaka et al., 1999).

Comportamento Pós-detecção

O comportamento pós-detecção corresponde a geração de respostas após a detecção de um evento intrusivo. Algumas das respostas comumente geradas por IDS são confecções de relatórios de resultados, coleta de informações e reconfigurações de serviços (Bace & Mell, 2001).

Abordagens passivas pós-detecção consistem na emissão de alarmes e notificações para informar aos usuários quando um ataque é detectado. A outra classe de abordagens pós-detecção é a ativa, onde uma das principais respostas utilizadas é a coleta de informações adicionais sobre os ataques. Além disso, outra resposta viável é interromper um ataque em andamento e em seguida bloquear o acesso do atacante (Bace & Mell, 2001).

Frequência de Uso

A frequência de uso do IDS refere-se ao tempo decorrido entre os eventos que são monitorados e a análise desses eventos.

Os IDS baseados em intervalos, também conhecidos como IDS de análise periódica, não possuem fluxo contínuo entre os pontos de monitoramento e os mecanismos de análise. Os dados coletados pelos pontos de monitoramento são analisados em períodos predefinidos. Desse modo, o uso de processamento pode ser reduzido e melhor gerenciado (Ilgun, 1993). Importante ressaltar que IDS baseados em intervalos não conseguem realizar respostas ativas (Ilgun, 1993).

Diferentemente dos IDS baseados em intervalo, os IDS de monitoramento contínuo, também conhecidos como IDS de tempo real, realizam uma análise imediata nos eventos coletados pelos pontos de monitoramento utilizado por soluções baseadas em rede, pois os dados analisados são provenientes de tráfego constante de pacotes na rede, exigindo uma análise imediata (Bace & Mell, 2001). As soluções baseadas em *host* também podem operar em modo tempo real, porém esta abordagem acaba por degradar o desempenho do *host*, prejudicando as demais tarefas que o *host* esteja executando.

2.5 Considerações Finais

Neste capítulo apresentaram-se os conceitos relativos a segurança de redes de computadores e a importância dos mecanismos de proteção. Também foram abordados também os conceitos, funcionalidades, padronização e classificação de sistemas de detecção de intrusão.

No Capítulo 3 são trabalhados os conceitos de Inteligência Artificial (IA), Redes Neurais Artificiais (RNA), algoritmo *K-Nearest Neighbor* (KNN), entre outras técnicas. Por fim serão discutidas algumas características estruturais relevantes e alternativas de aplicação de abordagens e técnicas de IA.

Capítulo 3

Inteligência Artificial

3.1 Considerações Iniciais

Em Russell & Norvig (2010) são apresentadas diversas definições para o termo Inteligência Artificial (IA). Embora não haja um consenso, que de forma geral, as conceitualizações convergem para a capacidade de fazer máquinas reproduzirem atividades inteligentes e capacidades cognitivas encontradas em humanos.

Ao analisarmos de modo mais enfático as soluções atualmente propostas na área de segurança computacional, verifica-se cada vez mais forte a pesquisa e a aplicação de inteligência computacional para a melhorias no processo de detecção de intrusões.

Motivado por este contexto, neste capítulo apresenta-se conceitos de inteligência computacional, com ênfase em assuntos relacionados ao presente trabalho, tais como: mineração de dados, Redes Neurais Artificiais (RNA) e do algoritmo *K-Nearest Neighbor* (KNN), expondo algumas características estruturais relevantes e alternativas de aplicação.

Para facilitar o entendimento do Capítulo, na primeira seção são apresentados os conceitos e etapas do processo de mineração de dados. Na 3.3 seguinte são abordadas técnicas de seleção de atributos que são aplicadas na etapa de pré-processamento. Nas seções 3.4 e 3.5 são abordadas as técnicas RNA e KNN, respectivamente, essas técnicas são aplicadas na etapa de processamento do processo de mineração de dados.

3.2 Mineração de Dados

A mineração de dados consiste na utilização de técnicas com o objetivo de extrair conhecimento dos dados, sendo tal processo denominado *Knowledge Discovery in Databases* (KDD) (Fayyad, 1996).

De acordo com Fayyad (1996), KDD é o processo de extrair conhecimento de alto nível a

partir de um conjunto de dados de baixo nível. De modo complementar, cabe definir que o processo de KDD é composto por um conjunto de etapas: seleção de dados, pré-processamento, transformação de dados, mineração de dados (data mining) e pós-processamento (interpretação/avaliação dos modelos).

Esta seção é dedicada a apresentar uma visão geral o processo de KDD, abordando todas as suas fases e dedicando atenção especial a etapa de processamento, tendo em vista a sua importância para o presente trabalho, onde são aplicadas as técnicas de *data mining*.

3.2.1 Seleção de Dados

A primeira etapa do processo de KDD é a seleção de dados, a qual consiste na seleção do conjunto de dados pertencentes ao domínio que fará parte da análise. Deste modo, esta etapa possui influência nas demais fases e principalmente vai ter forte impacto sobre o resultado final do KDD, isto é, análise realizada no pós-processamento (Fayyad, 1996).

3.2.2 Pré-processamento

A etapa de pré-processamento é crucial para o sucesso do processo de KDD, tendo em vista que a qualidade dos dados vai impactar na eficácia dos algoritmos de mineração. É nesta fase que são aplicadas técnicas para eliminar dados redundantes e inconsistentes, avaliar dados discrepantes*outliers* e lidar com dados faltantes (Fayyad et al., 1996). Além disso, pode-se também utilizar técnicas para redução de atributos, com o intuito de melhorar o desempenho do algoritmo de análise.

3.2.3 Transformação dos Dados

Após o pré-processamento ter sido realizado, realiza-se uma padronização dos dados, com o intuito de facilitar a manipulação dos mesmos pelos algoritmos de análise (Norton, 1999).

3.2.4 Processamento

Apesar de todas as etapas do processo KDD terem importância, o processamento/análise recebe maior destaque para efeito deste trabalho. É nesta etapa que o conjunto de dados é submetido as técnicas de *data mining*. *Data mining* é o processo de explorar grandes quantidades de dados de modo a extrair conhecimento dos mesmos (Fayyad et al., 1996).

Não existe técnica ideal para todos os problemas de *data mining*. Cada método possui vantagens e desvantagens, e se adequa melhor face as características do problema e objeti-

vos. Entre as técnicas comumente usadas cita-se: Redes Neurais Artificiais (Haykin, 2001), Árvores de Decisão (Russell & Norvig, 2010), algoritmos de classificação baseada em exemplos (Mitchell, 1997), Support Vector Machine (SVM) (Steinwart & Christmann, 2008), entre outros.

3.2.5 Pós-processamento

O pós-processamento é a ultima fase do processo de KDD, momento em que os resultados alcançados, face a utilização dos métodos de análise aplicados sobre o conjunto de dados, são analisados e avaliados (Fayyad et al., 1996).

Após a apresentação de todas as fases do KDD, são abordadas na próxima seção algumas técnicas de seleção de atributos, que fazem parte da etapa de preprocessamento.

3.3 Seleção de Atributos

A atividade de seleção de atributos tem como objetivo selecionar um subconjunto de atributos para serem utilizados, buscando uma melhoria na taxa de classificação e no tempo de processamento.

Os atributos podem ser selecionados de acordo com a sua relevância e qualidade para a tarefa de classificação. Uma das maneiras para se medir a qualidade de um atributo é avaliar o seu grau de associação com a classe através da medida do ganho de informação (Manzoor et al., 2017).

Considerando uma base de dados $D(A_1, A_2, ..., A_n, C)$, com $n >= 1$, onde C é o atributo classe e o seu domínio é $(c_1, c_2, ..., c_m)$, com $m >= 2$, assume-se que os valores dos atributos e da classe são discretos.

A medida ganho de informação é baseada no conceito de entropia. Considerando como entropia a medida de impureza e falta de homogeneidade de um atributo, a fórmula da entropia é apresentada na Equação 3.1 para um atributo A (Quinlan, 1986), cujo domínio é $(a_1, a_2, ..., a_k)$, com $k >= 1$. Os valores p_i, com $1 <= i <= k$, correspondem a razão entre o número de instâncias da base em que ocorre o valor a_i para o atributo A e o número total de instâncias.

$$Entropia(A) = \sum_{i=1}^{m} p_i \log_2(p_i) \tag{3.1}$$

A entropia da classe C, representada por $Entropia(C)$, pode ser calculada da mesma forma, considerando p_j a razão entre o número de instâncias em que o valor c_j da classe, com $1 <= j <= m$, ocorre na base e o número total de instâncias (Quinlan, 1996).

O ganho de informação de um atributo é determinado pela redução da sua entropia. A Equação 3.2 calcula o ganho de informação de um atributo A em relação a classe C (Karegowda et al., 2010).

$$Ganho(C, A) = Entropia(C) - Entropia(C, A) \tag{3.2}$$

onde $Entropia(C)$ é a entropia da classe C, e $Entropia(C, A)$ é entropia da classe relativa ao atributo A, Calculada pela Equação 3.3. Onde $Entropia(C|A = A_j)$ é a entropia relativa ao subconjunto de instâncias que tem um valor A_j para o atributo A. Se A é um bom descritor para a classe, cada valor de A terá uma baixa entropia distribuída entre as classes, ou seja, cada valor deve estar predominantemente em uma classe.

$$Entropia(C, A) = -\sum_{j=1}^{m} p_(A, C) \log_2(p_(A, C)) \tag{3.3}$$

Temos que o valor de ganho de informação consiste na variação da impureza, ou seja, atributos com menor entropia possuem maior ganho de informação. Esta medida tem um *bias* natural pois favorece atributos que possuem muitos valores (Quinlan, 1993).

A medida de razão de ganho de informação tenta corrigir o *bias* do Ganho de Informação (Quinlan, 1993).

$$Razão\ de\ Ganho(C, A) = \frac{Ganho(C, A)}{Entropia(A)} = \frac{Entropia(C) - Entropia(C, A)}{Entropia(A)} \tag{3.4}$$

Após o cálculo do mérito de cada atributo pela Equação 3.4, é gerado um ranking e seleciona-se os N melhores atributos desse ranking de acordo com algum critério.

Nas próximas seções são abordadas as técnicas RNA e KNN. Essas técnicas são utilizadas na etapa de processamento e foram utilizadas para compor o método híbrido proposto neste trabalho.

3.4 Redes Neurais Artificiais

O trabalho em Redes Neurais Artificiais (RNA), desde sua concepção, foi motivado pelo reconhecimento de que o cérebro humano processa informações complexas de modo diferente e mais efetivo, quando comparado a computadores digitais convencionais(Haykin, 2001).

As RNA possuem diferentes arquiteturas e tem sido aplicadas no apoio a resolução de problemas em diferentes áreas de conhecimento, dado que sua estrutura paralelamente distri-

buída e sua capacidade de aprender e generalizar, constitui uma poderosa ferramenta para várias aplicações. Dentre as áreas de conhecimento em que são aplicadas RNA, citam-se: reconhecimento de padrões (Azad et al., 2014), finanças (Kristjanpoller & Minutolo, 2015), química (Silva et al., 2015), etc.

As RNA tem como inspiração as redes neurais biológica. Na próxima subseção apresentam-se tais conceitos.

3.4.1 Inspiração Biológica

Tendo em mente a inspiração das redes neurais na biologia, é apropriado abordamos brevemente o cérebro humano e seus níveis estruturais de organização.

De acordo com (Haykin, 2001), o sistema nervoso humano possui três (3) estágios, os quais estão ilustrados na Figura 3.1. O órgão central do sistema é o cérebro, representado pela rede neural nervosa, que recebe continuamente estímulos, os processa e toma decisões. Os receptores são responsáveis por receber estímulos do corpo humano ou do ambiente externo e convertê-los em impulsos elétricos, que transmitem informações para a rede neural. Os atuadores por sua vez, convertem os impulsos elétricos gerados pela rede neural em respostas discerníveis como saídas do sistema. As setas da esquerda para direita indicam a propagação do sinal de informação e as setas da direita para esquerda indicam a presença de realimentação no sistema.

Figura 3.1: Representação em diagrama em blocos do sistema nervoso. Adaptado de (Haykin, 2001).

O corpo humano é constituído por complexos circuitos neurais cerebrais, os quais possuem inúmeras interconexões. Estes circuitos são formados por células nervosas chamadas neurônios. O neurônio é a unidade fundamental dos circuitos neurais, com a função básica de receber, processar e enviar informações. Estima-se que existam aproximadamente 86 bilhões de neurônios no encéfalo humano adulto (Haykin, 2001).

Embora os neurônios possuam formas e tamanhos diferentes, cada um apresenta quatro regiões especializadas morfologicamente com papéis específicos. Para ilustrar, na Figura 3.2 apresentam-se os dendritos, o corpo celular, os axônios, e as terminações do axônio.

O corpo celular é onde esta localizado o núcleo e as organelas necessárias para manter a vitalidade do neurônio. Ao redor do corpo celular de um neurônio exitem conjuntos

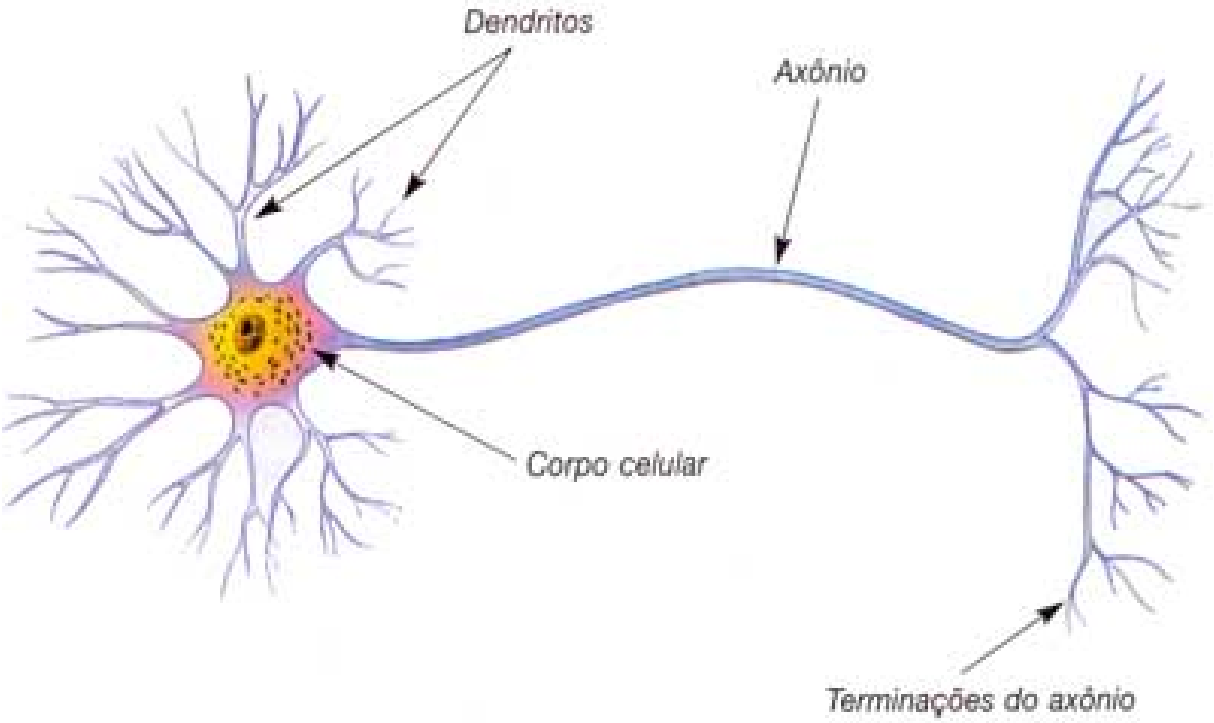


Figura 3.2: Neurônio biológico. Fonte: (Machado, 2005a).

de fibra irregulares, conhecidos como dendritos. Os dendritos são numerosos prolongamentos especializados em receber estímulos e funcionam como dispositivos de entrada, recebendo estímulos nervosos do ambiente e de outros neurônios e os repassando para o corpo celular (Machado, 2005a).

O axônio é um prolongamento longo e fino, e consiste na parte do neurônio responsável pela condução de impulsos que partem do corpo celular até pequenas ramificações conhecidas como terminações do axônio. Essas ramificações funcionam como dispositivo de saída, seja para um músculo ou para outros neurônios (Squire et al., 2012)(Kovács, 2002).

Os neurônios entram em contato com outros neurônios passando-lhes informações através das terminações axônicas. De acordo com (Machado, 2005a), os locais onde ocorrem esses contatos são chamados de sinapses. Os neurônios que transmitem as informações são chamados pré-sinápticos e os que as recebem são chamados pós-sinápticos. A transmissão dos sinais através das sinapses pode tanto excitar como inibir os neurônios pós-sinápticos (Lent, 2004).

Um fenômeno interessante que ocorre no neurônio é o acúmulo de um potencial de ativação na membrana axonal, que é definido como uma onda de despolarização de curta duração, através do recebimento de estímulos vindos de outros neurônios.

A membrana de um neurônio está polarizada quando possui carga elétrica positiva do lado externo e negativa do lado interno. Quando um estímulo chega ao neurônio, pode ocorrer alteração da permeabilidade da membrana, permitindo entrada de sódio e saída de potássio, e, deste modo, ocorre uma inversão de cargas e a membrana fica despolarizada (Machado, 2005a).

Tais estímulos recebidos pelo neurônio podem agir de maneira excitatória, despolarizando a membrana, ou de maneira inibitória, hiperpolarizando a membrana. Quando uma membrana sofre despolarização suficientemente alta para superar um valor conhecido como limiar de disparo, o impulso nervoso é transmitido de um neurônio para outro. Logo após a passagem do impulso a membrana sofre repolarização e volta ao seu estado de repouso (Squire et al., 2012).

Os neurônios biológicos são cinco (5) vezes mais lentos que as portas lógicas de silício. No entanto, o cérebro compensa isso através do grande número de neurônios, com conexões maciças entre si, formando uma enorme estrutura paralela de processamento (Machado, 2005a). O processamento inteligente das redes neurais é realizado através dessas interligações dos neurônios, que são organizados de forma complexa, não-linear, e paralela (Haykin, 2001).

3.4.2 Fundamentos Teóricos

O cérebro possui neurônios densamente interconectados formando uma estrutura altamente complexa, inspirados nessa estrutura os neurônios artificiais foram propostos (Haykin, 2001).

O neurônio artificial é uma estrutura lógico matemática que tem como objetivo simular a forma, o comportamento e as funções do neurônio biológico. Os dentritos são substituídos por entradas, e as ligações dessas entradas com o corpo celular artificial são conhecidas como pesos, os quais simulam as sinapses. Os estímulos recebidos pelas entradas são processados pela função de soma e o limiar de disparo do neurônio biológico é simulado pela função de ativação no neurônio artificial (Chua & Yang, 1988).

A Figura 3.3 apresenta o modelo simplificado de um neurônio artificial. Onde o neurônio k recebe uma entrada x $(x_1, x_2, ..., x_n)$ que entra pela sinapse j.

Cada sinal x_j da entrada x, que se conecta com sinapse j ,é multiplicada por um peso w_{kj}. O resultado deste processo passa por uma função que soma os sinais de entrada ponderados pelos pesos das respectivas sinapses juntamente com um $bias$ externo (b_k). Em relação a determinado peso sináptico w_{kj}, o primeiro índice (k) refere-se ao neurônio em questão e o segundo índice (j) refere-se a entrada da sinapse ao qual o peso esta relacionado (Haykin, 2001).

O $bias$ b_k, citado anteriormente, possui a atribuição de aumentar ou diminuir o valor gerado pelo somador, antes de repassá-lo para uma função de ativação. Tal função transforma a saída em um intervalo fechado geralmente entre $[0, 1]$ ou $[-1, 1]$, o qual é repassado para outros neurônios (Haykin, 2001)(Fausett & Fausett, 1994).

Em termos matemáticos um neurônio pode ser descrito pelo seguinte par de equações (Fausett & Fausett, 1994):

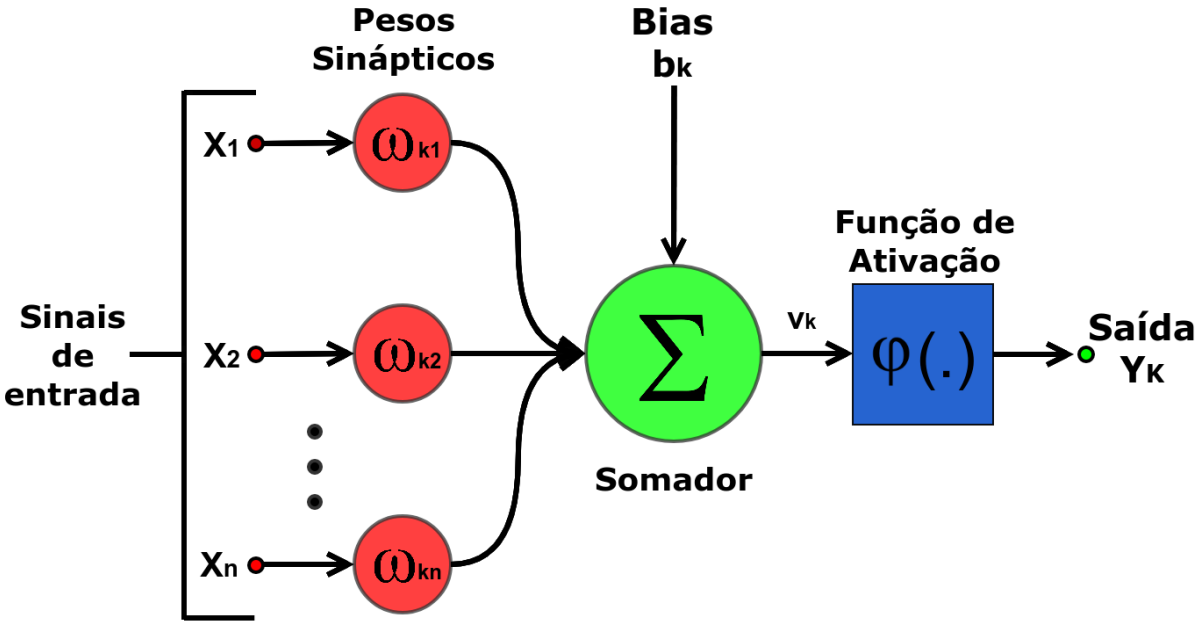


Figura 3.3: Modelo simplificado de um Neurônio Artificial. Adaptado de (Haykin, 2001).

$$v_k = \sum_{j=1}^{m} w_{kj} x_j \tag{3.5}$$

$$y_k = \varphi(v_k + b_k) \tag{3.6}$$

De acordo com Dalton & Deshmane (1991), os pesos sinápticos possuem importante papel em neurônios artificiais. O propósito dos pesos é ponderar a influência dos sinais de entrada nos neurônios pós-sinápticos. Os pesos positivos tendem a incrementar o nível de ativação de um neurônio, o que consiste em uma conexão excitatória. Já os pesos negativos tendem a diminuir o nível de ativação, sendo chamadas conexões inibitórias.

A função de ativação, que na Figura 3.3 está representada por $\varphi(.)$, define qual será a saída do neurônio de acordo com a entrada recebida (Haykin, 2001). Portanto, temos que a função de ativação $\varphi(v)$ define a saída do neurônio, de acordo com o resultado do somatório (v) das entradas ponderadas.

Existem vários tipos de função de ativação utilizadas em redes neurais, entre elas:

- **Função Limiar:** A função de ativação limiar, descrita na Figura 3.4 e na Equação 3.7 (Haykin, 2001), define a saída em termos do resultado do somatório v:

$$\varphi(v) = \begin{cases} 1 \ \ se \ \ v \geq 0 \\ 0 \ \ se \ \ v < 0 \end{cases} \tag{3.7}$$

Portanto, neste modelo, caso o resultado do somador do neurônio em questão for maior ou

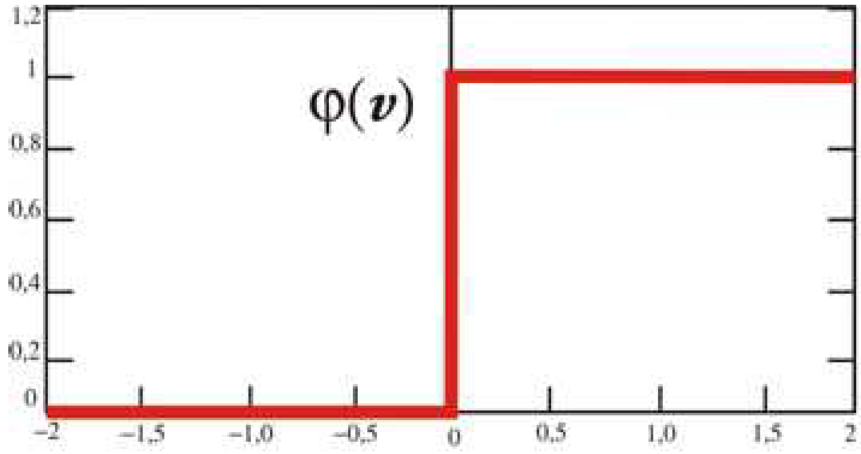


Figura 3.4: Função de ativação Limiar. Fonte: (Haykin, 2001).

igual a 0 (positivo), a saída do neurônio assume valor . Em contrapartida, se o resultado do somatório for negativo, o valor de saída do neurônio será 0. Desse modo, a saída do neurônio *k* que emprega essa função de ativação pode ser expressa como (Haykin, 2001):

$$y_k = \begin{cases} 1 \ \ se \ \ v_k \geq 0 \\ 0 \ \ se \ \ v_k < 0 \end{cases} \tag{3.8}$$

onde v_k corresponde a

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + b_j \tag{3.9}$$

O modelo de neurônio que utiliza esse tipo de função de ativação é conhecido como modelo McCulloch-Pitts devido ao trabalho pioneiro de McCulloch e Pitts em 1943 (Haykin, 2001).

- **Função Linear por Partes:** A função linear por partes é descrita na Figura 3.5. Este tipo de função possui duas situações particulares. Possui resultado linear se a região de operação é mantida fora da faixa de saturação, ou seja, se v estiver no intervalo -0.5 até 0.5. Caso a faixa de saturação seja atingida, ou seja, v é menor que -0.5 ou maior que 0.5, o comportamento da função se iguala a função limiar. A Equação 3.10 apresenta a função linear por partes (Haykin, 2001).

$$\varphi(v) = \begin{cases} 1 \ \ se \ \ v \geq 0.5 \\ v \ \ se \ \ 0.5 > v > -0.5 \\ 0 \ \ se \ \ v \leq 0.5 \end{cases} \tag{3.10}$$

- **Função Sigmóide:** A função de ativação sigmóide tem gráfico em forma de "S", e é uma das funções de ativação mais usadas em RNA. Ela exibe um balanceamento adequado entre comportamento linear e não linear. Um exemplo de comportamento sigmóide é

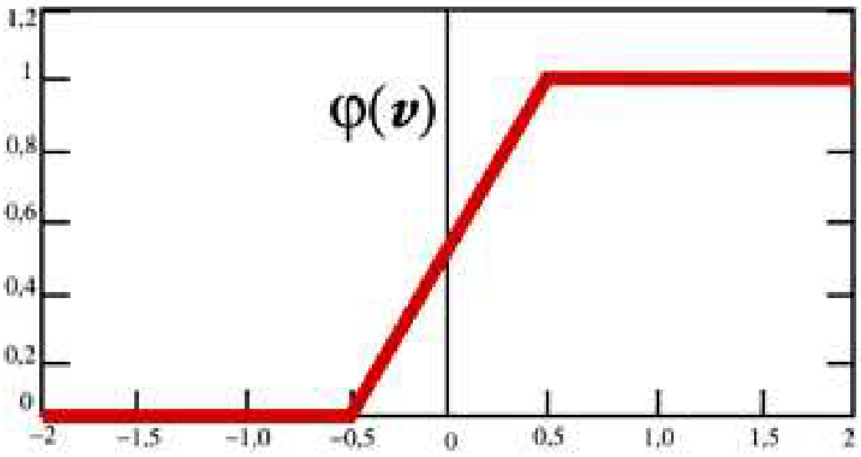


Figura 3.5: Função de ativação Linear por partes. Fonte: (Haykin, 2001).

a função Logística, definida na Equação 3.11, onde a é o parâmetro de inclinação da função sigmóide. Quando este parâmetro tende ao infinito, a função aproxima-se da função limiar. Enquanto a função limiar assume o valor de 0 ou 1, ao passo que a função logística assume um intervalo contínuo de valores entre 0 e 1 (Haykin, 2001). A Figura 3.6 apresenta funções sigmóides com variações no valor de a e as variações de inclinação obtidas a partir disso.

$$\varphi(v) = \frac{1}{1 + exp(-av)} \tag{3.11}$$

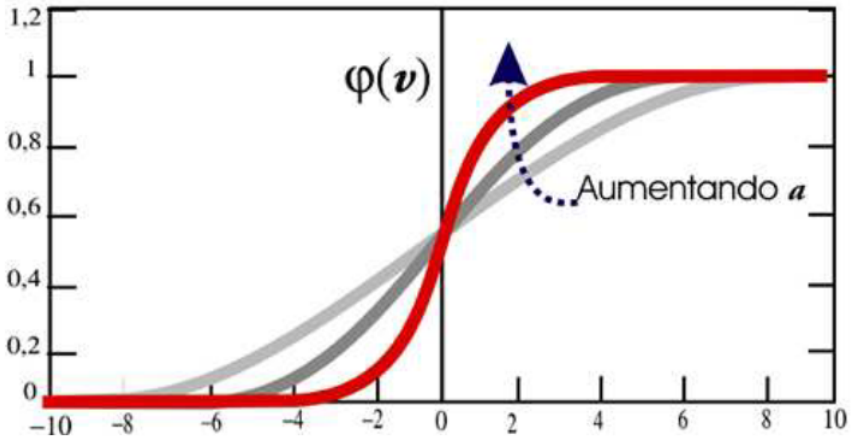


Figura 3.6: Função de ativação sigmóide para parâmetro de inclinação a variável. Fonte: (Haykin, 2001).

As funções de ativação ilustradas nas Equações 3.7, 3.10 e 3.11 se estendem de 0 à 1. No entanto, algumas vezes é necessário que a função de ativação se estenda de -1 à 1. Desse modo, a função limiar da Equação 3.7 é agora definida na Equação 3.12 para valores entre -1 e 1 (Haykin, 2001).

$$\varphi(v) = \begin{cases} 1 \ \ se \ \ v > 0 \\ 0 \ \ se \ \ v = 0 \\ -1 \ \ se \ \ v < 0 \end{cases} \tag{3.12}$$

Para a forma correspondente de uma função sigmóide pode-se usar a função tangente hiperbólica, que é definida na Equação 3.13. Está função é similar a sigmóide logística, porém seus valores contínuos variam de -1 à 1.

$$\varphi(v) = tanh(v) \tag{3.13}$$

O fato de permitir que uma função de ativação assuma valores negativos fornece benefícios analíticos e vantagens durante a fase de treinamento (Haykin, 2001).

3.4.3 Processo de Aprendizagem

Uma característica essencial para uma RNA é a habilidade de aprender através de seu ambiente e de melhorar seu desempenho por meio de treinamento. Uma RNA aprende mais sobre o ambiente em que está inserida através de um processo iterativo de ajuste dos seus pesos sinápticos e níveis de bias, processo este definido como treinamento. A RNA se torna mais conhecedora de seu ambiente após cada iteração do processo de treinamento (Haykin, 2001).

O processo de aprendizagem comumente envolve os seguintes eventos:

- RNA é estimulada por um ambiente;
- RNA sofre modificações em seus parâmetros devido a esta estimulação;
- RNA responde de maneira nova devido as modificações feitas em seus parâmetros.

O conhecimento da RNA é representado por meio dos pesos sinápticos, formando uma representação compacta e distribuída desse conhecimento e proporcionando capacidades de generalização e adaptabilidade à rede neural. No entanto, o fato das RNA não serem baseadas em regras, torna impossível explica, de forma abrangente, o processo computacional pelo qual tomou a decisão exposta por suas saídas (Haykin, 2001).

A seguir são apresentados sucintamente os dois (2) paradigmas de aprendizado comumente adotados:

- **Aprendizagem Supervisionada:** O paradigma de aprendizagem supervisionada é também conhecido como aprendizado com professor, devido ao fato de receber auxílio de um supervisor que possui conhecimento sobre o ambiente e o representa através de conjunto

de treinamento, com entradas e saídas desejadas. A Figura 3.7 apresenta o diagrama em blocos representando este modo de aprendizado.

Conforme pode ser observado na Figura 3.7 ,o supervisor passa o seu conhecimento através de um conjunto de entradas e respectivas saídas desejadas, sendo possível assim avaliar o desempenho obtido pela rede e então melhorá-lo (Dalton & Deshmane, 1991).

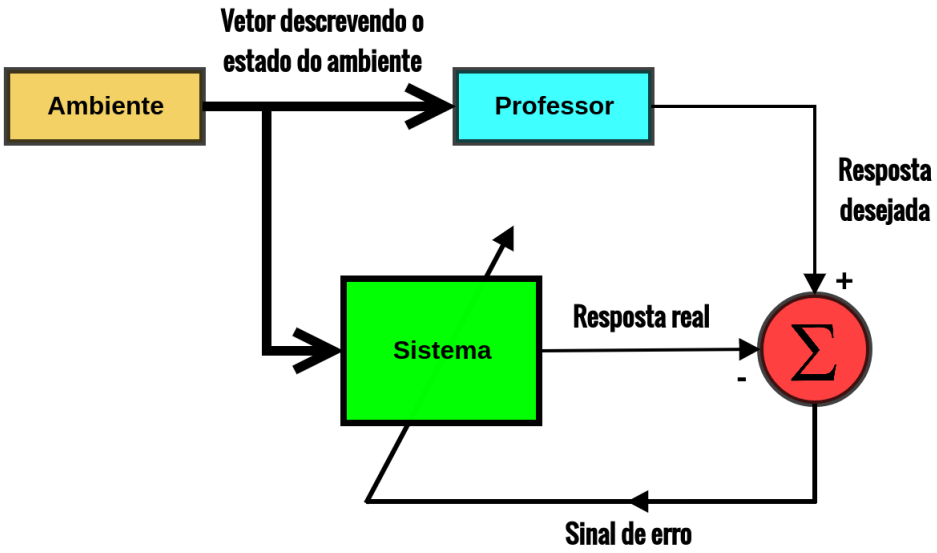


Figura 3.7: Diagrama em blocos da aprendizagem supervisionada. Adaptado de (Haykin, 2001).

Após o processo de treinamento, o professor terá transferido seu conhecimento para a RNA, podendo ser dispensado, deixando a rede lidar com as novas situações por si própria (Lima, 2005).

- **Aprendizagem Não Supervisionada:** Diferentemente do paradigma de aprendizagem supervisionada, onde existe o papel de um professor que supervisiona o treinamento, no paradigma de aprendizagem não supervisionada o processo de aprendizagem ocorre sem a presença de um supervisor (Zurada, 1992). Este tipo de aprendizado envolve processos de competição e colaboração entre os neurônios.

Existem muitas técnicas de aprendizagem para projeto de RNA, cada uma com vantagens e desvantagens específicas. A diferença neste modo de aprendizagem é que é feito o ajuste dos pesos sinápticos dos neurônios. Entre as técnicas aplicadas em RNA, destacam-se a aprendizagem por correção de erro, a aprendizagem baseada em memória, e a aprendizagem competitiva.

A técnica de *aprendizagem por correção de erro* se enquadra no paradigma de aprendizado supervisionado. Nesta abordagem os pesos são atualizados de acordo com o erro obtido entre a saída obtida e a saída esperada, com o objetivo de minimizar este erro (Haykin, 2001).

Considerando o caso simples de um neurônio k, que é o único nó na camada de saída de uma rede neural alimentada adiante, conforme ilustrado na Figura 3.8. O neurônio k é

estimulado por um sinal $x(n)$, produzido por uma ou mais camadas ocultas, estas por sua vez acionadas por um estímulo aplicado nas camadas de entrada. O argumento n representa o passo de tempo de um processo iterativo de ajuste dos pesos sinápticos do neurônio k (Haykin, 2001).

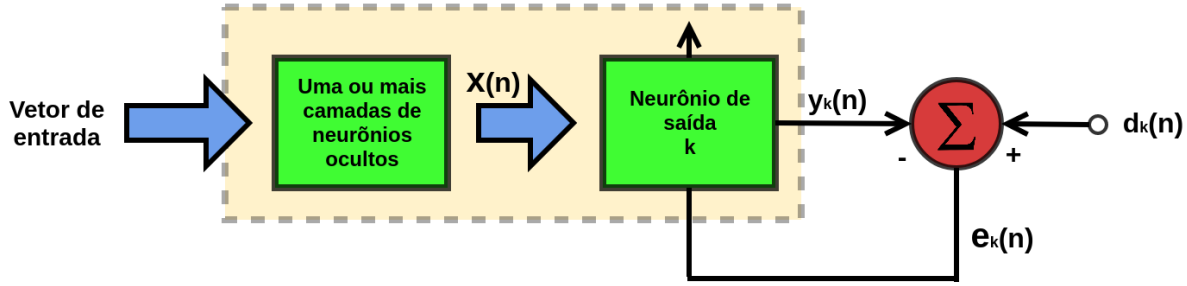


Figura 3.8: Diagrama em blocos de uma rede neural, ressaltando único neurônio da camada de saída. Adaptado de (Haykin, 2001).

O sinal de saída do neurônio k é representado por $y_k(n)$. Este sinal é comparado com a saída desejada ou saída alvo, que é representada por $d_k(n)$. Considerando essas duas saídas é determinado o sinal de erro $e_k(n)$, através da Equação 3.14 (Haykin, 2001).

$$e_k(n) = d_k(n) - y_k(n) \tag{3.14}$$

A partir de então é realizado um conjunto de ajustes corretivos nos pesos sinápticos do neurônio, com o objetivo de aproximar passo a passo o sinal de saída $y_k(n)$ da saída desejada $d_k(n)$.

Para determinar o ajuste a ser aplicado sobre o peso sináptico de um neurônio pode-se utilizar a função delta apresentada na Equação 3.15 (Haykin, 2001). Considerando que $w_{kj}(n)$ representa o valor de peso sináptico w_{kj} do neurônio k, estimulado por um elemento $x_j(n)$ do vetor de entrada $x(n)$ no passo n de tempo. η é uma constante positiva que determina a taxa de aprendizado (Haykin, 2001).

$$\Delta w_{kj}(n) = \eta e_k(n) x_j(n) \tag{3.15}$$

Desse modo, o novo valor para o peso w_{kj} é obtido através da Equação 3.16 (Haykin, 2001).

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj}(n) \tag{3.16}$$

Os termos $w_{kj}(n)$ e $w_{kj}(n+1)$ são os valores antigo e novo do peso sináptico w_{kj}. A Figura 3.9 ilustra o processo de aprendizado por correção de erro.

Na *aprendizagem baseada em memória*, todas as amostras apresentadas são armazenados

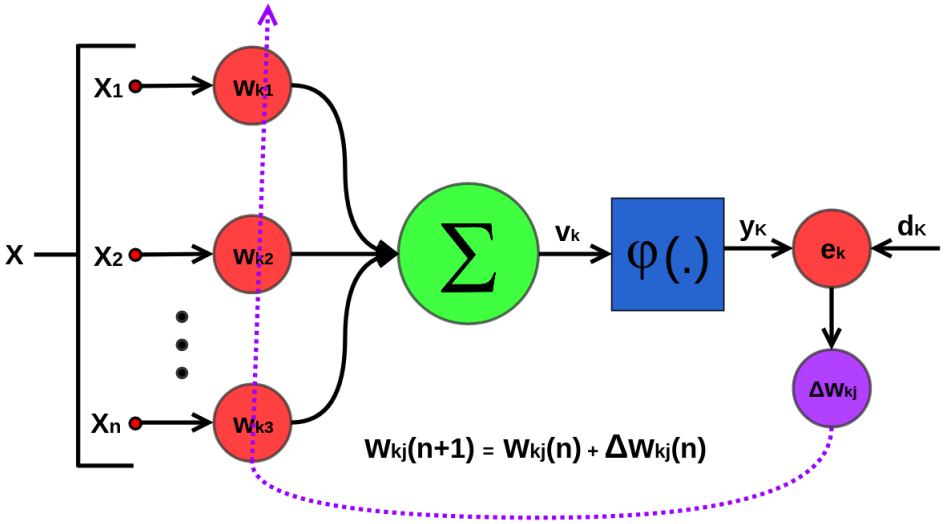


Figura 3.9: Processo de aprendizagem por Correção de Erro. Adaptado de (Haykin, 2001).

em uma memória de exemplos de entrada e saída classificadas corretamente. As amostras dessa base de exemplo servirão de base para as classificações futuras, através da busca por similaridades (Haykin, 2001).

Na técnica de *aprendizagem competitiva*, os neurônios competem entre si para determinar apenas um neurônio ativo, que será a saída da rede neural. Portanto a camada de saída terá apenas um neurônio ativo de cada vez. Nesse tipo de aprendizado, entradas que possuem semelhanças tendem a excitar o mesmo neurônio de saída (Rojas, 2013).

3.4.4 Topologias de Redes Neurais

A maneira que os neurônios estão dispostos em um rede neural está diretamente relacionada com o algoritmo de aprendizagem. Para efeito do presente trabalho, abordaremos as duas principais classes de topologias de redes neurais, as redes neurais alimentadas adiante (Russell & Norvig, 2010) e as redes neurais recorrentes (Haykin, 2001).

Em uma RNA em que os neurônios são dispostos em camadas, normalmente aplica-se a arquitetura de rede alimentadas adiante, conhecidas face a língua Inglesa como redes *feed-foward*. Este modelo não possui ciclos de realimentação entre os neurônios, portando o fluxo do processo sináptico ocorre da camada de entrada em direção a camada de saída. A RNA alimentadas adiante podem ser classificadas em redes de camada única e de múltiplas camadas (Mitchell, 1997).

As redes neurais de camada única possuem apenas uma camada de neurônios, a qual realiza todo o processamento. Uma RNA com esta topologia é ilustrada na Figura 3.10. Neste exemplo a RNA está estruturada com seis neurônios de entrada e dois neurônios de saída. A nomenclatura *camada única* se refere a camada de saída da rede. A camada de entrada não é

considerada em nenhuma topologia, pois não realiza nenhum tipo de processamento (Mitchell, 1997).

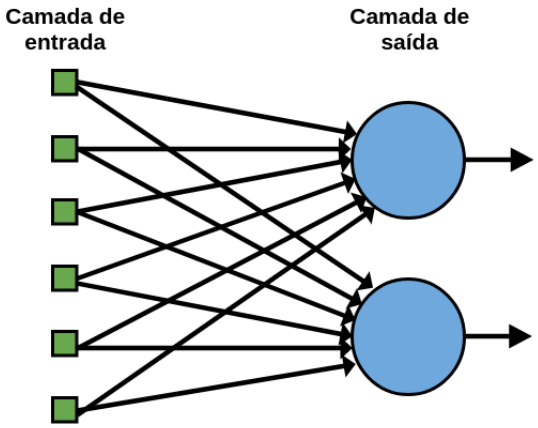


Figura 3.10: Rede neural alimentada adiante com camada única. Adaptado de (Russell & Norvig, 2010).

As redes neurais de múltiplas camadas se diferenciam das rna de camada única, justamente pela presença de uma ou mais camadas ocultas, conforme é apresentado na Figura 3.11. As camadas ocultas possuem a função de intermediar de forma útil e não-linear o processamento entre a entrada e a saída da rede, aumentando o poder de processamento, tornando a rede capaz de extrair estatísticas de ordem elevada. As saídas da camada de entrada se tornam entrada nos neurônios de segunda camada, e assim sucessivamente, até a camada de saída (Haykin, 2001).

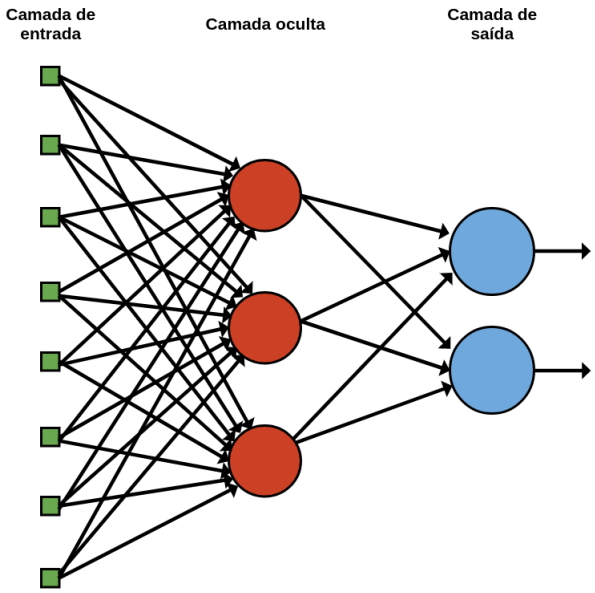


Figura 3.11: Rede neural alimentada adiante com múltiplas camadas. Adaptado de (Russell & Norvig, 2010).

Uma RNA é considerada totalmente conectada quando cada neurônio de uma determinada camada esta conectado diretamente a todos os neurônios da camada seguinte. Caso exista

um neurônio que não possua conexão com algum neurônio da camada seguinte, esta rede é considerada parcialmente conectada (Haykin, 2001).

Ao contrário das redes alimentadas adiante, as redes recorrentes também conhecidas como redes *feedback* (do ingês), possuem no mínimo um processo de repetição entre seus neurônios. A Figura 3.12 apresenta uma rede desta arquitetura.

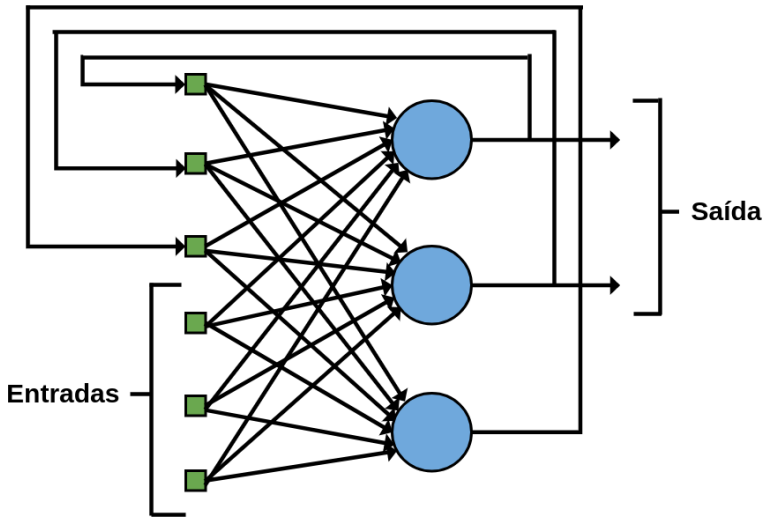


Figura 3.12: Rede neural recorrente. Adaptada de (Haykin, 2001).

A realimentação pode estar presente em neurônios de camadas anteriores ou na mesma camada,ou até mesmo a saída de um neurônio pode alimentar sua própria entrada, sendo este último caso denominado auto-realimentação. De acordo com (Russell & Norvig, 2010), a presença de laços de realimentação em RNA pode fazer com que elas apresentem comportamentos instáveis e caóticos, no entanto também tem impacto positivo, tanto na capacidade de aprendizado da rede quanto em seu desempenho.

3.4.5 Redes Neurais Perceptron

Em 1958, Rosenblatt (1958) propôs um modelo de rede neural chamado de Perceptron. Neste modelo, os neurônios eram dispostos exclusivamente em camadas de entrada e saída e os pesos das conexões eram adaptados com o objetivo de obter uma melhor performance da rede.

A rede Perceptron é uma RNA simples, aplicável para a classificação de padrões linearmente separáveis. Neste modelo cada neurônio da camada de entrada é diretamente conectado em cada neurônio da camada de saída. Desse modo as entradas são diretamente mapeadas em um conjunto de padrões de saída, não sendo possível a formação de uma representação interna (Russell & Norvig, 2010).

Neste tipo de RNA, os padrões de entrada com estruturas similares mas que pertencem a classes diferentes, não são possíveis de serem identificados e corretamente classificados. Isso

ocorre porque as RNA Perceptron funcionam adequadamente quando as classes envolvidas são linearmente separáveis. Isso significa que os padrões envolvidos devem ser separados suficientemente para assegurar que a superfície de decisão consista em um hiperplano. A Figura 3.13a apresenta um exemplo bidimensional onde os dois padrões $\mathscr{C}_1$ e $\mathscr{C}_2$ estão separados suficientemente, sendo possível traçar uma superfície de decisão em forma de hiperplano, neste caso uma reta. Entretanto, se os dois padrões forem muito próximos, como o exemplo apresentado na Figura 3.13b, não é possível utilizar um hiperplano como superfície de decisão, e portanto, os padrões não são linearmente separáveis. Esta é uma situação que está além da capacidade das redes Perceptron de uma única camada (Haykin, 2001).

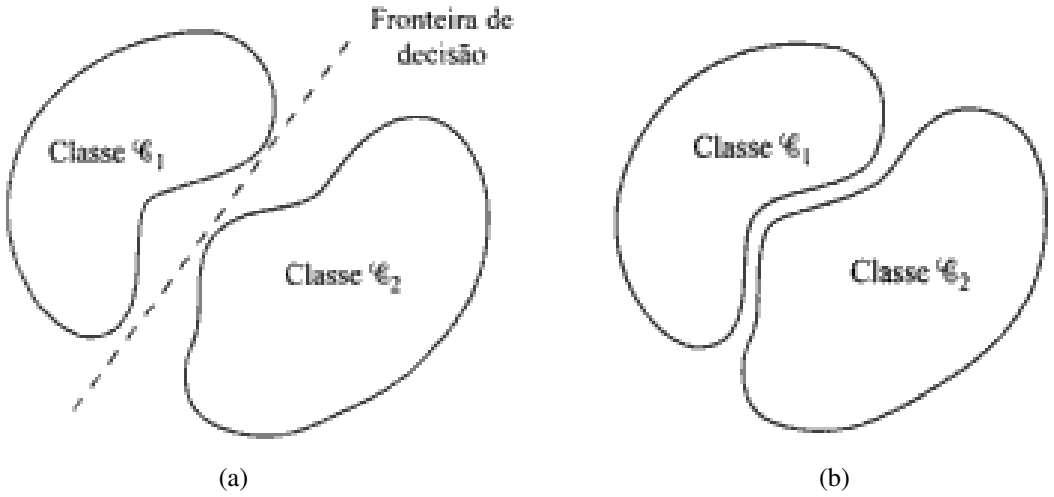


Figura 3.13: (a) Padrões linearmente separáveis. (b) Padrão não linearmente separáveis. Fonte: (Haykin, 2001).

Exemplos clássicos de padrões linearmente separáveis são os conectivos lógicos E (AND) e OU (OR), ilustrados nas Figuras 3.14a e 3.14b. A Figura3.14c apresenta a representação gráfica do padrão não linearmente separável OU Exclusivo (XOR), que está além das capacidades de uma rede perceptron de uma única camada, pois não é possível construir uma linha reta como fronteira de decisão entre as classes (Rojas, 2013).

As limitações do Perceptron de Rosenblatt (1958) foram abordadas em uma fundamentação matemática sólida no livro *Perceptrons* de Marvin Minsky e Seymour Papert (Minsky & Papert, 1969). Os autores demostraram que as RNA Perceptron de uma camada não são capazes de solucionar problemas que não sejam linearmente separáveis, e além disso, não acreditavam na possibilidade de desenvolvimento de um método de treinamento para redes Perceptron com mais de uma camada, e concluíram assim, que as redes neurais sempre seriam suscetíveis a esta limitação. A publicação deste livro lançou sérias dúvidas sobre as capacidades computacionais das redes neurais e até causou desinteresse de alguns pesquisadores, até que novas técnicas surgiram para superar essas limitações, como as perceptron de múltiplas camadas, que são abordadas na próxima seção (Engelbrecht, 2007).

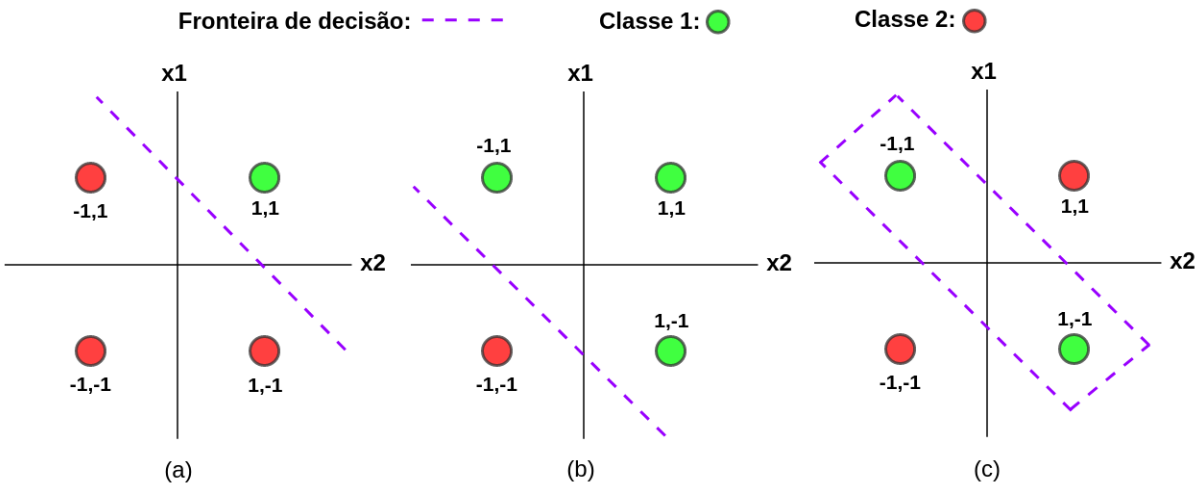


Figura 3.14: (a) Representação gráfica de padrão linearmente separável (conectivo E). (b) Representação gráfica de padrão linearmente separável (conectivo OU). (c) Representação gráfica de padrão não linearmente separável (conectivo XOR). Fonte: (Lima, 2005).

3.4.6 Rede Perceptron de Múltiplas Camadas

Para solucionar as limitações das RNA Perceptrons simples surgiram as redes neurais alimentadas adiante com múltiplas camadas, e por isso conhecidas como *Multi Layer Perceptron* (MLP). Esse tipo de rede neural consiste em um conjunto de unidades sensoriais na camada de entrada, uma ou mais camadas ocultas de neurônios computacionais e uma camada de saída de neurônios computacionais. A propagação do sinal nesse tipo de rede ocorre para frente através da rede, camada por camada (Haykin, 2001).

As demonstrações das limitações das redes neurais de uma única camada foram um fator significante para o declínio do interesse em redes neurais na década de 70. No entanto, a descoberta e disseminação generalizada de método efetivo de treinamento para redes de múltiplas camadas fez ressurgir o foco nas RNA como uma ferramenta para resolver uma grande variedade de problemas (Rumelhart et al., 1985).

O treinamento realizado em redes de múltiplas camadas é aplicação do método supervisionado por meio do algoritmo de treinamento conhecido como retro-propagação de erro (*back-propagation*), o qual é baseado na aprendizagem por correção de erro (Haykin, 2001).

O processo de aprendizagem por retro-propagação de erro consiste de dois passos principais. No primeiro, o padrão apresentado à camada de entrada é propagado pelas camadas subsequentes, produzindo uma resposta pela camada de saída. Nesta etapa os pesos sinápticos se mantém inalterados em toda a rede (Haykin, 2001).

No segundo passo, os pesos sinápticos são ajustados de acordo com a regra de correção de erro apresentada na Equação 3.15 da seção 3.4.3. A resposta gerada pela camada de saída é comparada com a saída desejada, produzindo um sinal de erro. Este sinal de erro então é propagado para trás através da rede, camada por camada, de modo a ajustar os pesos sinápticos

internos de acordo com a regra delta, estimulando assim que as respostas geradas pela rede sejam estatisticamente mais próximas das respostas desejadas (Haykin, 2001).

O algoritmo de retro-propagação converge quando o sinal de erro for suficientemente pequeno, sendo este um dos critérios de parada para finalizar o treinamento. Após o término do processo de treinamento, a RNA está treinada e pronta para ser utilizada, passando a operar apenas em modo progressivo (*feedfoward*) (Lima, 2005).

O treinamento de redes neurais utilizando o algoritmo *backpropagation* possui alguns parâmetros que devem ser ajustados de modo a otimizar o aprendizado da rede. O parâmetro *taxa de aprendizagem* é utilizado para definir a taxa de variação dos pesos sinápticos da rede. Neste sentido, quanto menor for a taxa de aprendizagem, menor será a atualização dos pesos sinápticos a cada iteração do processo de aprendizagem, proporcionando uma atualização gradativa dos pesos, e tendo como custo de um tempo de treinamento normalmente mais longo. Aumentando a taxa de aprendizagem, o processo de treinamento é acelerado com ajustes mais significativos dos pesos sinápticos a cada iteração. No entanto, taxas de aprendizado muito alta podem ocasionar oscilações no treinamento, deixando a rede instável e comprometendo o processo de aprendizagem (Danh et al., 1999).

O problema das oscilações no treinamento causadas por uma taxa de aprendizado alta pode ser contornado adicionando o termo de *momentum* à regra delta. Esse termo consiste em um número positivo comumente conhecido como *contante de momento*, o qual tem como objetivo aumentar a velocidade do treinamento e diminuir os riscos de instabilidade (Mitchell, 1997).

O aprendizado utilizando o algoritmo de retro-propagação resulta das várias apresentações de um conjunto de exemplos de treinamento a rede. Uma apresentação completa do conjunto de treinamento é denominado uma *época*. O processo de aprendizado é realizado de época em época, até que os pesos sinápticos e o *bias* sejam estabilizados e o erro médio convirja para um valor mínimo. A forma de apresentação dos exemplos do conjunto de treinamento à rede pode ser realizado de modo sequencial, onde o ajuste dos pesos ocorre a cada exemplo de treinamento apresentado à rede, ou de modo por lote, onde os pesos são ajustados somente após todos os exemplos de treinamento terem sido apresentados (Haykin, 2001).

Na próxima seção são apresentados os conceitos relacionados ao algoritmo KNN.

3.5 K-Nearest Neighbor

Os métodos de aprendizado, como as RNA, constroem modelos e descrições gerais explícitas da função alvo, quando são fornecidos os dados de treinamento. No entanto, os métodos de aprendizagem baseada em instâncias simplesmente armazenam os dados de treinamento (Russell & Norvig, 2010). Deste modo, a cada nova instância a ser classificada, é calculada

sua relação com os exemplos previamente armazenadas, a fim de atribuir um valor de função destino.

O algoritmo KNN é o método mais básico de aprendizado baseado em instâncias, o qual assume que todos as exemplos correspondem a pontos em um plano n-dimensional R^n, onde n é o número de atributos utilizados para representar os exemplos. O KNN utiliza uma função distância para determinar o quão próximo uma instância está da outra (Mitchell, 1997). Embora seja um algoritmo simples, ele possui baixa taxa de erro (Cover & Hart, 1967).

Na Figura 3.15 é ilustrado este problema de classificação, onde temos a classe positiva e a classe negativa. O conjunto de exemplos de treinamento é descrito por dois atributos. Considerando o algoritmo KNN para classificação, com k = 1, o novo exemplo X_i seria classificado de acordo com o único vizinho mais próximo, que é da classe negativa.

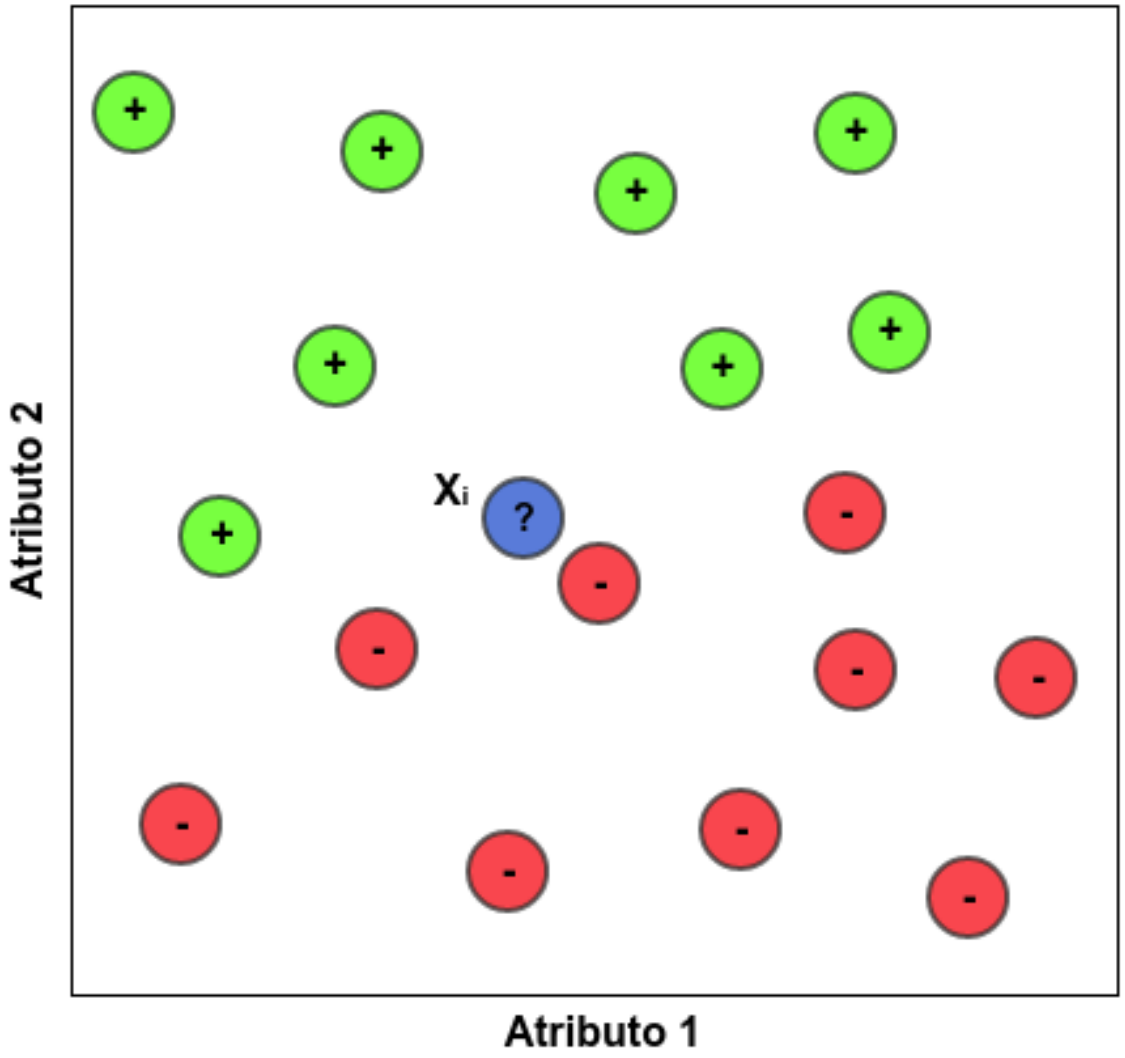


Figura 3.15: Exemplo de classificação do método KNN. Adaptado de (Ferrero, 2009).

Para classificar uma instância X_i, ainda não classificada, o algoritmo realiza as seguintes atividades:

1. É calculada a similaridade entre a instância X_i e cada uma das instâncias que foram previamente classificadas pelos especialistas que construíram a base de exemplos;

2. As k instâncias da base de exemplos mais similares a instância X_i são selecionadas (K vizinhos mais próximos);

3. A instância X_i é classificada em determinada categoria de acordo com algum critério de agrupamento dos k vizinhos mais próximos selecionados na etapa anterior.

O algoritmo KNN possui três pontos importantes que devem ser considerados: conjunto de exemplos de treinamento, medida de similaridade entre os exemplos e tamanho do k. Nas próximas subseções são descritos estes três (3) critérios.

3.5.1 Conjunto de Exemplos de Treinamento

A quantidade de exemplos de treinamento a serem armazenados pelo KNN possui influência no tempo de processamento e na qualidade da classificação, pois é necessário comparar as novas instâncias com todas as instâncias armazenadas na base de exemplo (Russell & Norvig, 2010). Como existem problemas que possuem domínios muito grandes, com muitos exemplos, o processo de classificação pode se tornar muito lento, a ponto de extrapolar o tempo máximo de resposta do problema (Russell & Norvig, 2010).

Para reduzir o custo de classificação de novos exemplos e o espaço de armazenamento em memória da base de exemplos, ao invés de armazenar todos as instâncias de treinamento na base de exemplos, normalmente opta-se por manter somente as instâncias mais representativas de cada classe, resumindo a informação mais importante em uma quantidade menor de instâncias. Várias pesquisas foram realizadas para selecionar instâncias mais representativas e reduzir a quantidade de amostras de treinamento (Aha et al., 1991)(Dasarathy, 1994)(Kuncheva & Bezdek, 1998)(Hattori & Takahashi, 1999).

3.5.2 Medida de Similaridades

Um ponto importante do algoritmo é o modo de medir a similaridade entre a nova instância a ser classificada em relação aos exemplos armazenados na base. Para tal proposito, existem diversas medidas, entre as quais estão as medidas de distância e de correlação (Ferrero, 2009).

Para calcular a similaridade, quando o conjunto de dados é descrito por atributos numéricos, são utilizadas medidas de distâncias, de modo que a menor distância corresponde a maior similaridade. Dentre as medidas comumente aplicadas, destaca-se a grande utilização da Distância Euclidiana apresentada na Equação 3.17 (Mitchell, 1997).

$$D = \sqrt{(p_1 - q_1)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{1}^{n}(p_i - q_i)^2} \tag{3.17}$$

onde, $p = (p_1, ..p_n)$ e $q = (q_1, ..q_n)$ são dois pontos n-dimensionais.

Para classificar uma instância x descrita pelo vetor de atributos:

$$[a_1(x), a_2(x), ..., a_n(x)] \tag{3.18}$$

onde, o a_k denota ao valor do atributo ${}_kth$ da instância x, e calcula-se a distância entre cada uma das instâncias da base de exemplos.

Para calcular a distância da instância x com uma instância y da base de exemplos, é utilizada a Distância Euclidiana apresentada na Equação 3.17. O atributo k da instância x é calculado com o atributo k da instância y. Então, a distância entre duas instâncias x e y é definida como $D(x, y)$, e apresentada na Equação 3.19 (Mitchell, 1997).

$$D(x, y) == \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2} \tag{3.19}$$

Portanto para N amostras com D atributos, esta abordagem escala como $O[DN^2]$.

A similaridade entre exemplos pode também ser determinada pelo coeficiente de correlação, levando em consideração o padrão dos valores dos atributos e não a magnitude (Ferrero, 2009).

A medida de similaridade possui influência no resultado da classificação. Um estudo mais aprofundado sobre a influência da métrica de similaridade no desempenho do KNN foi realizado por Short & Fukunaga (1981).

3.5.3 Tamanho do k

O algoritmo KNN classifica novas instâncias considerando a classe dos k exemplos mais próximos. Se $k = 1$, então é atribuída a essa nova instância a mesma classe do exemplo mais próximo segundo a medida de similaridade utilizada. Se $k > 1$, então são levadas em consideração as classes dos k exemplos mais próximos para realizar a classificação, sendo que neste caso, a abordagem mais comum é atribuir a nova instância a classe majoritária presente no conjunto dos k exemplos mais próximos.

A Figura 3.16 ilustra os dois casos de classificação citados anteriormente para a classificação de um exemplo x_i. Temos um conjunto de exemplos positivos $(+)$ e negativos $(-)$ descritos

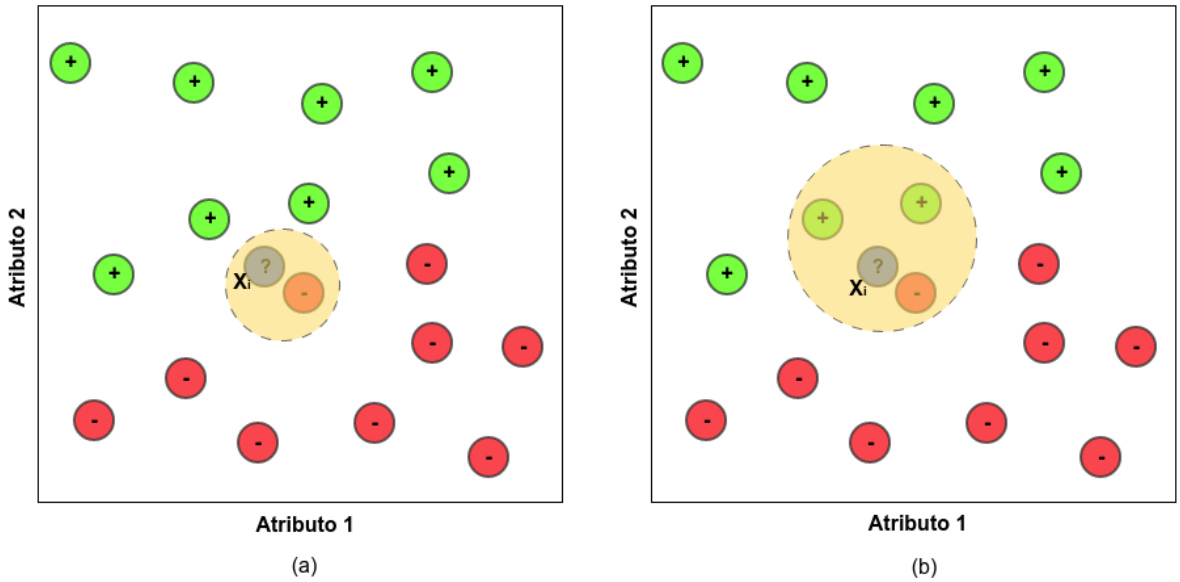


Figura 3.16: (a) Exemplo de 1NN. (b) Exemplo de 4NN. Adaptado de (Ferrero, 2009).

por dois atributos.

A Figura 3.16a apresenta o caso de $k = 1$, onde o exemplo x_i será classificado como negativo, devido ao fato, do exemplo mais próximo ser negativo. Por outro lado, é possível observar que o mesmo exemplo x_i terá uma classificação diferente na Figura 3.16b, isto porque, foi utilizado $k = 3$. Como a classe majoritária entre os três exemplos mais próximos é positiva, o exemplo x_i será classificado como positivo (Ferrero, 2009).

Conforme demonstrado, o número de vizinhos a ser considerado na classificação possui forte influência no resultado da classificação. No entanto não existe um valor de k ideal, apropriado para todos os problemas, e deste modo, esse valor deve ser avaliado e analisado para cada problema em particular. Na abordagem em que se rotula a nova instância com a classe majoritária entre os K exemplos mais próximos é recomendado utilizar k ímpar para evitar situações de empate (Ferrero, 2009).

3.6 Considerações Finais

Esse capítulo foi destinado a apresentar os conceitos de Inteligência Artificial, mineração de dados, com ênfase nos conceitos e características de Redes Neurais Artificiais e do algoritmo *K-Nearest Neighbor*.

No próximo capítulo são abordados alguns trabalhos científicos, considerando-se o estado da arte em detecção de intrusão utilizando métodos de inteligência artificial. Nas abordagens selecionadas destacam-se os métodos e recursos computacionais aplicados.

Capítulo 4

Trabalhos Relacionados

4.1 Considerações Iniciais

Neste capítulo apresentam-se trabalhos científicos, considerando-se o estado da arte em detecção de intrusão, com ênfase em métodos aplicando recursos de inteligência artificial e nos dados utilizados para avaliação.

4.2 Estado da Arte

Para a validação e a experimentação de métodos de detecção de intrusão, os pesquisadores utilizam bases de dados públicas, permitindo mensurar as taxas de detecção de intrusão e acurácia. Neste contexto as mais comumente aplicadas são a KDD CUP 99[1] e a DARPA[2]. Tais bases são constituídas por exemplos de ataques e atividades normais capturados em redes de computadores.

A base KDD CUP é composta por 42 atributos, dos quais 41 contém características de comportamentos da rede e um (1) denota o rótulo de classificação deste, determinando assim algum tipo de ataque ou comportamento normal (Tavallaee et al., 2009).

Estudos realizados nestas bases observaram que elas possuem algumas deficiências, como grande quantidade de amostras, exemplos redundantes e atributos excessivos (McHugh, 2000). Com o intuito de melhorar esta base, os pesquisadores Tavallaee et al. (2009) propuseram uma nova base denominada NSL-KDD[3]. Esta base vem sendo amplamente utilizada (Raman, Somu, Kirthivasan, Liscano & Sriram, 2017; Aburomman & Reaz, 2017; Ashfaq et al., 2017; Gunupudi et al., 2017; Osanaiye et al., 2016). A base NSL-KDD consiste em registros selecionados do conjunto completo de dados da KDD CUP 99, buscando sanar os problemas supra mencionados.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] https://ll.mit.edu/ideval/data/
[3] http://www.unb.ca/cic/datasets/nsl.html

Em 2007, os pesquisadores Boukerche et al. (2007) propuseram um método de detecção de intrusão baseado nos paradigmas do sistema imunológico humano e de agentes móveis. A arquitetura do modelo é baseada em *host* e paradigma de detecção por anomalia. A análise dos eventos foi realizada usando as ferramentas Syslog-ng[4] e Logcheck[5]. As tarefas de monitoramento, distribuição de carga de trabalho de detecção e armazenamento de informações foram realizadas pelos agentes móveis.

O poder de generalização das Redes Neurais Artificiais (RNA) facilita o reconhecimento de variações de um mesmo tipo de ataque, permitindo classificá-lo corretamente. Esta característica motivou vários trabalhos aplicando RNA para detectar intrusões. Dentro deste contexto, Vieira et al. (2010) propuseram um sistema que integra abordagens de análise por conhecimento e por comportamento, para detectar intrusões em ambiente de nuvem. Neste trabalho foi utilizada uma RNA *feed-forward* para a abordagem comportamental, permitindo identificar atividades legítimas e desvios do comportamento esperado. Já a abordagem de análise por conhecimento aplicou regras, também conhecidas como assinaturas de atividades maliciosas, onde realiza o monitoramento do fluxo de eventos para encontrar circunstâncias que se enquadrem em alguma das regras conhecidas. Neste método cada nó identifica eventos locais que podem ser violações e alertam os outros nós. Após os experimentos observou-se que a análise individual em cada nó reduz a complexidade e o volume de dados em relação a abordagens que concentram as análises em um único ponto.

Em 2010, Wang et al. (2010) propuseram uma abordagem chamada **FC-ANN**, utilizando RNA e clusterização *fuzzy* para detectar intrusões. Tal método utilizou o agrupamento *fuzzy* para gerar vários subconjuntos de treinamento, e diferentes modelos de RNA foram treinados com base nos subconjuntos, gerando diferentes modelos. Posteriormente o módulo de agregação *fuzzy* foi empregado parar concatenar os resultados. Os experimentos foram realizados sobre a base KDD CUP 99 e os resultados mostraram que a abordagem FC-ANN obteve uma taxa de acurácia de 96,71% e superou os métodos clássicos até então aplicados no que concerne ao nível de precisão de detecção, tais como os classificadores RNA que obteve 96,65% de acurácia e *Naive Bayes* (Russell & Norvig, 2010) que obteve 96,11% de acurácia.

Outra técnica para classificação utilizada no domínio de detecção de intrusão é o algoritmo *K-Nearest Neighbor* (KNN), já abordado na seção 3.5. Este algoritmo armazena todos os exemplos de treinamento e, quando um evento é submetido para ser classificado, os exemplos do conjunto de treinamento são recuperados e utilizados para classificar por similaridade o novo evento. A seguir são apresentados alguns trabalhos da literatura empregando este método de classificação para detecção de intrusão.

Em 2010, Tsai & Lin (2010) propuseram uma abordagem híbrida (TANN) para detecção de intrusão utilizando clusterização e o algoritmo KNN. Primeiramente aplicaram o algoritmo

[4]https://syslog-ng.org/
[5]http://logcheck.org/

k-MEANS (Mitchell, 1997) para obter os centros dos *cluster* correspondentes as classes de ataques. A partir deste passo, calculavam uma nova característica para cada exemplo, baseada na área triangular de cada exemplo em relação a pares de centros de *cluster*. Finalmente, o classificador KNN era aplicado para categorizar ataques semelhantes com base no novo atributo representado pela área triangular. Para a experimentação foi utilizada a base KDD CUP 99. De acordo com os autores Tsai & Lin (2010), os resultados demonstraram que a taxa de detecção de acurácia da abordagem TANN é de 99,01% e supera os métodos KNN e Support Vector Machine (SVM) (Mitchell, 1997) que obtiveram 93,87% e 94,98% de acurácia, respectivamente.

Os pesquisadores Govindarajan & Chandrasekaran (2011), em 2011, desenvolveram um método que consiste no emprego de duas técnicas de classificação tradicionais, uma RNA MultiLayer Perceptron (MLP) e uma RNA Radial Basis Function (RBF) (Haykin, 2001). Tal método possui características híbridas, de modo que cada saída do classificador recebe um peso, na escala (0-1), dependendo do desempenho de generalização. Quando ambos os classificadores produzirem a mesma saída, ela é aceita, e em caso de peso conflitante, o classificador com maior peso determina a saída. Os resultados obtidos neste trabalho mostraram que o método híbrido proposto teve uma uma precisão significativamente maior que a utilização das técnicas empregadas de maneira individual.

Em 2011, os autores (Panda et al., 2011) propuseram o uso de 10 (dez) diferentes abordagens de aprendizado de máquina para detectar intrusões. Para a avaliação experimental dos métodos foi utilizado a base de dados NSL-KDD.

Kim et al. (2014) propuseram uma abordagem para detecção de intrusão híbrida, a qual aplica um modelo de detecção por abuso e por anomalia, de modo hierárquico. Primeiramente o modelo de detecção por abuso foi construído com base no algoritmo de árvore de decisão C4.5 (Quinlan, 2014). Então os resultados das detecções de ataque do modelo baseado em abuso são considerados, e os eventos classificados como normais são decompostos em subconjuntos menores, e a partir disso, são criados vários modelos de SVM (Mitchell, 1997). Como resultado, cada modelo de detecção de anomalia não usa apenas a informação de ataque conhecida, mas também constrói os perfis do comportamento normal de forma muito precisa. De acordo com os autores, o experimento demonstrou que a abordagem híbrida proposta é melhor do que os métodos convencionais em termos de desempenho de detecção, tempo de treinamento e tempo de teste.

Assim como a abordagem TANN (Tsai & Lin, 2010), outros trabalhos aplicando KNN para detecção de intrusão juntamente com outras técnicas foram realizados. Os autores Lin et al. (2015) propuseram, em 2015, uma abordagem híbrida (CANN) para detecção de intrusão utilizando a técnica KNN conjuntamente com clusterização. Esta abordagem utiliza um valor baseado na soma de duas distâncias, para criar uma nova característica para cada exemplo. A primeira consiste na distância entre cada amostra e o centro de seu *cluster*, e a segunda, a distância entre cada amostra e o seu vizinho mais próximo no mesmo *cluster*. Então um classificador

KNN é aplicado para detectar os exemplos intrusivos com base nesta nova característica. Utilizando a base de dados KDD CUP 99 como conjunto de dados, os pesquisadores obtiveram taxas de detecção melhores ou semelhantes, e com um menor tempo de processamento em relação as abordagens com KNN individual e com SVM (Mitchell, 1997).

Os autores Singh et al. (2015) conceberam um método de detecção de intrusão baseada na rede neural Online Sequential Extreme Learning Machine (OS-ELM) (Liang et al., 2006). O trabalho utiliza três (3) técnicas para realizar a seleção de atributos: Filtros, Correlação e Consistência. Com o objetivo de reduzir o número de amostras a serem processadas durante a fase de treinamento, as conexões duplicadas e similares são agrupadas pelo algoritmo de clusterização *Density-based spatial clustering of applications with noise* (DBSCAN) (Ester et al., 1996). Os centros de cluster representem perfis de conexão. Por fim, a OS-ELM é treinada com os perfis de conexão resultantes do passo anterior. Os experimentos foram realizados com a base NSL-KDD e no tráfego real coletado da Universidade de Kyoto. A abordagem proposta obteve no experimento com a base NSL-KDD uma taxa de acurácia de 98,66%, superior as abordagens de RNA que obteve 94,89% de acurácia e da *Naive Bayes* (Russell & Norvig, 2010) que obteve 87,29% de acurácia.

Como os Sistemas de Detecção de Intrusão (IDS) trabalham com uma grande quantidade de dados, é de suma importância manter a qualidade dos recursos que representam todos os dados e remover características redundantes e irrelevantes. Neste contexto, se enquadram as técnicas de seleção de atributos, uma vez que a sua aplicação diminui a dimensionalidade dos dados, podendo também melhorar o desempenho do classificador. Os pesquisadores Eesa et al. (2015) propuseram uma abordagem de redução de atributos baseada no algoritmo *Cuttlefish* (Eesa et al., 2013). Tal trabalho aplicou Árvores de Decisão para o processo de classificação. Os resultados dos experimentos, por meio da base KDD CUP 99, demonstraram que o subconjunto reduzido pela abordagem *Cuttlefish* obteve uma melhor acurácia e taxa de detecção, em relação aos resultados obtidos com a base completa.

Com base nas abordagens TANN (Tsai & Lin, 2010) e CANN (Lin et al., 2015), Wang et al. (2016) propuseram um método de detecção de intrusão combinando características como densidade, centros de *cluster* e vizinhos mais próximos. Neste contexto, foi aplicada a mesma técnica que a abordagem CANN para criar uma nova característica para cada exemplo do conjunto de dados, e além disso, adicionaram uma outra característica baseada na densidade local de cada ponto de amostra. O classificador KNN foi aplicado para detectar os eventos com base nessas duas novas características. Os resultados experimentais, utilizando a base KDD CUP 99, mostraram uma melhor taxa de detecção em relação a abordagem CANN e ao algoritmo KNN. No entanto comparações com outros métodos clássicos de detecção de intrusão não foram realizadas

Em 2016, (Ma et al., 2016) propuseram um sistema (SCDNN) que combina Agrupamento Espectral (Clustering Spectral - SC) (Ng et al., 2002) e Redes Neurais Profundas (Deep Neural

Network - DNN) (Goodfellow et al., 2016). Para a avaliação do modelo proposto os autores executaram experimentos com as bases KDD CUP 99, NSL-KDD e um conjunto de dados de sensores de rede. Os resultados obtidos mostraram que a abordagem SCDNN obteve melhores taxas de detecção que RNA Multi-Layer Perceptron e SVM (Mitchell, 1997).

Ainda em 2016, os autores Osanaiye et al. (2016) propuseram um método para seleção de atributos multi-filtro baseado em conjuntos. A abordagem combina a saída de quatro métodos de seleção de filtros: Ganho de Informação (*Information Gain - IG*), Relação de Ganho (*Gain Ratio* - GR), chi-squared e ReliefF para obter uma solução ideal. Para a avaliação experimental, o método proposto foi aplicado na base de dados NSL-KDD utilizando o classificador de Árvore de Decisão J48, uma versão do algoritmo C4.5 (Quinlan, 2014). Os resultados demonstraram que a abordagem proposta pode efetivamente reduzir o número de atributos de 41 a 13 e possui alta taxa de detecção e precisão de classificação quando comparado a outras técnicas de classificação.

O trabalho desenvolvido por Chen et al. (2016) teve como foco o desenvolvimento de um classificador usando um sistema imunológico artificial (*Artificial Immune Systems* - AIS) (De Castro & Timmis, 2002) , combinado com uma variação de Algoritmos Genéticos chamada *Population-Based Incremental Learning* (PBIL) (Baluja, 1994) e filtragem colaborativa (*Collaborative filtering* - CF) para detecção de intrusão de rede. PBIL é um processo de aprendizagem que foi utilizado para melhorar o efeito de evolução do AIS para a criação de novos anticorpos. Para a avaliação experimental, foi utilizada a base KDD-CUPP 99 e uma base de dados da *Australia Credit Approval*. Os resultados obtidos são positivos em relação a outros métodos tradicionais como SVM (Mitchell, 1997), *Naive Bayes* (Russell & Norvig, 2010) e KNN.

Atualmente, o movimento Internet of Things (IoT) está se difundindo em todas as áreas que aplicam recursos computacionais. A introdução destes novos tipos de dispositivos pequenos e baratos, e que, além disso, são capazes de se conectar à Internet fez com os objetos usados no dia-a-dia, pudessem ser conectados a Internet e a dispositivos como computadores e *smartphones* (Tsai et al., 2014) (Miorandi et al., 2012) (Dhillon et al., 2017) (Ni et al., 2017). A ideia consiste em cada vez mais tornar o mundo físico e o digital um só, através da comunicação dos objetos com outros dispositivos, os *data centers* e suas nuvens. Os ambientes inteligentes estão se tornando reais e possíveis por meio da IoT, porém também não estão livres de ameaças de segurança e vulnerabilidade. O IoT torna possível ter ambientes domésticos inteligentes e estes podem ser bem comprometidos através da análise simples do tráfego de rede. Os autores Zodik (2015) discutem uma direção futura para a segurança e privacidade em soluções IoT. A pesquisa de Schurgot et al. (2015) concentra-se em considerações relacionadas a segurança para IoT.

Em 2017, cientes dos recursos limitados dos dispositivos utilizados nas aplicações de IoT e da complexidade computacional das tarefas de detecção de intrusão, os pesquisadores Gunupudi et al. (2017) propuseram um método em que uma função de pertinência *fuzzy* foi

projetada para reduzir a complexidade computacional e melhorar a precisão dos algoritmos classificadores. O método foi validado através de vários experimentos com as conhecidas bases NSL-KDD e DARPA, usando os classificadores KNN, J48 que consiste em uma versão do algoritmo C4.5 (Quinlan, 2014) e CANN (Lin et al., 2015). Os resultados alcançados revelaram uma melhora na precisão de detecção dos ataques U2R (User to Root) e R2L (Remote to Local).

Manzoor et al. (2017) realizaram experimentos de seleção de atributos com a base KDD CUP 99, com o intuito de reduzir a dimensionalidade da base sem promover perda de desempenho em relação a abordagem normal. Para este propósito, foram utilizadas duas abordagens para ranquear os atributos, a primeira baseada em ganho de informação formando o conjunto **GI**, e a segunda baseada em correlação, formando o conjunto **CR**. A escolha dos atributos para compor o modelo reduzido foi realizada através da união dos 10 (dez) primeiros atributos de **GI** com os 10 (dez) primeiros de **CR**, formando um conjunto final de 15 (quinze) atributos, sendo que cinco (5) atributos foram escolhidos pelas duas abordagens. Em seguida, outros 10 (dez) atributos foram adicionados ao conjunto final, oriundos da intersecção dos 11-30 de **GI** com os 11-30 de **CR**. Os demais atributos foram descartados por serem considerados inúteis. Portanto, ocorreu uma redução de 41 para 25 atributos. Com o objetivo de avaliar a eficácia do método proposto foram realizados experimentos com uma RNA *feed foward* com 10 neurônios na camada oculta. O método relatou um aumento da taxa de detecção e diminuição da taxa de alarmes falso em relação a base com 41 atributos.

O trabalho realizado por Ashfaq et al. (2017) definiu uma nova abordagem de aprendizagem semi-supervisionada baseada em *fuzzy*, utilizando amostras não categorizadas com algoritmo de aprendizado supervisionado para melhorar o desempenho do classificador. Uma RNA *feed foward* com uma (1) camada oculta, é treinada com o conjunto de treinamento original para produzir um vetor de associação difuso, as amostras não categorizadas são então categorizadas em três grupos (baixa, média e alta confusão). O classificador é treinado depois de incorporar cada grupo separadamente no conjunto de treinamento original. Os resultados experimentais obtidos aplicando o método proposto no conjunto de dados NSL-KDD mostram que amostras não categorizadas pertencentes a grupos de confusão baixos e altos fazem grandes contribuições para melhorar o desempenho do classificador.

As técnicas tradicionais propostas para detecção de intrusão, como RNA, possuem certa instabilidade na detecção de ataques menos frequentes. Com o objetivo de melhorar a detecção de ataques menos frequentes, os autores Raman, Somu, Kirthivasan & Sriram (2017) propuseram uma abordagem híbrida usando a propriedade *Helly* dos Hipergrafos e uma RNA Probabilística (Probabilistic Neural Network - PNN) (Specht, 1990) baseada em resíduos aritméticos. A propriedade *Helly* dos Hipergrafos foi aplicada para encontrar o subconjunto ótimo de atributos e os resíduos aritméticos do conjunto ótimo de atributos são utilizados para treinar a PNN. A abordagem foi avaliada através de experimentos com a base de dados KDD CUP 99, e os resultados demonstraram um melhor desempenho da abordagem proposta em relação aos classificadores existentes, em termos de detecção de ataques menos frequentes.

O método proposto pelos autores Hoque et al. (2017) consiste em uma abordagem de detecção de ataques distribuídos de negação de serviço (*Distributed Denial of Service* - DDoS), em tempo real, que usa uma nova medida de correlação para identificar ataques DDoS. A eficácia do método é avaliada com três (3) conjuntos de dados da rede. Foram utilizadas as bases de dados DARPA e duas outras bases específicas de ataques DDoS: CAIDA DDoS 2007[6] e TUIDS (Xuan et al., 2010). Além disso, o método proposto foi implementado em um *Field Programmable Gate Array* (FPGA) [7] para analisar seu desempenho. Os resultados dos experimentos mostram que o método produz alta precisão e devido a implementação de FPGA leva menos de um microssegundo para identificar um ataque.

Os sistemas tradicionais de detecção de intrusão são capazes de classificar apenas as intrusões previamente definidas no projeto do sistema. O trabalho de Leite & Girardi (2017) propõe a HyLAA, uma arquitetura de *software agent* que combina raciocínio baseado em casos (*Case Based Reasoning* - CBR) (Mitchell, 1997) e detecção por abuso (regras, assinaturas). Através do seu mecanismo de aprendizagem, essa abordagem é capaz de adaptar-se ao ambiente e identificar novas intrusões não especificadas no projeto do sistema. Isso é feito aprendendo novas regras para a detecção por abuso, através das soluções recorrentes do sistema CRB.

O trabalho realizado por Raman, Somu, Kirthivasan, Liscano & Sriram (2017) consiste em um método de detecção de intrusão adaptativa usando Algoritmos Genéticos baseada em Hipergrafos (HG-GA) para seleção de atributos, e configuração de parâmetros de uma SVM (Mitchell, 1997). A abordagem HG-GA SVM foi avaliada experimentalmente utilizando a base de dados NSL-KDD em dois cenários: (i) utilizando todos os atributos e (ii) utilizando apenas atributos selecionados pelo HG-GA. Os resultados dos experimentos demostram que a abordagem proposta tem um desempenho melhor que as técnicas existentes em relação a acurácia, taxa de detecção e taxa de falsos alarmes.

Em complemento aos trabalhos supramencionados, o presente trabalho integra uma linha de pesquisa em segurança computacional da qual são parceiros o Laboratório de Segurança Computacional (LaPSeC) da Universidade Estadual do Oeste do Paraná (UNIOESTE) e o laboratório DMC-NS (*Distributed Mobile Computing and Network Security*), situado no Departamento de Informática e Estatística da Universidade Federal de Santa Catarina (UFSC). A colaboração entre as instituições citadas acontece desde 2005 e desde então vem trazendo avanços na área de atuação, como uma proposta de abordagem utilizando sistemas imunológicos artificiais para a identificação de ataques à redes (Machado, 2005b) e um sistema de detecção de intrusão utilizando redes neurais artificiais (Lima, 2005)

O trabalho realizado por Machado (2005b) consiste em uma abordagem para detecção de intrusão inspirada nos conceitos, princípios e propriedades do sistema imunológico humano. A abstração computacional aplica a técnica de detecção baseada em anomalias e tem por base a

[6] http://www.caida.org/data/passive/ddos-20070804_dataset.xml

[7] FPGA é um circuito integrado projetado para ser configurado por um consumidor ou projetista após a fabricação.

monitoração dos registros de auditoria dos sistemas operacionais *Unix-like*. Sua arquitetura é baseada em *host* e distribuída. O processo de geração de eventos é realizado pelo *Syslog-ng*, a análise é responsabilidade da ferramenta *Logcheck* e a distribuição e persistência dos registros, assim como a incorporação de ações reativas e pró-ativas são implementadas por uma arquitetura baseada em agentes móveis. A aplicação da abordagem foi realizada em dois ambientes computacionais: uma empresa de computação e um provedor de internet. Os resultados indicaram uma redução significativa do número de registros reportados e analisados, como também facilitou a observação e análise eficaz das atividades dos *hosts* monitorados e possibilitou a implementação de respostas pró-ativas. Os trabalhos propostos por Bodnar (2013) e Costa (2014) reimplementaram o modelo em tecnologias mais atuais, mantendo as características originais do método. Em de tal modelo, umas das melhorias necessárias é a utilização de métodos de detecção com maior acurácia na caixa de análise, tais como técnicas de inteligência artificial (Boukerche et al., 2007).

O trabalho realizado por Lima (2005) caracteriza uma abordagem de detecção de intrusão, aplicando técnicas de inteligência artificial para a classificação de eventos em redes de computadores. Para tal propósito o autor utilizou monitoramento do fluxo de dados da rede, o qual é considerado subsídio para o uso de análise semântica e redes neurais artificiais. Os resultados gerados pelo processo de análise são valores numéricos, os quais classificam os eventos analisados em padrões considerados intrusivos ou normais. A eficiência desta classificação é modelada durante a fase de aquisição de conhecimento, denominada treinamento. O método de detecção aplicando RNA *multilayer perceptron* conseguiu chegar a uma precisão de 94,36%.

4.3 Considerações Finais

Os trabalhos analisados, sobre o estado da arte, apresentam vários métodos e abordagens que foram pesquisadas e propostas. Conforme pode ser observado, os métodos híbridos de detecção, criados pela combinação ou integração de múltiplas técnicas de classificação, alcançaram resultados mais satisfatórios que a utilização das técnicas de maneira individual (Kumar et al., 2010; Tombini et al., 2004; Guo et al., 2014).

No Próximo Capítulo será apresentado minuciosamente o método híbrido para detecção de intrusão, os materiais, as configurações experimentais e todas as definições técnicas propostas neste trabalho.

Capítulo 5

Materiais e Métodos

5.1 Considerações Iniciais

Neste capítulo serão apresentados os materiais e métodos aplicados neste trabalho, o qual consiste em um método híbrido para detecção de intrusão. Também será descrito em detalhes o modelo experimental para a avaliação do método proposto para detecção de intrusões.

Para a validação e experimentação do método proposto neste trabalho, o mesmo será comparado e avaliado estatisticamente em relação as abordagens individuais de Redes Neurais Artificiais (RNA) e K-*Nearest Neighbor* (KNN). Para subsidiar os experimentos, foi utilizada a base de dados pública NSL-KDD (Tavallaee et al., 2009), que consiste em uma melhoria do conjunto completo de dados da KDD CUP 99 (Blake & Merz, 1998).

5.2 Método Híbrido para Detecção de Intrusão

O método proposto neste trabalho foi motivado pelas observações de que os métodos híbridos, disponíveis na bibliografia (Capítulo 4), alcançaram maior êxito para o processo de detecção de intrusão (Kumar et al., 2010; Tombini et al., 2004; Guo et al., 2014).

Assim, lança-se a hipótese de que a utilização de um método híbrido composto por uma Redes Neurais Artificiais (RNA) e pelo algoritmo *K-Nearest Neighbors* (KNN), alcançará desempenho superior em relação às abordagens individuais destas técnicas.

A partir desse contexto, delineou-se o presente trabalho, o qual consiste em uma abordagem híbrida, aplicando RNA e o algoritmo KNN. A arquitetura do método é apresentada na Figura 5.1.

De acordo com a padronização CIDF (Seção 2.4.1), a abordagem híbrida proposta é classificada como um método da caixa de análise (A-Box). Além disso, em termos de classificação de IDS, a abordagem se enquadra como um método para IDS baseados em rede (Seção 2.4.2).

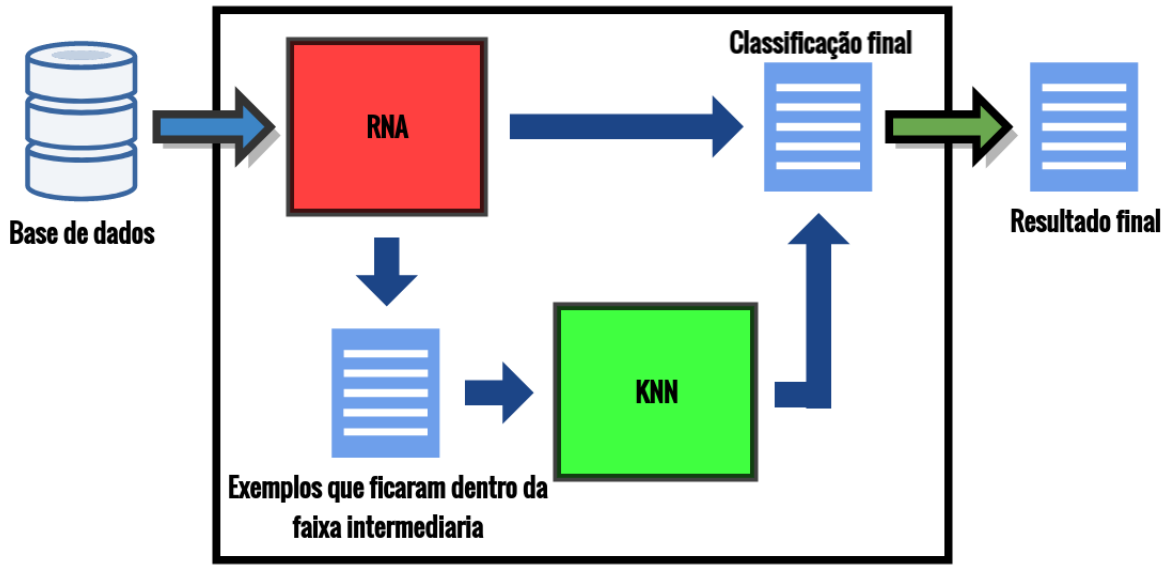


Figura 5.1: Ilustração da abordagem proposta. Fonte: O autor.

Na sequência serão apresentados os componentes arquiteturais da solução e o protocolo da abordagem proposta.

5.2.1 Rede Neural Artificial - RNA

Um dos métodos de classificação adotados no método proposto foi a RNA *Feedfoward* do tipo *Multilayer Perceptrons*, face a sua capacidade de resolver problemas não linearmente separáveis. Para o treinamento foi escolhido o algoritmo de retro-propagação de erro (*Backpropagation*) (Haykin, 2001). A função tangente hiperbólica foi utilizada como função de ativação da rede, pois determina resultados dentro do limiar de -1 a 1, que foi definido para a saída do modelo neural.

De acordo com as características de rede presentes em cada amostra da base de dados, e da necessidade de classificação em apenas duas categorias (intrusivo e não intrusivo), fixou-se a entrada da rede neural em 41 neurônios, referente aos 41 atributos da base de dados, e a saída em 1 neurônio.

Para a camada oculta, Lippmann (1987) afirma que a mesma deve ter $s(e+1)$ neurônios, onde s é o número de neurônios de saída e e o número de neurônios na entrada. Portanto utilizou-se esse cálculo para determinar a quantidade de neurônios na camada oculta, os quais atuam como extrator de características e se conectam ao neurônio de saída. Por fim, o neurônio de saída informa o valor final para cada amostra analisada.

5.2.2 K-Nearest Neighbor - KNN

O método proposto também utiliza o algoritmo KNN. O objetivo principal do algoritmo KNN é determinar a classe de uma instância a ser classificada, baseado nos exemplos presentes na base construída pelos especialistas do domínio. Este algoritmo considera os K vizinhos mais próximos. Inicialmente definimos um valor de $K = 1$. O valor de K foi escolhido baseado em resultados obtidos em testes anteriores (Rosa, 2017).

Além disso, para o calculo de similaridade da instância a ser classificada em relação as instâncias do conjunto de treinamento, utilizou-se a distância Euclidiana, cuja fórmula é apresentada na Seção 3.5.2.

5.2.3 Protocolo de Funcionamento do Método Proposto

A sequência de ações realizadas pelo método híbrido consiste em passar as instâncias a serem classificadas primeiramente pelo modelo neural, o qual gera uma saída entre -1 e 1 para cada exemplo submetido. Neste contexto de classificação, os exemplos mais próximos a -1 se assemelham ao comportamento não intrusivo. Já os exemplos mais próximos de 1 se assemelham ao comportamento intrusivo. Por fim, as instâncias que obtiveram valores de saída intermediários caracterizam situações em que o modelo neural não obteve clareza ou precisão. Nestes casos, inicialmente propôs-se a utilização dos valores entre -0.7 e 0.7 como os valores de fronteira dessa faixa, denominada neste trabalho como *faixa intermediária*.

Deste modo, a ideia da presente abordagem é obter o valor de saída do modelo neural para cada exemplo, e submeter os exemplos cujos valores de saída estão dentro da *faixa intermediária* à um segundo método de classificação, no caso, o algoritmo KNN, com o objetivo de ter uma classificação mais precisa desses exemplos.

Para realizar a classificação utilizando a abordagem híbrida, os exemplos são primeiramente submetidos a RNA, os exemplos que obtiveram valores fora dos limites da *faixa intermediária* são armazenados em um arquivo de classificação final. Os exemplos que obtém valores de saída dentro da *faixa intermediária* são submetidos ao algoritmo KNN. Por fim, após estes exemplos serem classificados pelo algoritmo KNN, os mesmos são salvos no arquivo de classificação final. A Figura 5.1 ilustra o processo de classificação de uma base de dados utilizando abordagem proposta.

Para ajustar os valores de fronteira da *faixa intermediária* foram realizados experimentos levando em consideração as faixas de inicio e fim dos falsos negativos e falsos positivos.

5.3 Local de Experimentação

Os experimentos foram realizados no Laboratório de Pesquisa em Segurança Computacional (LaPSeC), da Universidade Estadual do Oeste do Paraná (UNIOESTE), utilizando o computador de alto desempenho (HPC), disponibilizado pela Universidade Federal do Paraná (UFPR) através do Centro de Computação Científica e Software Livre (C3LS).

5.4 Materiais Aplicados

Os materiais utilizados na experimentação foram os seguintes:

- Notebook;
- Linguagem *R*[1] versão 3.3.2;
- Linguagem *Python*[2] versão 2.7.12;
- Biblioteca *Keras*[3];
- Biblioteca de inteligência de máquina *TensorFlow*[4];
- Biblioteca *Scikitlearn*[5] versão 0.18.1;
- Computador de alto desempenho HPC[6];

5.4.1 Configurações

- **Notebook:** Acer Aspire 4733, com processador *Intel Pentium Dual Core CPU T4500 @ 2.30GHz, memória RAM de 4 GB, HD de 500 GB*;
- **HPC**
  - Composição: 6 nodos de processamento, cada um com 4 sockets Intel Xeon E5-4627 v2 @ 3.30GHz (8 núcleos por socket);
  - Memória RAM: 256 GB de RAM.

[1] https://www.r-project.org/
[2] https://www.python.org/
[3] https://keras.io/
[4] https://www.tensorflow.org/
[5] http://scikit-learn.org
[6] https://www.c3sl.ufpr.br/c3hpc/

5.5 Método Híbrido de Detecção de Intrusão e Método Experimental

Para a realização do presente trabalho, primeiramente foram realizados estudos da literatura contemplando os seguintes temas: segurança de redes de computadores, tipos de ataques e técnicas de invasão computacionais, sistemas de detecção de intrusão e métodos específicos de Inteligência Artificial.

O objetivo da abordagem introduzida neste trabalho é realizar a detecção de ataques, mas não identificar qual é o tipo do ataque. Portanto a saída do método será apenas a classificação de cada exemplo em *intrusivo* ou *não intrusivo*.

O método aplicado para a avaliação da abordagem híbrida proposta e sua comparação com outros métodos, foi desenvolvido de acordo com as atividades e técnicas propostas no processo de *Knowledge Discovery in Databases* (KDD) abordado no Capítulo 3. Dentro deste contexto, passa-se a abordar as decisões de projeto realizadas em cada uma de suas fases, as quais são apresentadas nas próximas seções.

5.5.1 Seleção de Dados

Para a validação e experimentação de métodos de detecção de intrusão, os pesquisadores geralmente utilizam bases de dados públicas, permitindo mensurar as taxas de detecção de intrusão e acurácia. Neste contexto as mais comumente aplicadas são a KDD CUP 99 e a DARPA[7]. Tais bases são constituídas por exemplos de ataques e atividades normais capturados em redes de computadores (Ashfaq et al., 2017).

Cada exemplo da base KDD CUP 99 possui 42 atributos, dos quis 41 são características comportamentais da rede e são extraídos dos pacotes capturados durante a atividade da mesma. O último atributo é o que rotula a instância como algum tipo de ataque ou comportamento normal.

Os 41 atributos presentes na base KDD CUP 99 Data podem ser divididos em 4 categorias: atributos básicos, atributos de conteúdo, atributos de tráfego baseados em tempo e atributos de tráfego baseados em conexão.

A Tabela 5.1 apresenta os atributos que podem ser extraídos de um pacote de rede, sem considerar a carga útil do mesmo, ou seja, extraídos do cabeçalho do pacote. Tais atributos são denominados básicos.

A Tabela 5.2 apresenta os atributos extraídos da carga útil dos pacotes de rede, com o intuito de observar o comportamento suspeito dentro do segmento de carga útil, são chamados

[7]https://ll.mit.edu/ideval/data/

Tabela 5.1: Atributos básicos capturados de pacotes de rede. Fonte: (Ashfaq et al., 2017).

#	**Atributo**	**Descrição**	**Tipo**
1	duration	Duração da conexão	Contínuo
2	protocol_type	Tipo de protocolo (tcp, udp, etc)	Discreto
3	service	Tipo de serviço de destino (HTTP, Telnet)	Discreto
4	flag	Estado da conexão	Discreto
5	src_bytes	Números de bytes da origem ao destino	Contínuo
6	dst_bytes	Número de bytes do destino à origem	Contínuo
7	land	1 se o Host e a porta da origem e destino são os mesmos, 0 caso contrário	Discreto
8	wrong_fragment	Número de fragmentos errados	Contínuo
9	urgent	Número de Pacotes Urgentes	Contínuo

de atributos de conteúdo.

Tabela 5.2: Atributos de conteúdo capturados de pacotes de rede. Fonte: (Ashfaq et al., 2017).

#	**Atributo**	**Descrição**	**Tipo**
10	hot	Número de ações "hot"em uma conexão, como: inserir um diretório do sistema, criar e executar programas	Contínuo
11	num_failed_logins	Número de tentativas de login com falha	Contínuo
12	logged_in	1 se o login obteve sucesso e 0 caso contrário	Discreto
13	num_compromised	Número de condições comprometidas	Contínuo
14	root_shell	1 se o shell root é obtido 0 caso contrário	Discreto
15	su_attempted	1 se houver tentativa de conseguir “su root” 0 caso contrário	Discreto
16	num_root	Número de acessos como root	Contínuo
17	num_file_creations	Número de operações de criação de arquivos	Contínuo
18	num_shells	Número de shell prompts abertos	Contínuo
19	num_access_files	Número de operações a arquivos de controle de acesso	Contínuo
20	num_outbound_cmds	Números de comandos externos (sessão FTP)	Contínuo
21	is_hot_login	1 se o login pertence à lista “hot”, 0 caso contrário	Discreto
22	is_guest_login	1 se o login é do tipo “guest”, 0 caso contrário	Discreto

Os demais atributos correspondem a informações estatísticas relacionadas ao tráfego anterior e atual, e por essa razão são chamados de atributos de tráfego. Tais dados são divididos em dois grupos: atributos de tráfego baseados em tempo, que levam em consideração as últimas conexões em um intervalo de tempo; e os atributos de tráfego baseados em conexões, que consideram um número específico de últimas conexões.

Os atributos de tráfego levam em consideração dois pontos:

- atributos de *mesmo host*: são levadas em consideração apenas as sessões que possuem o

mesmo *host* que o pacote atual;

- atributos de *mesmo serviço*: são levadas em consideração apenas as conexões que possuem o mesmo serviço que o pacote atual.

Na Tabela 5.3 apresenta-se o grupo de atributos de tráfego baseados em tempo, levando em consideração os últimos dois (2) segundos.

Tabela 5.3: Atributos de tráfego baseados em tempo, no intervalo dos últimos dois segundos. Fonte: (Ashfaq et al., 2017).

#	Atributo	Descrição	Tipo
23	Count	Número de conexões para o mesmo host da conexão atual	Contínuo
24	srv_count	Número de conexões ao mesmo serviço da conexão atual	Contínuo
25	serror_rate	% de conexões que tiveram erros do tipo "SYN", dentre as conexões consideradas em count (23)	Contínuo
26	srv_serror_rate	% de conexões que tiveram erros "SYN", dentre as conexões consideradas em srv_count (24)	Contínuo
27	rerror_rate	% de conexões que tiveram erros do tipo "REJ", dentre as conexões consideradas em count (23)	Contínuo
28	srv_rerror_rate	% de conexões que tiveram erros "REJ", dentre as conexões consideradas em srv_count (24)	Contínuo
29	same_srv_rate	% de conexões ao mesmo serviço, dentre as conexões consideradas em count (23)	Contínuo
30	diff_srv_rate	% de conexões a diferentes serviços, dentre as conexões consideradas em count (23)	Contínuo
31	srv_diff_host_rate	% de conexões a diferentes hosts, dentre as conexões consideradas em srv_count (24)	Contínuo

No entanto, existem vários ataques de sondagem lenta que exploram, os *hosts* ou portas, usando um intervalo de tempo maior do que dois (2) segundos, por exemplo, um em cada minuto. Tais ataques não produzem padrões de intrusão com uma janela de tempo definido (Stolfo et al., 2000).

Para resolver este problema, os recursos "mesmo *host*" e "mesmo serviço" são recalculados, mas com base na janela de conexão de 100 conexões em vez de uma janela de tempo de dois (2) segundos. Tais atributos são apresentados na Tabela 5.4 e pertencem ao grupo de atributos de tráfego baseados em conexão, levando em consideração as últimas 100 conexões.

As instâncias da base estão classificadas em comportamento normal ou em ataque. A classificação de ataque pode ser de 22 tipos diferentes, listados na Tabela 5.5. Estes tipos de ataque podem ser agrupados em quatro (4) categorias principais: DoS, U2R, R2L e PROBE.

A Tabela 5.6 apresenta os valores que os atributos categóricos podem possuir. Os atributos como *land*, *logged_in*, *root_shell*, *su_attempted*, *is_host_login*, *is_guest_login* podem possuir

Tabela 5.4: Atributos de tráfego baseados em conexão. Fonte: (Ashfaq et al., 2017).

#	Atributo	Descrição	Tipo
32	dst_host_count	Número de conexões para o mesmo host da conexão atual dentre as últimas 100 conexões	Contínuo
33	dst_host_srv_count	Número de conexões ao mesmo serviço da conexão atual dentre as últimas 100 conexões	Contínuo
34	dst_host_same_srv_rate	% de conexões ao mesmo serviço, dentre as conexões consideradas em count (32)	Contínuo
35	dst_host_diff_srv_rate	% de conexões a diferentes serviços, dentre as conexões consideradas em count (32)	Contínuo
36	dst_host_same_src_port_rate	% de conexões a mesma porta de origem, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo
37	dst_host_srv_diff_host_rate	% de conexões a diferentes hosts, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo
38	dst_host_serror_rate	% de conexões que tiveram erros do tipo “SYN”, dentre as conexões consideradas em dst_host_count (32)	Contínuo
39	dst_host_srv_serror_rate	% de conexões que tiveram erros “SYN”, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo
40	dst_host_rerror_rate	% de conexões que tiveram erros do tipo “REJ”, dentre as conexões consideradas em dst_host_count (32)	Contínuo
41	dst_host_srv_rerror_rate	% de conexões que tiveram erros do tipo “REJ”, dentre as conexões consideradas em dst_host_srv_count (33)	Contínuo

dois valores, 0 ou 1. Outros recursos, como o tipo de protocolo, o serviço e a *flag*, possuem mais de dois (2) valores diferentes. O atributo *protocol_type* possui 3 valores distintos, o atributo *flag* possui 11 valores distintos e o atributo *service* possui 66 valores distintos.

Estudos realizados nestas bases observaram que elas possuem algumas deficiências, como grande amostras redundantes (McHugh, 2000). Com o intuito de resolver tais problemas, os pesquisadores Tavallaee et al. (2009) propuseram uma nova base denominada NSL-KDD[8].

Para a realização dos experimentos deste trabalho foi utilizada a base NSL-KDD. Ela foi construída com registros selecionados do conjunto completo de dados da KDD CUP 99, mantendo os mesmos atributos, e otimizando a questão de redundância de amostras. A base NSL-KDD é constituída de 125973 registros.

[8]http://www.unb.ca/cic/datasets/nsl.html

Tabela 5.5: Tipos de ataques. Fonte: (Ashfaq et al., 2017).

Deniel of service (DoS)	**User to root (U2R)**	**Remote to local (R2L)**	**Probing (PROBE)**
Back	Perl	FTP write	IP sweep
Ping of death	Buffer overflow	Guess password	NMAP
Neptune	Load module	IMAP	Port sweep
Smurf	Rootkit	Multi HOP	Satan
Land		Phf	
Teardrop		SPY	
		Wareclient	
		Warezmaster	

Tabela 5.6: Valores dos atributos categóricos. Fonte: (Ashfaq et al., 2017).

height **Atributos categóricos**	**Valores**	**#. de valores**
protocol_type	tcp, udp, icmp	3
service	smtp, ntp_u, shell, kshell, imap4, urh_i, netbios_ssn, ftp_u, mtp, uucp, nnsp, echo, tim_i, ssh, iso_tsap, time, netbios_ns, systat, hostnames, login, efs, supdup, http_8001, courier, ctf, finger, nntp, ftp_data, red_i, ldap, http, ftp, pm_dump, exec, klogin, auth, netbios_dgm, ...	66
flag	RSTR, S3, SF, RSTO, SH, OTH, S2, RSTOS0, S1, S0, REJ	11
land	0, 1	2
logged_in	0, 1	2
root_shell	0, 1	2
su_attempted	0, 1	2
is_host_login	0, 1	2
is_guest_login	0, 1	2

5.5.2 Pré-processamento

Conforme apresentado na seção 5.5.1, os exemplos das bases KDD CUP 99 e NSL-KDD são rotulados em 23 classes, sendo que 22 concernem a tipos de ataque e 1 a comportamento normal. O foco do presente trabalho é detectar comportamentos intrusivos e não intrusivos, não havendo uma preocupação em determinar a classe ou o ataque específico. Deste modo, a base foi modificada, de modo a rotular os exemplos apenas como intrusivos ou não intrusivos, onde as 22 classes de ataques passaram a compor uma única classe de amostras intrusivas. Tal procedimento é demonstrado na Figura 5.2.

O passo seguinte na etapa de pré-processamento foi a seleção de atributos. A atividade de seleção de atributos tem como objetivo selecionar um subconjunto de atributos para serem utilizados, buscando uma melhoria na taxa de classificação e no tempo de processamento (Hall, 1999).

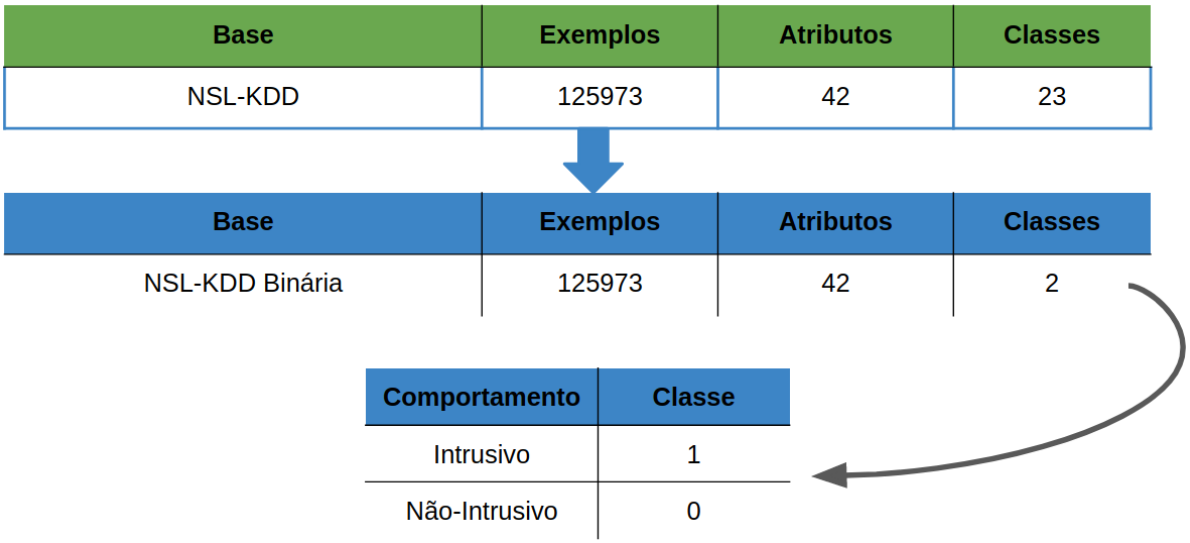


Figura 5.2: Transformação da base NSL-KDD. Fonte: O autor.

Os atributos podem ser selecionados de acordo com a sua relevância e qualidade para a tarefa de classificação. Uma das maneiras para se medir a qualidade de um atributo é avaliar o seu grau de associação com a classe, através da medida do ganho de informação.

Uma abordagem comumente utilizada para seleção de atributos é ilustrada na Figura 5.3, aplicando-se o algoritmo *Gain Ratio Attribute Evaluation* (Karegowda et al., 2010). Para um melhor entendimento do processo aplicado, pode-se definir a realização das seguintes etapas:

- Aplicação do algoritmo *Gain Ratio Attribute Evaluation* na base completa;
- O algoritmo retorna um valor para cada atributo da base, o qual corresponde a taxa de ganho de informação do atributo;
- Os atributos são então ranqueados de acordo com o valor da taxa de ganho de informação;
- A partir deste ranqueamento é realizada a seleção dos melhores atributos.

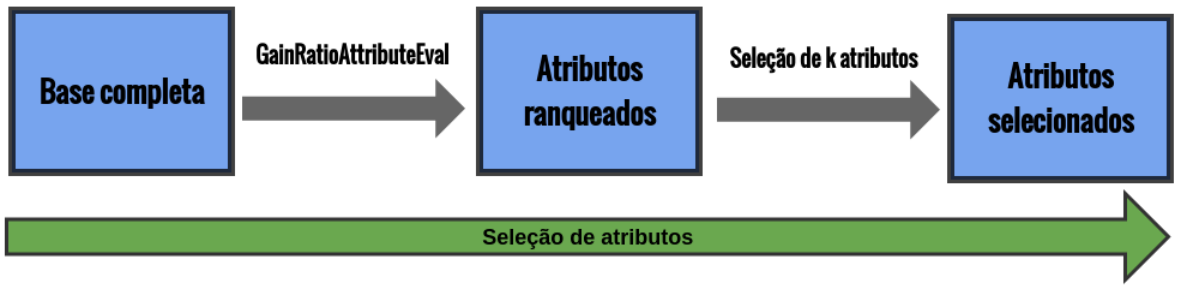


Figura 5.3: Diagrama da seleção de atributos. Fonte: O autor.

A Tabela 5.7 apresenta os atributos ranqueados de acordo com o valor da taxa de ganho de informação obtido por cada um deles após a aplicação do algoritmo *Gain Ratio Attribute Evaluation* na base NSL-KDD.

No processo de seleção de atributos foram formados três (3) subconjuntos, considerando o ranqueamento produzido pelo algoritmo *Gain Ratio Attribute Evaluation*, conforme ilustrado na Figura 5.4. As sub-bases possuem os 12, 20 e 30 atributos melhor classificados, respectivamente.

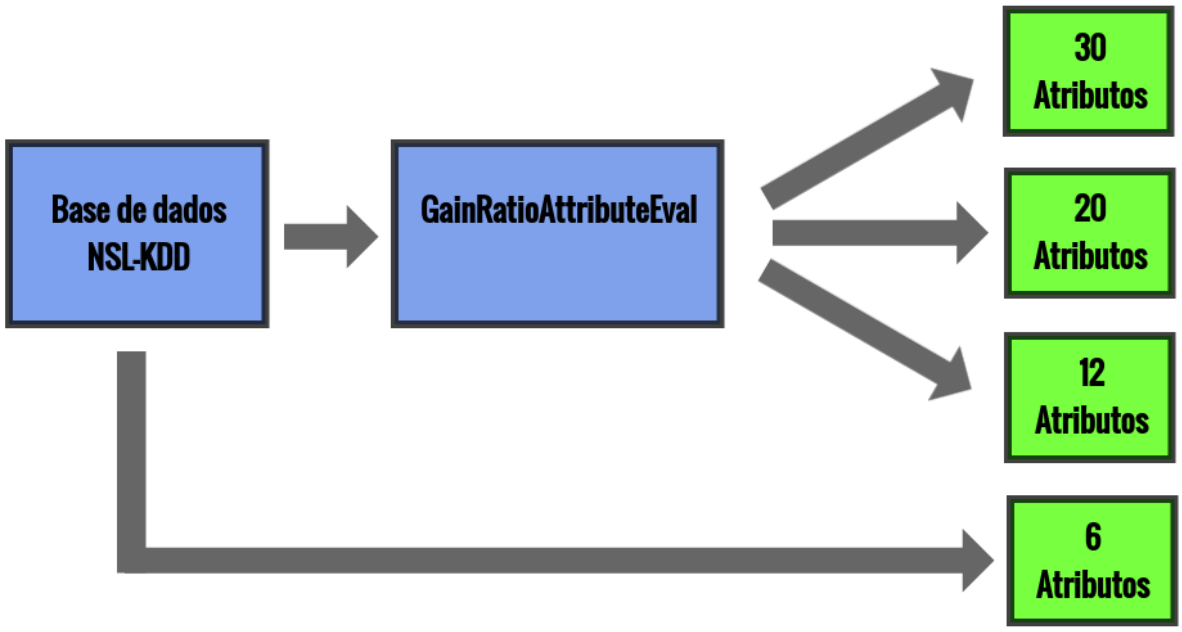


Figura 5.4: Sub-bases criadas através da seleção de atributos. Fonte: O autor.

Além dessas sub-bases previamente definidas, criou-se outra sub-base composta por atributos mais facilmente obtidos no processo de captura de pacotes em rede. Deste modo, a geração deste conjunto não foi definida com base no o algoritmo *Gain Ratio Attribute Evaluation*, e sim a partir da observação de ferramentas clássicas de captura de pacotes, entre as quais o *TCPDUMP*[9]. Dentre os atributos básicos apresentados na Tabela 5.1, foram selecionados seis (6), os quais são listados a seguir:

- *protocol_type*;
- *service*;
- *flag*;
- *src_bytes*;
- *dst_bytes*;
- *land*;

[9]http://www.tcpdump.org/

Tabela 5.7: Atributos ranqueados através de ganho de informação. Fonte: O autor.

Classificação	**Atributo**	**Ganho de Informação**
1	src_bytes	0.8162001
2	service	0.6715649
3	dst_bytes	0.6330489
4	flag	0.5193882
5	diff_srv_rate	0.518689
6	same_srv_rate	0.5098491
7	dst_host_srv_count	0.4759185
8	dst_host_same_srv_rate	0.4382138
9	dst_host_diff_srv_rate	0.4109107
10	dst_host_serror_rate	0.4059611
11	logged_in	0.4047515
12	dst_host_srv_serror_rate	0.3980669
13	serror_rate	0.39274
14	count	0.3835886
15	srv_serror_rate	0.3791249
16	dst_host_srv_diff_host_rate	0.2708369
17	dst_host_count	0.1980381
18	dst_host_same_src_port_rate	0.1888756
19	srv_diff_host_rate	0.1415535
20	srv_count	0.0942455
21	dst_host_srv_rerror_rate	0.0882668
22	protocol_type	0.0626391
23	rerror_rate	0.0567407
24	dst_host_rerror_rate	0.0521774
25	srv_rerror_rate	0.0515668
26	duration	0.0367247
27	hot	0.0115545
28	wrong_fragment	0.00961
29	num_compromised	0.0065041
30	num_root	0.003717
31	num_access_files	0.0021548
32	is_guest_login	0.0011684
33	num_file_creations	0.0009183
34	su_attempted	0.000514
35	root_shell	0.0003241
36	num_shells	0.0001043
37	num_failed_logins	0.0000604
38	land	0.0000381
39	urgent	0
40	num_outbound_cmds	0
41	is_host_login	0

5.5.3 Transformação dos Dados

A base NSL-KDD (Tavallaee et al., 2009) é uma base de dados pública já consolidada e formatada no padrão necessário para ser interpretada pelos métodos de análise. Desse modo não foi necessário nenhuma atividade de formatação dos dados.

5.5.4 Processamento

Nesta fase do processo KDD foram executados os experimentos, considerando os métodos de detecção abordados (Seção 5.2) no trabalho e o conjunto de dados selecionado nas fases anteriores (Seção 5.5.1).

Para a avaliação dos métodos de detecção utilizando uma base de dados pública NLS-KDD, foi escolhida a técnica de amostragem *cross-validation*, para realizar a avaliação da acurácia dos modelo de detecção.

Técnica de Amostragem

Como método de amostragem foi aplicado o *cross-validadion* utilizando 10 *folds*. Deste modo, a base é dividida em 10 subconjuntos de modo aleatório. A cada iteração são usados nove conjuntos para treinar e um para testar. Este processo é realizado por 10 iterações alternando o subconjunto de teste, conforme pode ser observado na Figura 5.5.

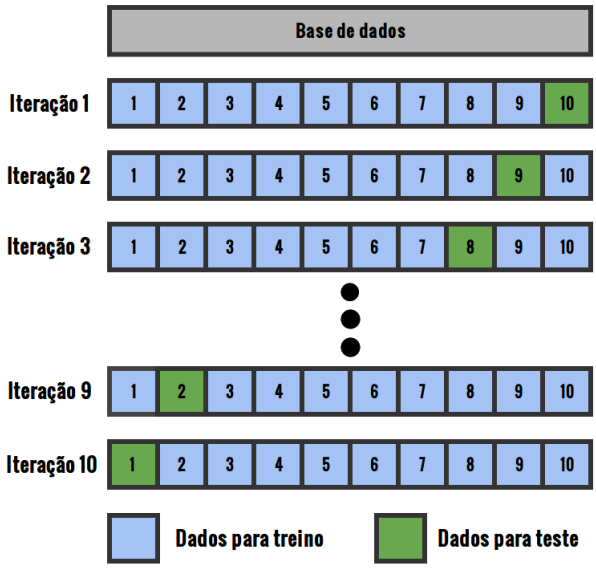


Figura 5.5: Representação do processo de *cross-validation* com 10 folds. Fonte: O autor.

Em relação ao processo ilustrado na Figura 5.6 é importante salientar que, inicialmente, o conjunto de instâncias da base de dados é dividida, aleatoriamente, em 10 subconjuntos de

mesmo tamanho. O conjunto de dados original da base NLS-KDD é composta por 125973 instâncias, e após essa etapa, cada um dos subconjuntos ficou com 12597 exemplos.

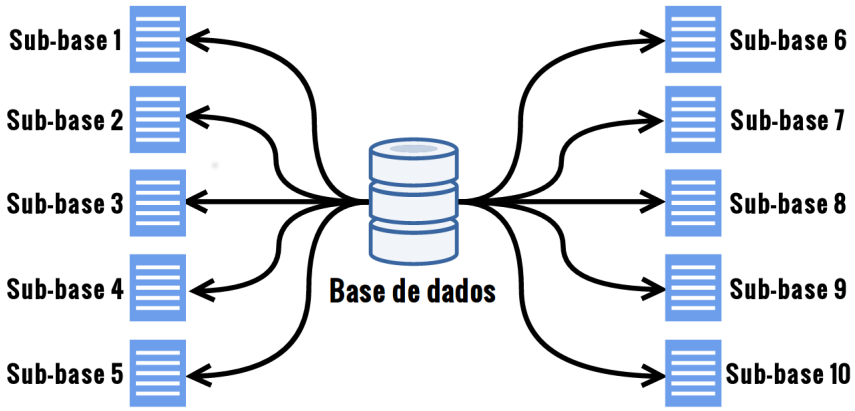
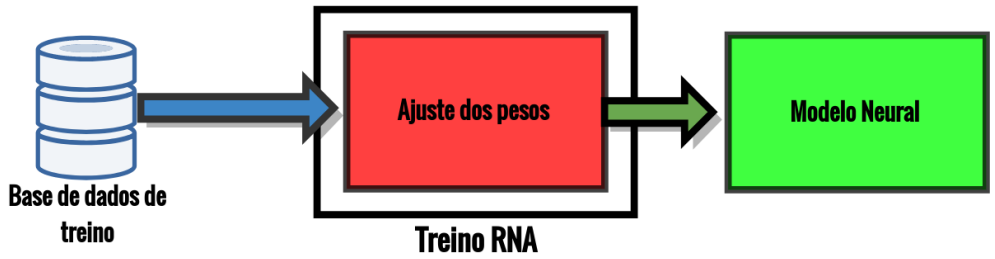


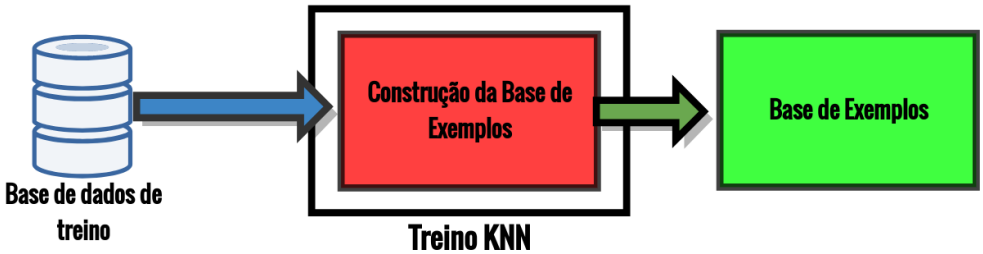
Figura 5.6: Processo de criação das 10 sub-bases de dados. Fonte: O autor.

Para cada uma das 10 iterações do processo de *cross-validation* são utilizados nove subconjuntos de dados para treinar os métodos de classificação. Os nove subconjuntos são aplicados para a fase de treino da RNA, auxiliando na calibração deste modelo, o qual atualiza os pesos das conexões sinápticas de acordo com tais amostras (Figura 5.7b).

Estes mesmos nove subconjuntos, aplicados no treinamento da RNA, são utilizados para construir a base de exemplos do método de detecção KNN (Figura 5.7b).



(a)



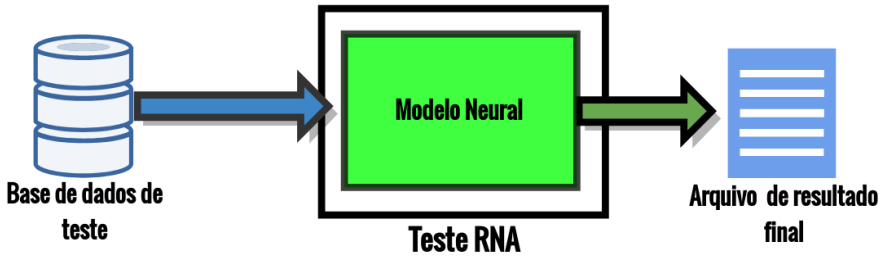
(b)

Figura 5.7: (a) Processo de treino do modelo neural. (b) Processo de criação da base de exemplos do algoritmo KNN. Fonte: O autor.

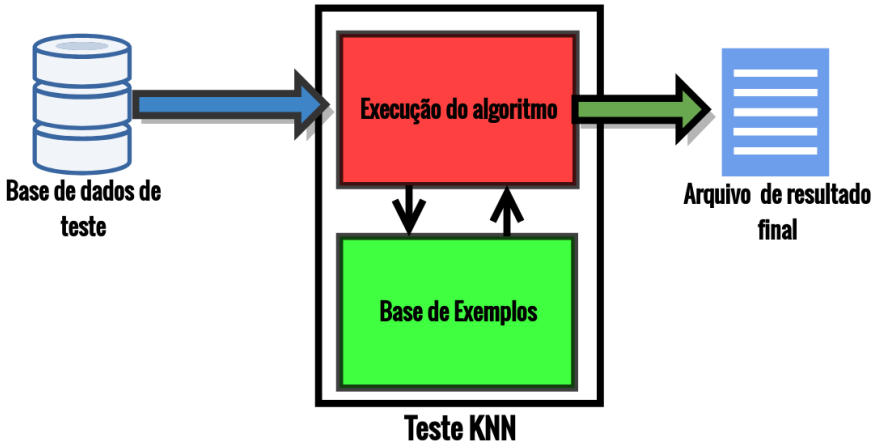
Após a construção dos modelos, aplica-se o subconjunto de teste, conforme apresentado na Figura 5.8. A parte (a) da referida Figura permite observar que o subconjunto não aplicado

no treinamento é utilizado para testar o modelo neural. A partir deste processo é possível avaliar o modelo neural gerado em relação a sua capacidade de predição.

Na parte (b) da Figura 5.8 ilustra-se o processo de teste do método KNN, onde aplica-se o mesmo subconjunto de teste utilizado para a avaliação da RNA. Nesta etapa do KNN, todas as instâncias do subconjunto de teste são comparadas com todos os exemplos da base KNN, realizando a classificação por similaridade. Deste modo, avalia-se também a capacidade preditiva do método KNN.



(a)



(b)

Figura 5.8: (a) Processo de teste do modelo neural. (b) Processo de teste do algoritmo KNN. Fonte: O autor.

O resultado do processo de *cross-validation* consiste em uma média dos resultados das execuções dos 10 *folds*.

Bases de Dados Utilizadas nos Experimentos

Para o processo de experimentação foram utilizadas as seguintes bases:

Tabela 5.8: Bases utilizadas nos experimentos. Fonte: O autor.

Bases	**Descrição**
NSL-KDD completa	Base pública NSL-KDD completa
NSL-KDD com 30 atributos	Sub-base derivada da base NSL-KDD com 30 atributos, definida a partir do processo de seleção de atributos (Seção 5.5.2)
NSL-KDD com 20 atributos	Sub-base derivada da base NSL-KDD com 20 atributos, definida a partir do processo de seleção de atributos (Seção 5.5.2)
NSL-KDD com 12 atributos	Sub-base derivada da base NSL-KDD com 12 atributos, definida a partir do processo de seleção de atributos (Seção 5.5.2)
NSL-KDD com 6 atributos	Sub-base derivada da base NSL-KDD com 6 atributos, definida a partir do processo de seleção de atributos (Seção 5.5.2)

Etapas do Processo de Experimentação

O processo de experimentação foi dividido em duas etapas:

- Na primeira realizou-se experimentos para encontrar as melhores *faixas intermediárias* do método híbrido, considerando cada umas das sub-bases definidas;
- Posteriormente foram executados experimentos para comparar o método híbrido proposto neste trabalho, com as técnicas RNA e KDD aplicadas isoladamente.

1ª Etapa: Experimentos para Definir Valor de Fronteira da Faixa Intermediária

A abordagem híbrida proposta no presente trabalho possui uma *faixa intermediária*, a qual delimita quais classificações são aceitas a partir do modelo neural, e quais instâncias são submetidas a um segundo método de classificação, que neste caso é o KNN. Os detalhes em relação ao protocolo experimental foram definidos na Seção 5.2.3.

Neste contexto, o objetivo da primeira classe de experimentos foi encontrar uma *faixa intermediária* que englobe a maioria dos falsos positivos e falsos negativos gerados pela RNA. Para um melhor entendimento, é importante ressaltar que a RNA gera uma saída entre -1 e 1 para cada amostra. Neste contexto, as instâncias que geram resultados mais próximos a -1 se assemelham a comportamento não intrusivo, e os exemplos com saídas mais próximas a 1 aproximam de comportamento intrusivo.

Deste modo, para a definição da *faixa intermediária* foi utilizado o conhecimento gerado durante o processo de treinamento. Após o treino, obtém-se o valor de saída do neurônio para cada instância, sendo estes utilizados para separar os conjuntos em resultados positivos e negativos. Cada um destes conjuntos é ordenado de modo crescente. A partir disso foram definidas as *faixas intermediárias*, seguindo os critérios referenciados na Tabela 5.9.

Tabela 5.9: Informações do método experimental para definir os valores de fronteira da *faixa intermediária*. Fonte: O autor.

Abordagens	**Método de detecção**	**Faixa Intermediária**
P1-N75	Híbrido	1º valor positivo a 75º percentil negativo
P1-N50	Híbrido	1º valor positivo a 50º percentil negativo
P1-N25	Híbrido	1º valor positivo a 25º percentil negativo
P1-N1	Híbrido	1º valor negativo a 1º valor negativo
P25-N75	Híbrido	25º percentil positivo a 75º percentil negativo
P25-N50	Híbrido	25º percentil positivo a 50º percentil negativo
P25-N25	Híbrido	25º percentil positivo a 25º percentil negativo
P25-N1	Híbrido	25º percentil positivo a 1º valor negativo

Para um melhor entendimento da nomenclatura aplicada na Tabela 5.9, P indica positivo e N negativo. A título de exemplo, a abordagem P1-N75 significa que os limites da *faixa intermediária* são:

- Limite superior: primeiro valor do conjunto positivo;
- Limite inferior: 75º percentil do conjunto negativo.

Na Figura 5.9 ilustra-se os limites superiores e inferiores das abordagens propostas na Tabela 5.9.

As abordagens para estabelecimento dos limites da *faixa intermediária*, apresentadas por meio da Tabela 5.9 e da Figura 5.9, foram avaliadas nas cinco bases definidas na Tabela 5.8.

Após a execução dos experimentos, as médias das taxas de detecção do processo de *cross-validation* obtidas por cada abordagem (Tabela 5.9), foram avaliadas para encontrar as melhores abordagens para a determinação das *faixas intermediárias* em cada uma das cinco bases de dados definidas na Tabela 5.8.

Posteriormente, estas abordagens selecionadas foram utilizadas na Etapa 2, que será descrita a seguir.

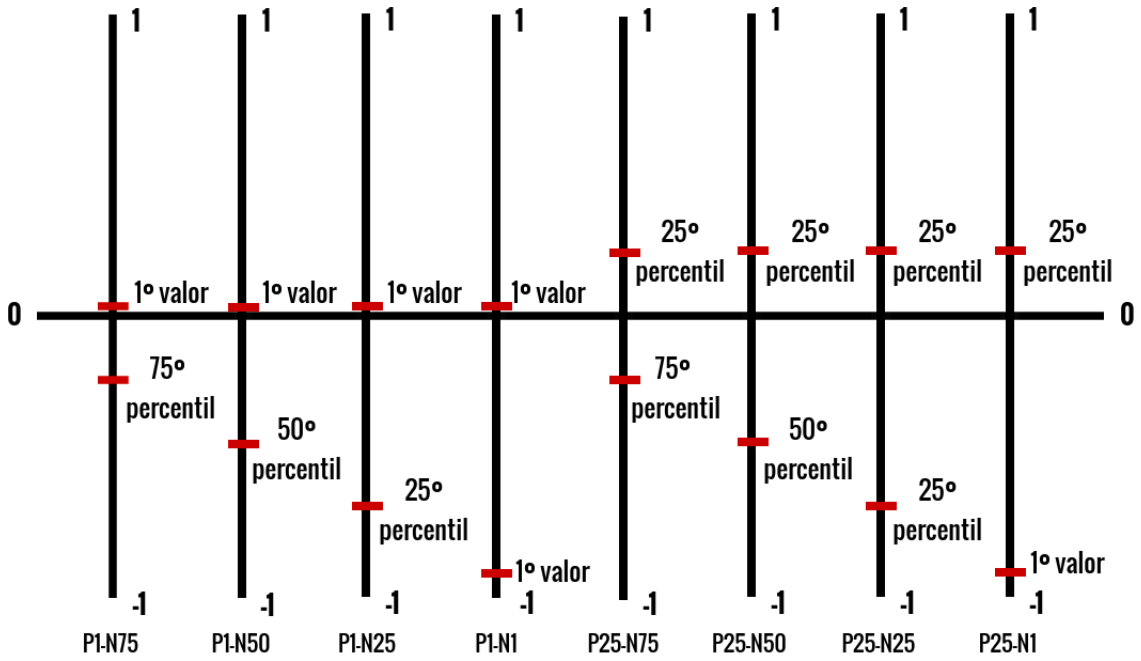


Figura 5.9: Representação dos valores para o cálculo da *faixa intermediária*. Fonte: O autor.

2ª Etapa: Experimentos entre os Métodos

Esta etapa de experimentos foi definida no intuito de avaliar comparativamente o método híbrido de detecção de intrusão, proposto neste trabalho, em relação aos métodos individuais baseados em RNA e KNN. Para tal propósito foram utilizadas as mesmas bases de dados definidas na Tabela 5.8.

A Figura 5.10 ilustra uma "caixa preta"criada para melhor representação do método experimental, contendo as técnicas de classificação RNA, KNN e Híbrida. Conforme demonstrado, pode-se aplicar variações do método Híbrido, tendo em vista as abordagens que forem selecionadas na etapa anterior para delimitar as *faixas intermediárias*.

Na caixa preta, primeiramente é realizado a divisão da base de dados de modo aleatório em 10 sub-bases de mesmo tamanho, de acordo com o processo de *cross-validation* abordado na Seção 5.5.4. Após esta divisão, as 10 sub-bases são utilizadas na execução do *cross-validation* para cada uma das abordagens de detecção (RNA, KNN e Híbridas [1..N]).

Na execução de cada iteração do processo *cross-validation*, os modelos são treinados e testados. Por fim, os resultado de cada abordagem de detecção consiste na média dos resultados das 10 iterações do processo *cross-validation*.

Conforme demonstrado na Figura 5.11, o protocolo experimental descrito anteriormente (Figura 5.10) é repetido para cada uma das bases definidas na Tabela 5.8.

Com o intuito de melhorar a confiabilidade dos resultados, aplicou-se um espaço amostral de 10 execuções do método experimental, conforme observa-se na Figura 5.12. Deste modo,

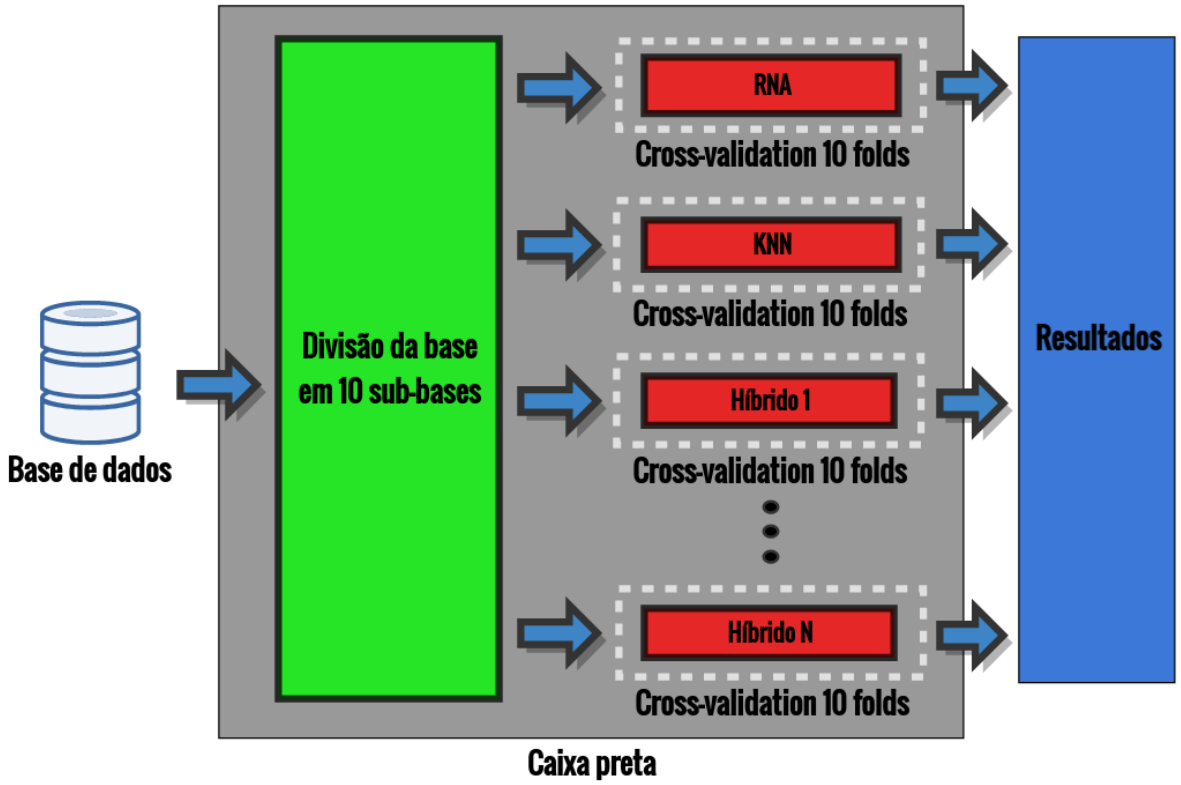


Figura 5.10: Representação da caixa do método experimental. Fonte: O autor.

os experimentos realizados em cada uma das bases (Figura 5.11) foi repetido 10 vezes, e os resultados foram avaliados na etapa de Pós-processamento do processo *KDD*.

Para a implementação dos métodos descritos nesta Seção foram utilizadas as seguintes tecnologias:

- Linguagem de programação Pyton;
- Biblioteca Keras;
- Biblioteca Tensorflow;
- Biblioteca Scikitlearn.

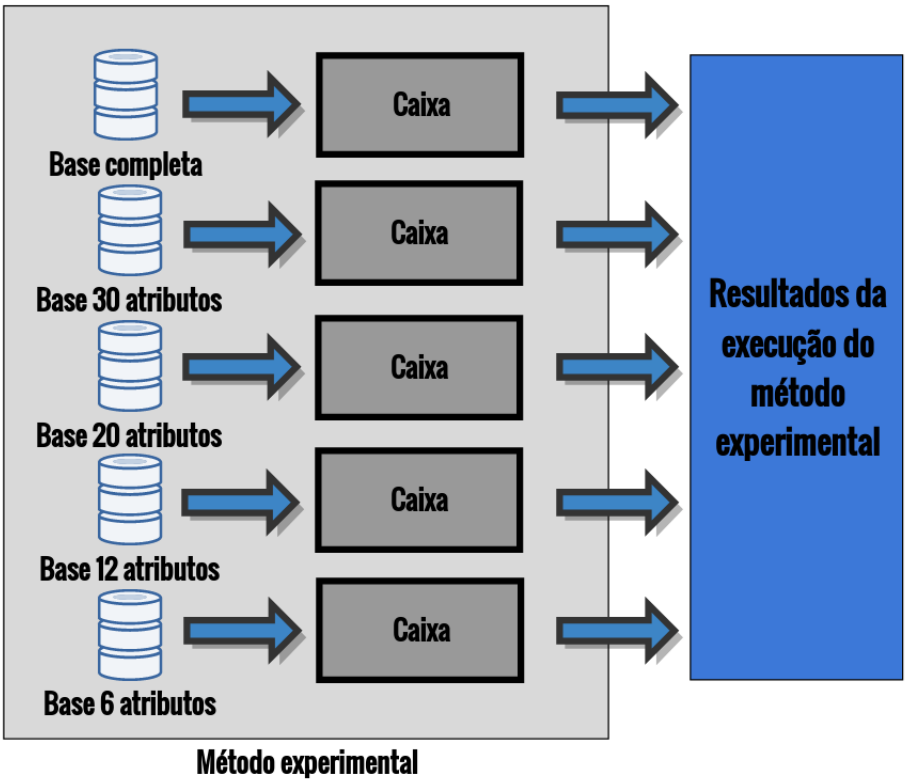


Figura 5.11: Representação do método experimental. Fonte: O autor.

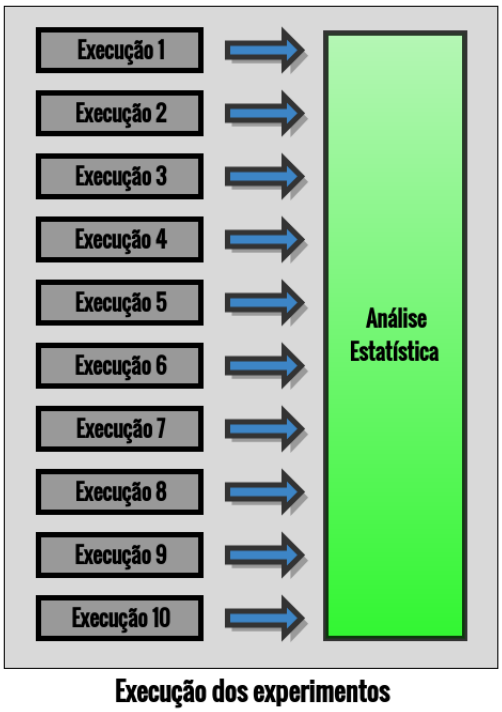


Figura 5.12: Representação do método completo com 10 execuções. Fonte: O autor.

5.5.5 Pós-processamento

Nesta etapa do *KDD*, os resultados dos experimentos foram avaliados, a fim de comparar o método híbrido, proposto neste trabalho, em relação as técnicas RNA e KNN utilizadas de modo individual. Para este propósito, foram considerados os seguintes critérios de classificação:

- **Falsos Negativos (FN):** Instâncias classificadas como normais pelo método de detecção, e que são intrusões;
- **Falsos Positivos (FP):** Exemplos não intrusivos, e classificados como tal pela técnica de detecção de intrusão;
- **Verdadeiros Negativos (VN):** Essa categoria abrange instâncias não intrusivas, e corretamente classificados pelo método de detecção;
- **Verdadeiros Positivos (VP):** Categoria que inclui exemplos corretamente reportados como intrusos pela técnica de deteção de intrusão.

A partir desta definição e dos resultados dos experimentos, foram calculadas alguns indicadores, tais como:

- Taxa de acurácia: Porcentagem de instâncias classificadas corretamente;
- Taxa de erro: Porcentagem de instâncias que foram classificadas incorretamente;
- Sensibilidade: Consiste no número de exemplos classificados como ataques dentre todos os exemplos intrusivos;
- Especificidade: Quantidade de instâncias classificadas como situação não intrusiva, dentre todos os exemplos não intrusivos.

A taxa de acurácia é definida na Equação 5.1.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{5.1}$$

A taxa de erro é calculada conforme a Equação 5.2.

$$Taxa\ de\ erro = \frac{FP + FN}{VP + VN + FP + FN} \tag{5.2}$$

A sensibilidade é calculada conforme a Equação 5.3.

$$Sensibilidade = \frac{VP}{VP + FN} \tag{5.3}$$

A especifidade é determinada de acordo com a fórmula apresentada na Equação 5.4.

$$Especificidade = \frac{VN}{VN + FP} \tag{5.4}$$

Na etapa de pós-processamento também foi realizada a avaliação estatística, no intuito de comparar os métodos Híbrido, RNA e KNN. Tal comparação entre os grupos foi realizada tendo em vista a taxa de acurácia.

Primeiramente, para definir o método estatístico a utilizar, foi realizado o teste de normalidade dos dados, utilizando-se o método Shapiro-Wilk (Shapiro & Wilk, 1965). De acordo com o método, utiliza-se a seguinte formulação de hipótese:

- H_0: A amostra possui uma distribuição normal.
- H_1: A amostra não possui uma distribuição normal.

Neste trabalho foi utilizado um nível de significância do teste de 95%. Portanto caso o valor $p - valor$ obtido através do método Shapiro-Wilk seja:

- $p - valor > 0,05$: H_0 é aceito, e podemos afirmar com nível de significância de 95% que a amostra provém de uma população normal.
- $p - valor <= 0,05$: H_0 é rejeitado, e podemos afirmar com nível de significância de 95% que a amostra não provém de uma população normal.

Após a verificação da normalidade dos dados, foi necessário a aplicação de outros testes estatísticos para verificar a existência de diferenças significativas entre as amostras.

Os dados utilizados na avaliação foram as taxas de acurácia obtidas através do processo de *cross-validation*, caracterizando dados numéricos, não pareados e não dependentes. Dadas tais características, são possíveis dois métodos para a avaliação:

- Análise de Variância (ANOVA) (Fisher, 1992): Teste paramétrico utilizado para os grupos que sejam independentes, caso as amostras sigam uma distribuição normal;
- Teste de Kruskal-Wallis (Kruskal & Wallis, 1952): Aplicável caso os resultados não caracterizem uma distribuição normal.

Assim como no testes estatísticos de normalidade, os testes ANOVA e Kruskal-Wallis são testes de hipóteses, e considera-se que:

- H_0: As médias de acurácia das abordagens são iguais;
- H_1: Existe pelo menos uma das médias de acurácia que é diferente.

Para a comparação das médias dos grupos também foi utilizado o nível de significância do teste de 95%. Portanto caso o valor $p - valor$ obtido através do método de comparação utilizado seja:

- $p - valor > 0,05$: H_0 é aceito, concluindo-se que com nível de significância de 95% que os grupos não são significativamente diferentes;
- $p - valor <= 0,05$: H_0 é rejeitado, e pode-se afirmar com nível de significância de 95% que ao menos um dos grupos é diferente dos demais.

Quando rejeitamos a hipótese nula H_0 no teste ANOVA ou no Kruskal-Wallis, ao menos um dos grupos é diferente dos demais. Porém, não tem como saber quais são diferentes. Assim, recomenda-se a aplicação do pós-teste Dunn (Everitt et al., 2001) para verificar onde estão as diferenças.

Na etapa de pós-processamento os resultados dos experimentos foram avaliados estatisticamente. Para este propósito foi utilizado o ambiente matemático e gráfico R[10], juntamente com a linguagem R[11].

5.6 Considerações Finais

Nesse capítulo foram apresentadas a abordagem híbrida para detecção de intrusão, proposta neste trabalho, as bases de dados utilizadas para a avaliação experimental, o processo de seleção de atributos, bem como o processo experimental seguindo as etapas do *KDD*.

No próximo capítulo serão apresentados e discutidos os resultados alcançados neste trabalho.

[10]https://www.rstudio.com/
[11]https://www.r-project.org/

Capítulo 6

Resultados e Discussões

6.1 Considerações Iniciais

No Capítulo 5 foram definidas a abordagem proposta e a arquitetura da Rede Neural Artificial (RNA) utilizada, além da configuração do algoritmo K-Nearest Neighbor (KNN).

Com o objetivo de avaliar o modelo proposto, vários experimentos foram realizados com a abordagem proposta e com abordagens individuais de RNA e KNN. Deste modo, os resultados experimentais demonstram as vantagens e diferenciais da abordagem proposta neste trabalho.

Neste capítulo serão apresentados e discutidos os resultados dos experimentos realizados. A discussão e os resultados foram organizados de modo a retratar cada uma das etapas do *Knowledge Discovery in Databases* (KDD) (Fayyad, 1996), de modo similar a organização apresentada no Capítulo 5.

6.2 Seleção de Dados

Os experimentos deste trabalho foram realizados utilizando a base NSL-KKD (Tavallaee et al., 2009), por ser uma das bases atuais mais consolidadas e já tendo sido utilizada em vários outros trabalhos (Raman, Somu, Kirthivasan, Liscano & Sriram, 2017; Aburomman & Reaz, 2017; Ashfaq et al., 2017; Gunupudi et al., 2017; Osanaiye et al., 2016), para avaliar métodos de detecção de intrusão. A NSL-KKD consiste em uma melhoria da base de dados KDD CUP 99 (Blake & Merz, 1998), eliminando-se as redundâncias. Portanto, a utilização da base em questão reduz a probabilidade dos classificadores se deslocarem para a falha. Além disso, o número de registros disponíveis é adequado para avaliar as precisões dos classificadores projetados.

6.3 Pré-processamento

Para o escopo deste trabalho, não foi necessário realizar tarefas comumente desempenhadas na etapa de pré-processamento, com exceção da seleção de atributos. A base NSL-KDD foi construída contemplando o tratamento dos problemas que impactam negativamente os conjuntos de dados (Tavallaee et al., 2009), como representatividade dos dados, dados faltantes, duplicatas, e outros.

O processo de seleção de atributos utilizando o algoritmo *Gain Ratio Attribute Evaluation* gerou um ranqueamento dos mesmos, e a partir disso foram criados três subconjuntos, com 30, 20 e 12 atributos. Como resultado do algoritmo de seleção de atributos, obteve-se uma lista dos atributos mais representativos em termos do cálculo do ganho de informação. Com isso, foram selecionados, como limites para definição das sub-bases, os valores próximos a mediana, primeiro quartil e terceiro quartil, pois são onde os valores de taxa de ganho de informação apresentaram uma diferença significativa em relação aos valores seguintes.

Em paralelo a seleção de atributos com o algoritmo *Gain Ratio Attribute Evaluation*, foi construída uma base de dados com 6 atributos, contendo os atributos que podem ser facilmente obtidos no processo de captura de pacotes em rede. A motivação para a criação de tal base deu-se pelo fato da maioria dos atributos presentes na base completa serem originados de pós-processamento de uma ou varias conexões ou sessões, fato que dificulta a utilização de um modelo treinado com esses atributos em tempo real. O objetivo foi então treinar um modelo apenas com atributos de captura instantânea ou com pouco processamento, facilitando assim a implantação desse método em uma aplicação de tempo real.

6.4 Processamento

Na etapa de processamento do processo KDD foram realizados experimentos considerando os métodos de detecção abordados (Seção 5.2) neste trabalho, assim como as bases de dados definidas na Tabela 5.8 (Seção 5.5.4). A técnica de *cross-validation* foi aplicada em cada experimento, com o objetivo de obter uma estimativa de quão preciso as abordagens são na prática.

O processo de experimentação foi dividido em duas etapas. Na primeira foram realizados experimentos para encontrar as melhores abordagens para definição dos limites das *faixas intermediárias* do método híbrido, considerando cada umas das sub-bases definidas na Tabela 5.8 (Seção 5.5.4). Na segunda etapa foram realizados experimentos para comparar o método híbrido proposto (Seção 5.2), com as técnicas RNA e KDD aplicadas isoladamente.

6.4.1 1ª Etapa: Experimentos para Definir Faixa Intermediária do Método Híbrido

A primeira etapa consiste na execução de experimentos para definir as melhores abordagens para a definição dos limites das *faixas intermediárias*, a ser aplicada na abordagem híbrida aplicando cada uma das bases de dados definidas na Tabela 5.8. Tais protocolos experimentais foram definidos na Seção 5.5.4.

O método Híbrido possui uma *faixa intermediária* em relação aos valores do neurônio de saída da RNA. As amostras que obtiveram valores finais dentro do conjunto de valores da *faixa intermediária*, caracterizam exemplos em que o modelo neural não conseguiu classificar de modo preciso, e desta forma, podem acarretar em falos positivos ou negativos. Já no método Híbrido, tais amostras que passam por um segundo método de classificação, o algoritmo KNN, possibilitando ter uma classificação mais precisa.

Durante os experimentos, observou-se que a RNA possui uma alta taxa de detecção para eventos não intrusivos, mas não tem um bom desempenho para detectar eventos intrusivos. Por outro lado, o método KNN possui uma alta taxa de detecção tanto para eventos intrusivos quanto para eventos não intrusivos, mas com um alto custo de processamento.

Assim, tendo em vista os falsos negativos gerados pela RNA, no método Híbrido é necessário submeter tais exemplos para o algoritmo KNN. As amostras intrusivas que foram classificadas erroneamente pela RNA como não intrusivas, obtiveram valor do neurônio de saída da RNA entre -1 à 0. Portanto a *faixa intermediária* deve abranger uma grande parte destes exemplos.

Seguindo esta linha de raciocínio, não é necessário que a *faixa intermediária* englobe muitas amostras com valor do neurônio de saída entre 0 à 1, pois a RNA acerta à maioria dos eventos não intrusivos. Portanto apenas uma pequena parte de não intrusivos obtém valor maior que 0 no neurônio de saída da RNA.

As oito abordagens para definição de limites da *faixa intermediária* foram apresentadas na Tabela 5.9 (Seção 5.5.4), tendo em vista as características supracitadas. Tais abordagens foram avaliadas em cada uma das cinco bases de dados definidas (Seção 5.5.2, Tabela 5.8). Os resultados obtidos em cada um dos experimentos executados para a definição dos limites das faixas de valores do método híbrido podem ser observados a seguir nas tabelas 6.1, 6.2, 6.3, 6.4 e 6.5. Do mesmo modo, a fim de facilitar o entendimento do desempenho de cada uma das diferentes configurações do método Híbrido, foram construídos os gráficos apresentados nas Figuras 6.1, 6.2, 6.3, 6.4 e 6.5.

Em relação às três primeiras bases testadas (NSL-KDD Completa, com 30 e com 20 atributos), a abordagem P25-N1 obteve a maior taxa de detecção, alcançando 99,70%, 99,70% e 99,68% de precisão, respectivamente para cada uma das bases; e submeteu 65,81%, 69,74% e

Tabela 6.1: Resultado dos experimentos para definir os valores de fronteira da *faixa intermediária* para a base completa. Fonte: O autor.

Abordagem	Taxa de Detecção	Exemplos Submetidos ao KNN
P1-N75	98,75%	18,52%
P1-N50	99,14%	32,51%
P1-N25	98,86%	43,97%
P1-N1	99,14%	54,22%
P25-N75	99,46%	30,09%
P25-N50	99,50%	45,91%
P25-N25	99,60%	56,37%
P25-N1	99,70%	65,81%

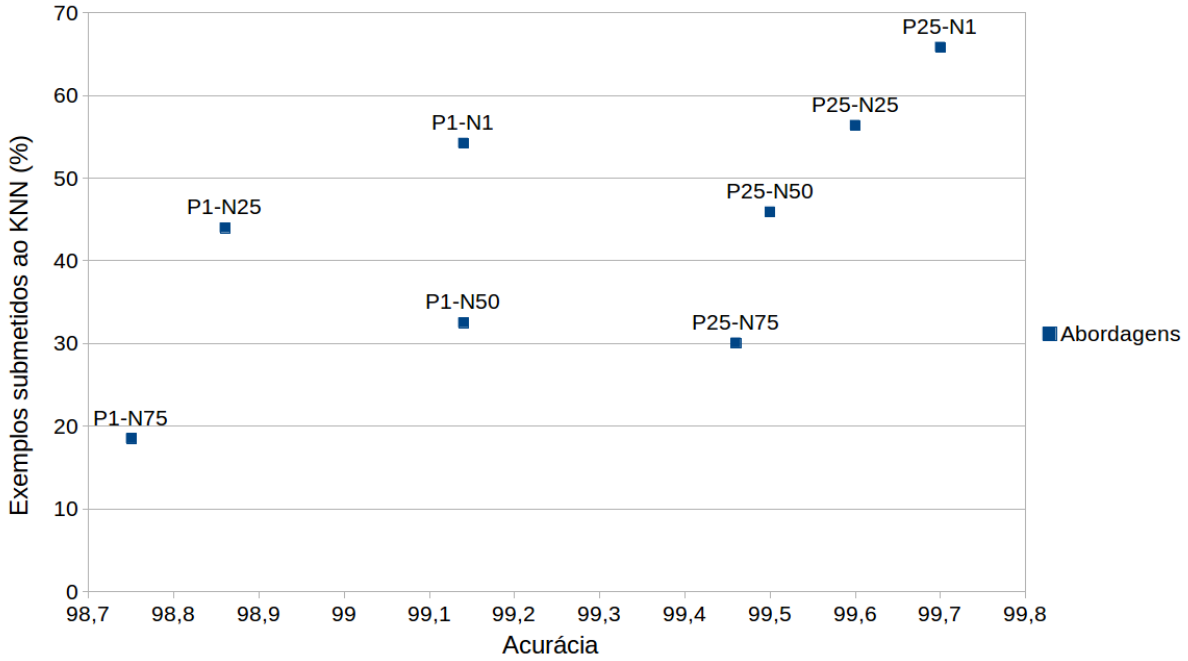


Figura 6.1: Gráfico dos resultados do experimento para definir a *faixa intermediária* para a base completa. Fonte: O autor.

65,68% dos exemplos para o algoritmo KNN. A abordagem P25-N75 apresentou uma taxa de detecção ligeiramente menor: 99,46% para a base completa; 99,44% para a base de 30 atributos; e 99,42% para a base de 20 atributos. Entretanto, essa abordagem submeteu apenas, respectivamente para cada base, 30,09%, 29,84% e 39,78% dos exemplos testados para o algoritmo KNN, acarretando em uma forte redução na carga de processamento necessária em relação às demais abordagens, inclusive a P25-N1. Devido a tais resultados, foram selecionadas as abordagens P25-N75 e P25-N1 do método Híbrido para as bases completa, com 30 e com 20 atributos. Sendo assim, tais abordagens compuseram a caixa preta ilustrada na Figura 5.10 do Capítulo 5, a qual será utilizada na fase posterior do processamento dos dados. Estas abordagens foram selecionadas porque apresentaram os resultados mais relevantes dentre as faixas de valores testadas para as bases em questão.

Tabela 6.2: Resultado dos experimentos para definir os valores de fronteira da *faixa intermediária* para a base com 30 atributos. Fonte: O autor.

Abordagem	Taxa de Detecção	Exemplos Submetidos ao KNN
P1-N75	97,40%	31,89%
P1-N50	98,90%	31,45%
P1-N25	99,21%	45,54%
P1-N1	99,19%	58,59%
P25-N75	99,44%	29,84%
P25-N50	99,37%	45,23%
P25-N25	99,67%	58,94%
P25-N1	99,70%	69,74%

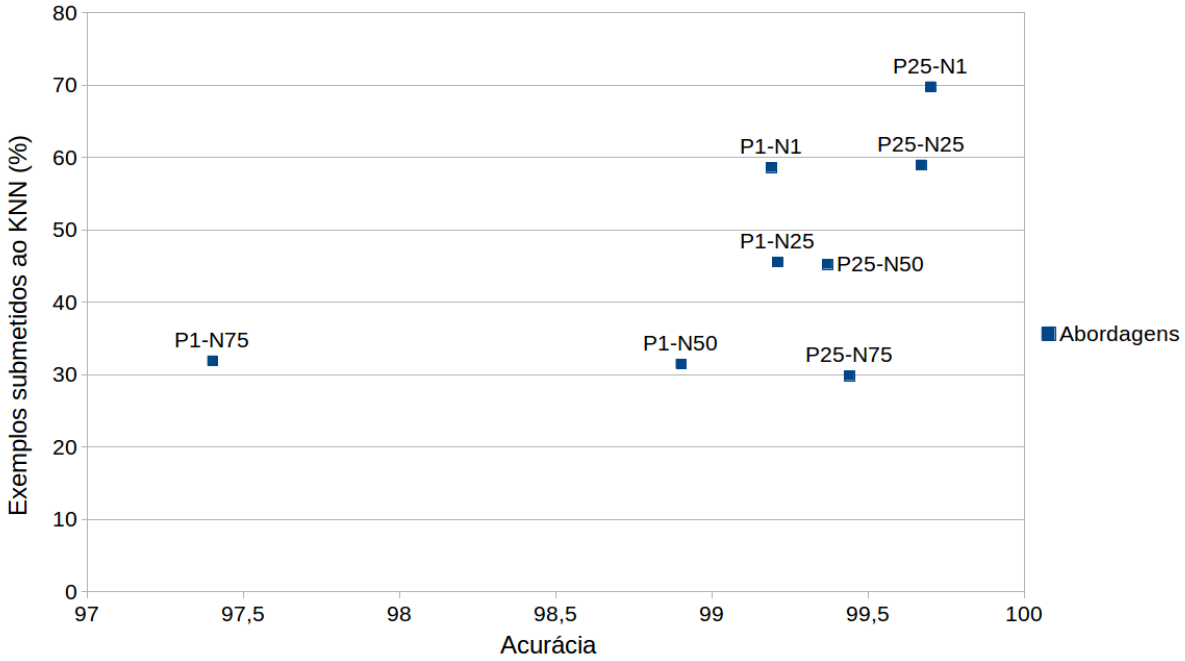


Figura 6.2: Gráfico dos resultados do experimento para definir a *faixa intermediária* para a base de 30 atributos. Fonte: O autor.

Em contrapartida, para a base de dados com 12 atributos foram selecionadas as abordagens P1-N75 e P1-N1 para compor a caixa preta do método Híbrido nos experimentos da etapa seguinte. A abordagem P1-N1 porque obteve a maior taxa de detecção dentre as abordagens aplicadas na base com 12 atributos, com aproximadamente 97,23% de acertos, submetendo 70,18% dos exemplos para o algoritmo KNN, e a abordagem P1-N75 porque mostrou-se a abordagem menos custosa em termos da capacidade de processamento exigida (cerca de 35,24% dos exemplos submetidos ao KNN), ainda assim mantendo uma taxa de detecção de 96,03%.

Seguindo a mesma linha das escolhas das abordagens do método Híbrido para as bases anteriores, na base com 6 atributos as abordagens P1-N75 e P25-N1 foram escolhidas para compor a caixa preta. Apresentando a melhor taxa de detecção, a abordagem P25-N1 garantiu que 97,10% dos exemplos fossem classificados corretamente, com 73,05% dos exemplos sendo

Tabela 6.3: Resultado dos experimentos para definir os valores de fronteira da *faixa intermediária* para a base com 20 atributos. Fonte: O autor.

Abordagem	Taxa de Detecção	Exemplos Submetidos ao KNN
P1-N75	99,06%	30,80%
P1-N50	99,26%	30,20%
P1-N25	99,39%	44,29%
P1-N1	99,38%	59,21%
P25-N75	99,42%	39,78%
P25-N50	99,49%	46,88%
P25-N25	99,59%	57,61%
P25-N1	99,68%	65,68%

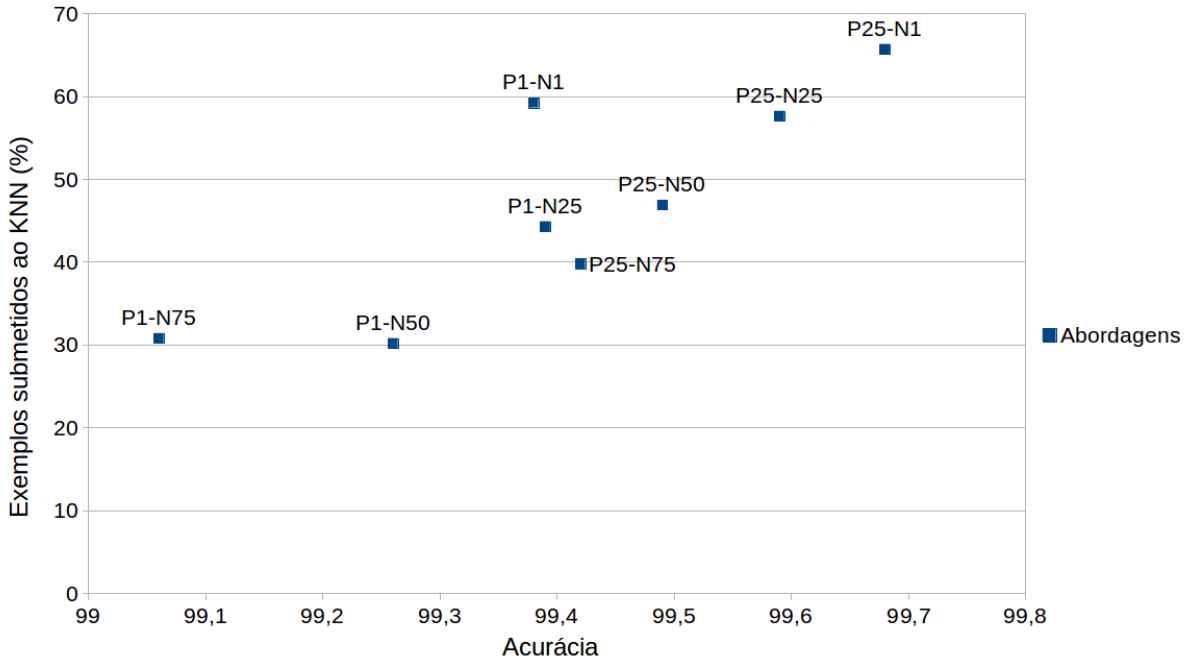


Figura 6.3: Gráfico dos resultados do experimento para definir a *faixa intermediária* para a base de 20 atributos. Fonte: O autor.

submetidos ao algoritmo KNN. Visando a redução do processamento necessário para a classificação, também foi escolhida a abordagem P1-N75, a qual alcançou 94,78% de taxa de acurácia e 30,08% dos exemplos enviados para serem classificados pelo algoritmo KNN.

Após a análise dos resultados gerados na 1ª fase da etapa de processamento, foi elaborada a Tabela 6.6. Esta tabela apresenta as diferentes configurações a serem utilizadas nos experimentos da 2ª fase, considerando variações das bases de dados utilizadas, dos métodos de classificação e de suas características.

Tabela 6.4: Resultado dos experimentos para definir os valores de fronteira da *faixa intermediária* para a base com 12 atributos. Fonte: O autor.

Abordagem	Taxa de Detecção	Exemplos Submetidos ao KNN
P1-N75	96,03%	35,24%
P1-N50	96,68%	44,86%
P1-N25	96,97%	51,72%
P1-N1	97,23%	70,18%
P25-N75	94,90%	28,83%
P25-N50	96,61%	52,89%
P25-N25	96,15%	61,11%
P25-N1	96,92%	70,28%

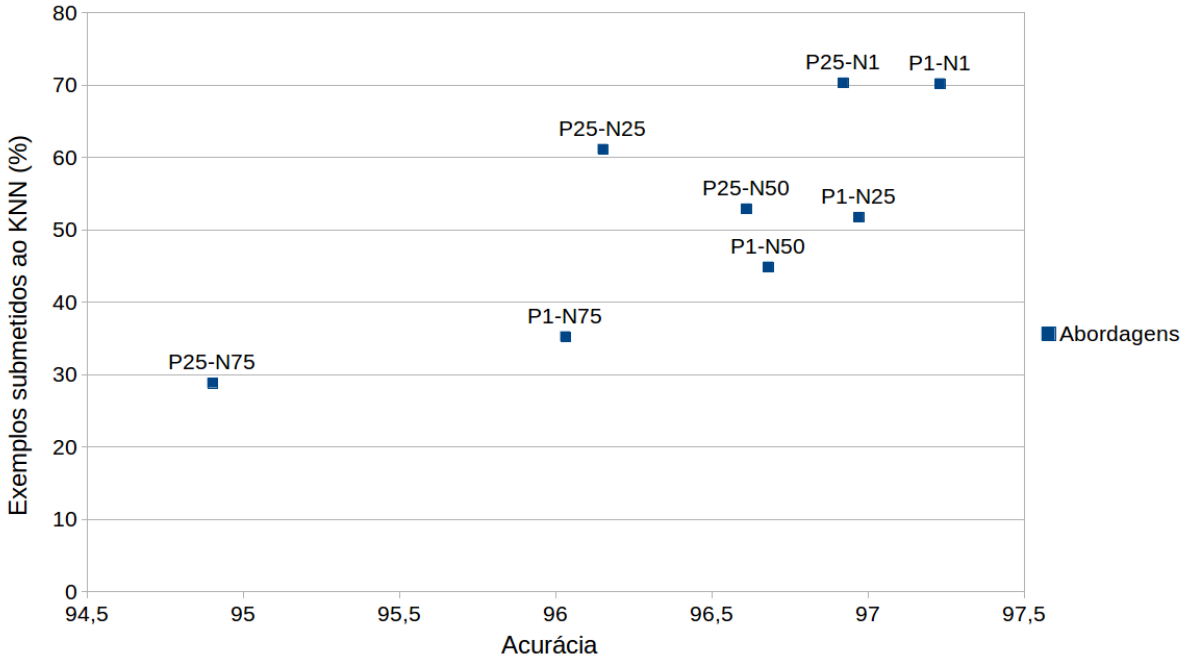


Figura 6.4: Gráfico dos resultados do experimento para definir a *faixa intermediária* para a base de 12 atributos. Fonte: O autor.

Tabela 6.5: Resultado dos experimentos para definir os valores de fronteira da *faixa intermediária* para a base de 6 atributos. Fonte: O autor.

Abordagem	Taxa de Detecção	Exemplos Submetidos ao KNN
P1-N75	94,78%	30,08%
P1-N50	96,36%	56,06%
P1-N25	96,98%	56,08%
P1-N1	97,02%	57,40%
P25-N75	94,90%	54,64%
P25-N50	95,53%	71,02%
P25-N25	96,87%	67,00%
P25-N1	97,10%	73,05%

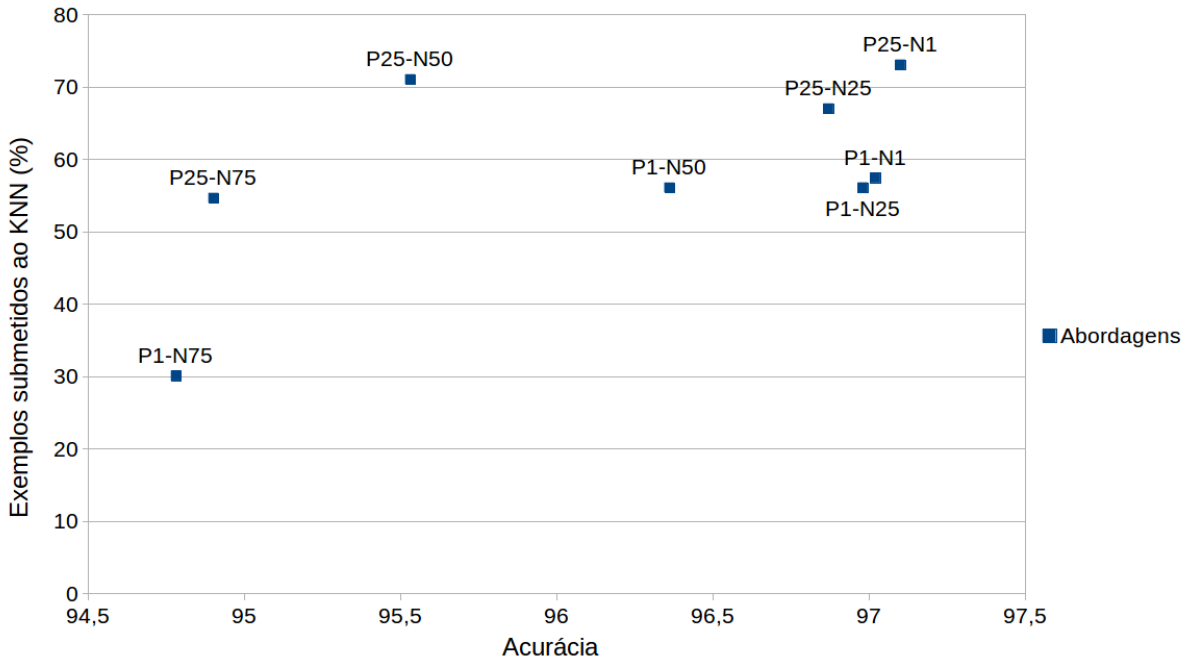


Figura 6.5: Gráfico dos resultados do experimento para definir a *faixa intermediária* para a base de 6 atributos. Fonte: O autor.

Tabela 6.6: Descrição das abordagens a serem utilizadas nos experimentos da 2ª fase de processamento. Fonte: O autor.

Configuração	**Base de dados**	**Método de detecção**
RNA-C	NSL-KDD Completa	RNA
KNN-C	NSL-KDD Completa	KNN
HIB(P25-N1)-C	NSL-KDD Completa	Híbrido (abordagem P25-N1)
HIB(P25-N75)-C	NSL-KDD Completa	Híbrido (abordagem P25-N75)
RNA-30	NSL-KDD 30 atributos	RNA
KNN-30	NSL-KDD 30 atributos	KNN
HIB(P25-N1)-30	NSL-KDD 30 atributos	Híbrido (abordagem P25-N1)
HIB(P25-N75)-30	NSL-KDD 30 atributos	Híbrido (abordagem P25-N75)
RNA-20	NSL-KDD 20 atributos	RNA
KNN-20	NSL-KDD 20 atributos	KNN
HIB(P25-N1)-20	NSL-KDD 20 atributos	Híbrido (abordagem P25-N1)
HIB(P25-N75)-20	NSL-KDD 20 atributos	Híbrido (abordagem P25-N75)
RNA-12	NSL-KDD 12 atributos	RNA
KNN-12	NSL-KDD 12 atributos	KNN
HIB(P1-N1)-12	NSL-KDD 12 atributos	Híbrido (abordagem P1-N1)
HIB(P1-N75)-12	NSL-KDD 12 atributos	Híbrido (abordagem P1-N75)
RNA-6	NSL-KDD 6 atributos	RNA
KNN-6	NSL-KDD 6 atributos	KNN
HIB(P25-N1)-6	NSL-KDD 6 atributos	Híbrido (abordagem P25-N1)
HIB(P1-N75)-6	NSL-KDD 6 atributos	Híbrido (abordagem P1-N75)

6.4.2 2ª Etapa: Experimentos entre os Métodos

Durante a execução dos experimentos foi possível observar que a técnica KNN possui uma taxa de acurácia melhor, quando comparada como o modelo neural. No entanto, a técnica KNN possui um custo de processamento bem mais elevado e realiza grande quantidade de operações durante a etapa de teste, fato que dificulta sua utilização em SDI para tempo real. Portanto, é importante que o método Híbrido melhore a taxa de acurácia em relação a RNA e também consiga reduzir o tempo de processamento, quando comparado com o método KNN.

Nesta fase de experimentação foram realizadas 10 execuções de cada uma das abordagens descritas na Tabela 6.6, e os resultados detalhados desses experimentos podem ser examinados no Apêndice A do presente trabalho. Os valores apresentados são referentes a média das 10 execuções para cada abordagem e os resultados da classificação das amostras estão organizados em termos de Falsos Negativos (FN), Falsos Positivos (FP), Verdadeiros Negativos (VN) e Verdadeiros Positivos (VP). As bases de dados utilizadas seguem as configurações definidas na subseção 5.5.2.

Além disso, a Tabela 6.7 apresenta as médias da quantidade de exemplos classificados corretamente e incorretamente por todas as abordagens aplicadas. As médias são referentes às 10 execuções realizadas para cada abordagem, conforme proposto no método experimental definido na Seção 5.5.4.

Todos os experimentos foram implementados em linguagem *Python*. A implementação dos métodos e de todo o processo de *cross-validation* foi realizada com auxílio das bibliotecas Keras[1], Tensorflow[2] e Scikitlearn[3]. Essas bibliotecas contemplam em sua estrutura todas a funcionalidades necessárias para criação, treinamento e utilização de RNA e do algoritmo KNN. Além disso, as bibliotecas empregadas na implementação do presente trabalho possuem vasta documentação e são todas ferramentas de software livre.

[1] https://keras.io/
[2] https://www.tensorflow.org/
[3] http://scikit-learn.org

Tabela 6.7: Média de exemplos classificados corretamente e incorretamente nos experimentos. Fonte: O autor.

Configuração	Média Acertos	Média Erros	Desvio Padrão
RNA-C	121342,5	4627,5	2624,744
KNN-C	125592,5	377,5	6,903
HIB(P25-N1)-C	125593,4	376,6	5,678
HIB(P25-N75)-C	125070,9	899,1	268,218
RNA-30	122761,5	3208,5	543,804
KNN-30	125590	380	7,629
HIB(P25-N1)-30	125586,9	383,1	7,609
HIB(P25-N75)-30	125082,8	887,2	165,639
RNA-20	122281,2	3688,8	1730,843
KNN-20	125579,9	390,1	8,312
HIB(P25-N1)-20	125579,1	390,9	10,034
HIB(P25-N75)-20	125082,5	887,5	125,768
RNA-12	112482,3	13487,7	4920,744
RNA-12	122277,5	3692,5	119,653
HIB(P1-N1)-12	122617,9	3352,1	233,984
HIB(P1-N75)-12	121414,7	4555,3	676,930
RNA-6	112637,6	13332,4	4594,642
RNA-6	122114,3	3855,7	372,907
HIB(P25-N1)-6	122330,4	3635,6	260,272
HIB(P1-N75)-6	120376,7	5593,3	1034,284

6.5 Pós-processamento

Nesta seção, os resultados obtidos nas 10 execuções do método experimental composto por 20 abordagens (cinco bases e quatro métodos), são avaliados e comparados.

Primeiramente, os valores de Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN) apresentados nas Tabelas A.1, A.2, A.3, A.4 e A.5 foram utilizados para calcular as taxas de acurácia, taxas de erro, taxas de sensibilidade e de especificidade, cujas fórmulas foram apresentadas na Seção 5.5.5.

Para a construção dos gráficos *boxplot* das Figuras 6.6, 6.7, 6.8, 6.9 e 6.10 foram utilizadas as taxas de acurácia das 10 execuções do processo de *cross-validation* para cada abordagem.

Nas Tabelas 6.9, 6.10, 6.11, 6.12 e 6.13 são apresentadas as médias das taxas de acurácia, taxas de erros, taxas de sensibilidade, taxas de especificidade, tempo de treino e tempo de teste das 20 abordagens selecionadas e aplicadas na segunda etapa dos experimentos, as quais foram apresentadas na Tabela 6.6. Além disso, as tabelas também dispõe os valores dos desvios padrão entre os elementos utilizadas no cálculo da média. As médias são referentes às 10 execuções realizadas para cada abordagem, conforme proposto na descrição do método experimental (Seção 5.5.4). A nomenclatura utilizada nas tabelas foi a seguinte: média (MD), desvio padrão

(DP) e segundos (s).

Para a comparação estatística entre as abordagens propostas na Tabela 6.6 foram utilizados os resultados do processo de *cross-validation* das 10 execuções de cada abordagem.

Inicialmente, foi aplicado o método Shapiro-Wilk para verificação da normalidade dos dados. O teste de normalidade apresentou *p-value* de $2.2e-16$, indicando que a hipótese de que os dados seguem uma distribuição normal foi rejeitada. É possível afirmar com nível de significância de 95% que os dados não seguem uma distribuição normal.

Devido ao fato dos resultados não seguirem uma distribuição normal, para a comparação das médias entre as abordagens foi utilizado o método não paramétrico para dados não pareados Kruskal-Wallis (Kruskal & Wallis, 1952). O resultado da comparação foi um $p-value$ de $2.2e-16$. Portanto, a hipótese de que todas as abordagens são iguais foi rejeitada. Desse modo, conclui-se que existe pelo menos uma abordagem que difere significativamente das demais.

Com o objetivo de encontrar quais abordagens se diferem significativamente, foi realizado o pós-teste de Dunn (Everitt et al., 2001) com nível de significância de 95%. Para melhor visualização, as tabelas apresentadas a seguir referentes aos resultados deste pós teste (Tabela 6.8 e Tabela 6.14) apresentam os valores do pós-teste de forma reduzida. O resultado completo do teste de pós-teste Dunn é apresentado na Tabela B.1 do Apêndice B.

6.5.1 Resultados e análise das abordagens aplicadas em cada base

Nesta seção serão apresentados os resultados referentes a aplicação das diferentes abordagens propostas neste trabalho em cada uma das bases de dados definidas anteriormente. Também será efetuada a avaliação do desempenho de cada uma delas, em termos das taxas de acurácia e dos tempos de processamento na execução da tarefa de classificação de intrusos. A Tabela 6.8 apresentada a seguir servirá como subsídio para as análises subsequentes.

Base NSL-KDD Completa

A Tabela 6.9 apresenta os resultados obtidos com a execução das abordagens RNA-C, KNN-C, HIB(P25-N1)-C e HIB(P25-N75)-C na base de dados NSL-KDD completa. Observa-se que a RNA-C possui uma média de acurácia de 96,33%, resultado consideravelmente inferior a média das abordagens KNN-C, HIB(P25-N1)-C e HIB(P25-N75)-C, cujos valores são, respectivamente, de 99,70%, 99,70% e 99,29%.

Tabela 6.8: Resultado reduzido do teste estatístico Dunn para comparação entre as abordagens. Comparações que tiveram diferenças significativas são mostradas em negrito. Fonte: O autor.

Comparações	Acurácia dos métodos	p-value
RNA-C – **KNN-C**	**96,33 - 99,70**	**1.537779e-04**
RNA-C – **HIB(P25-N1)-C**	**96,33 - 99,70**	**1.968701e-04**
RNA-C – HIB(P25-N75)-C	96,33 - 99,29	6.539991e-02
KNN-C – HIB(P25-N1)-C	99,70 - 99,70	9.513481e-01
KNN-C – HIB(P25-N75)-C	99,70 - 99,29	7.392223e-02
HIB(P25-N1)-C – HIB(P25-N75)-C	99,70 - 99,29	8.329375e-02
RNA-30 – **KNN-30**	**97,45 - 99,70**	**1.100504e-03**
RNA-30 – **HIB(P25-N1)-30**	**97,45 - 99,70**	**2.400526e-03**
RNA-30 – HIB(P25-N75)-30	97,45 - 99,29	1.679812e-01
KNN-30 – HIB(P25-N1)-30	99,70 - 99,70	8.616343e-01
KNN-30 – HIB(P25-N75)-30	99,70 - 99,29	1.315064e-01
HIB(P25-N1)-30 – HIB(P25-N75)-30	99,70 - 99,29	1.315064e-01
RNA-20 – **KNN-20**	**97,07 - 99,69**	**5.299997e-03**
RNA-20 – **HIB(P25-N1)-20**	**97,07 - 99,69**	**4.471831e-03**
RNA-20 – HIB(P25-N75)-20	97,07 - 99,30	1.375698e-01
KNN-20 – HIB(P25-N1)-20	99,69 - 99,69	9.618168e-01
KNN-20 – HIB(P25-N75)-20	99,69 - 99,30	2.354972e-01
HIB(P25-N1)-20 – HIB(P25-N75)-20	99,69 - 99,30	2.193949e-01
RNA-12 – KNN-12	89,29 - 97,07	8.834888e-02
RNA-12 – **HIB(P1-N1)-12**	**89,29 - 97,54**	**1.055913e-02**
RNA-12 – HIB(P1-N75)-12	89,29 - 96,38	2.985967e-01
KNN-12 – HIB(P1-N1)-12	97,07 - 97,54	4.656275e-01
KNN-12 – HIB(P1-N75)-12	97,07 - 96,38	5.747478e-01
HIB(P1-N1)-12 – HIB(P1-N75)-12	97,54 - 96,38	1.577834e-01
RNA-6 – KNN-6	89,42 - 96,94	1.288171e-01
RNA-6 – HIB(P25-N1)-6	89,42 - 97,11	5.216304e-02
RNA-6 – HIB(P1-N75)-6	89,42 - 95,56	5.489246e-01
KNN-6 – HIB(P25-N1)-6	96,94 - 97,11	7.269848e-01
KNN-6 – HIB(P1-N75)-6	96,94 - 95,56	4.281828e-01
HIB(P25-N1)-6 – HIB(P1-N75)-6	97,11 - 95,56	2.147020e-01
RNA-C - RNA-30	96,33 - 97,45	7.822897e-01
RNA-C - RNA-20	96,33 - 97,07	8.584615e-01

A Figura 6.6 apresenta o gráfico *boxplot* das taxas de acurácia das abordagens aplicadas na base NSL-KDD completa. No gráfico, é possível observar que a abordagem RNA-C possui uma alta variância entre suas taxas de acurácia. Por outro lado, as abordagens KNN-C, HIB(P25-

Tabela 6.9: Resultado dos experimentos realizados na base NSL-KDD completa. Fonte: O autor.

		RNA-C	**KNN-C**	**HIB(P25-N1)-C**	**HIB(P25-N75)-C**
Acurácia	MD(%)	96,33	99,70	99,70	99,29
	DP	2,082	0,005	0,004	0,214
Taxa de erro	MD(%)	03,67	00,30	00,30	00,71
	DP	2,082	0,005	0,004	0,214
Taxa de sensibilidade	MD(%)	93,78	99,68	99,69	98,81
	DP	4,260	0,010	0,009	0,281
Taxa de especificidade	MD(%)	98,55	99,72	99,71	99,70
	DP	0,485	0,011	0,007	0,284
Tempo de treino	MD(s)	1809,29	0,307	2351,190	1810,756
	DP	227,038	0,407	335,594	265,987
Tempo de classificação	MD(s)	1,650	244,855	166,326	68,133
	DP	0,224	16,182	13,272	10,037

N1)-C e HIB(P25-N75)-C possuei baixo desvio padrão, pois os valores de acurácia, resultantes das 10 execuções de cada abordagem, são muito próximos. saída

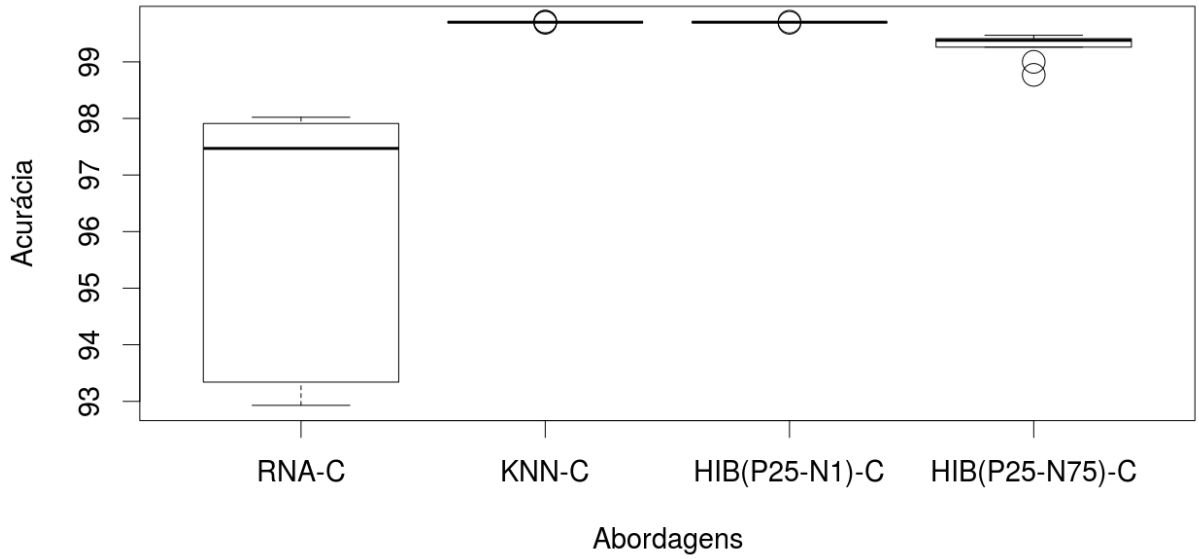


Figura 6.6: Gráfico *boxplot* das abordagens aplicadas na base completa. Fonte: O autor.

De acordo com a Tabela 6.9, a abordagem RNA-C tem um desempenho excelente em termos de tempo com apenas 1,650 segundos para classificar todos os exemplos da base NSL-KDD completa. No entanto, conforme supracitado, a taxa de acurácia é inferior as demais abordagens aplicadas na base completa.

Ao compararmos os tempos de execução de cada abordagem, não foi considerado o tempo de treino, pois o treinamento é uma tarefa onde os modelos de detecção são construídos. Somente após gerado e treinado, o modelo de detecção é aplicado para a tarefa de classificação propriamente dita. Por outro lado, o tempo de teste é bastante importante, pois corresponde ao

desempenho que cada abordagem leva para realizar a classificação dos exemplos.

As abordagens KNN-C e HIB(P25-N1)-C possuem taxa de acurácia média de 99,70%. É possível observar na Figura 6.6 do gráfico *boxplot* que os valores de acurácia de cada aplicação do *cross-validation*, para estas abordagens, obtiveram um comportamento muito semelhante e com baixa variância. Além disso, foi comprovado estatisticamente que não existem diferenças significativas entre os dois métodos, conforme o resultado do teste de Dunn apresentado na Tabela 6.8. Além de igualar a abordagem KNN-C em termos de taxa de acurácia, a abordagem HIB(P25-N1)-C conseguiu reduzir o tempo de teste/classificação, conforme pode ser observado na Tabela 6.9. A abordagem KNN-C levou 244,855 segundos para classificar a base NSL-KDD completa, enquanto a abordagem HIB(P25-N1)-C classificou a base NSL-KDD completa em 166,326 segundos. A redução foi de 32,07% do tempo de classificação da abordagem KNN-C.

Algumas aplicações, como soluções de *Internet of Things* (IoT), possuem recursos limitados. A utilização de IDS em aplicações desse tipo necessitam do menor processamento possível. Neste contexto, a abordagem HIB(P25-N75)-C torna-se bastante atrativa. Ela alcançou uma acurácia média de 99,29%, superior aos 96,33% da RNA-C, e inferior aos 99,70% das abordagens KNN-C e HIB(P25-N1)-C. A abordagem HIB(P25-N75)-C, de acordo com o gráfico *boxplot* da Figura 6.6, não apresenta grandes variações em seus valores de acurácia. O ponto positivo desta encontra-se no tempo de teste/classificação. Conforme apresentado na Tabela 6.9, ela classificou a base NSL-KDD completa em 68,133 segundos, apresentando uma expressiva redução de 72,17% em relação ao tempo da abordagem KNN-C e de 59,04% em relação ao tempo da abordagem HIB(P25-N1)-C. Além disso, a comparação estatística não apontou diferenças significativas entre o KNN-C e o HIB(P25-N75)-C, nem entre o HIB(P25-N1)-C e HIB(P25-N75)-C, fato que pode ser observado na Tabela 6.8.

Base NSL-KDD com 30 atributos

As abordagens RNA-30, KNN-30, HIB(P25-N1)-30 e HIB(P25-N75)-30 foram aplicadas na base NSL-KDD com 30 atributos. Os resultados obtidos são apresentados na Tabela 6.10. As médias de acurácia das abordagens induzem a conclusão de que o desempenho das demais abordagens são superiores a abordagem RNA-30, pois a RNA-30 alcançou uma acurácia média de 97,45%, enquanto as demais abordagens alcançaram taxas médias superiores a 99,29%. De fato, foi comprovado estatisticamente, através dos resultados apresentados na Tabela 6.8, que existem diferenças significativas entre as abordagens RNA-30 e KNN-30 e entre as abordagens RNA-30 e HIB(P25-N1)-30. No entanto, não foi possível comprovar estatisticamente que existem diferenças significativas entre as abordagens RNA-30 e HIB(P25-N75)-30.

A Figura 6.7 apresenta o gráfico *boxplot* das taxas de acurácia das abordagens RNA-30, KNN-30, HIB(P25-N1)-30 e HIB(P25-N75)-30. Este gráfico expõe maior variância entre os valores de acurácia da abordagem RNA-30 do que em relação aos valores das demais abordagens

Tabela 6.10: Resultado dos experimentos realizados na base NSL-KDD com 30 atributos. Fonte: O autor.

		RNA-30	**KNN-30**	**HIB(P25-N1)-30**	**HIB(P25-N75)-30**
Acurácia	MD(%)	97,45	99,70	99,70	99,29
	DP	0,430	0,007	0,005	0,131
Taxa de erro	MD(%)	02,55	00,30	00,30	00,71
	DP	0,430	0,007	0,005	0,131
Taxa de sensibilidade	MD(%)	96,68	99,68	99,69	98,78
	DP	0,245	0,011	0,011	0,307
Taxa de especificidade	MD(%)	98,12	99,71	99,71	99,75
	DP	0,811	0,008	0,009	0,149
Tempo de treino	MD(s)	1933,043	0,138	1957,657	1996,810
	DP	273,302	0,013	417,567	315,688
Tempo de classificação	MD(s)	1,779	225,956	155,684	81,038
	DP	0,271	13,455	12,724	16,733

aplicadas na base com 30 atributos.

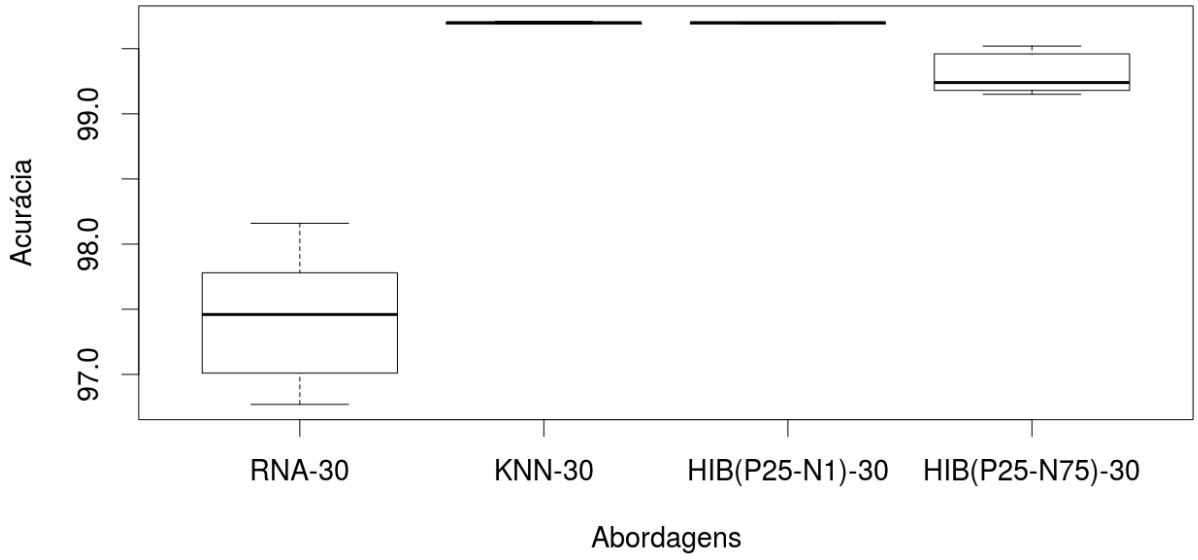


Figura 6.7: Gráfico *boxplot* das abordagens aplicadas na base com 30 atributos. Fonte: O autor.

A abordagem HIB(P25-N1)-30 alcançou 99,70% de taxa de acurácia média, se igualando com os 99,70% do KNN-30 neste critério. Além disso, é possível observar na Tabela 6.8 que não existem diferenças significativas entre as taxas de acurácia desses métodos. Portanto, além de atingir uma taxa de acurácia equivalente ao KNN, o método Híbrido realizou a tarefa realizando menos processamento. De acordo com a Tabela 6.10, a abordagem KNN-30 realizou a classificação dos exemplos da base NSL-KDD com 30 atributos em 225,956 segundos, enquanto a abordagem Híbrida HIB(P25-N1)-30 realizou a classificação dos exemplos do mesmo conjunto de dados em 155,684 segundos. Deste modo, houve uma redução de 31,10% no tempo de classificação entre os dois métodos citados.

A abordagem HIB(P25-N75)-C aplicada na base NSL-KDD completa e na base com 30 atributos, HIB(P25-N75)-30, tiveram resultados similares, principalmente tendo em vista aplicações que primam por melhorar o tempo de processamento mantendo boas taxas de acurácia. A abordagem HIB(P25-N75)-30 alcançou uma taxa de acurácia média de 99,29%, conforme consta na Tabela 6.10. Além disso, de acordo com a Tabela 6.8, o teste estatístico realizado não indicou diferença significativa entre a abordagem HIB(P25-N75)-30 e as abordagens KNN-30 e HIB(P25-N1)-30, as quais apresentaram as maiores taxas médias de acurácia. A redução do tempo de processamento em relação a essas abordagens também é um ponto satisfatório da abordagem HIB(P25-N75)-30. Ela classificou os exemplos da base NSL-KDD com 30 atributos em 81,038 segundos, cerca de 64,14% mais rápido que o KNN-30 (225,956 segundos) e 47,95% mais rápido que o HIB(P25-N1)-30 (155,684 segundos).

Base NSL-KDD com 20 atributos

Os resultados apresentados na Tabela 6.11 foram obtidos utilizando-se a sub-base NLS-KDD com 20 atributos, considerando as abordagens RNA-20, KNN-20, HIB(P25-N1)-20 e HIB(P25-N75)-20. A taxa de acurácia média obtida pela RNA-20 foi de 97,07%, inferior aos valores médios obtidos pelas demais abordagens aplicadas na base NSL-KDD com 20 atributos. Tal fato caracterizou o mesmo padrão comportamental observado em relação as mesmas abordagens aplicadas nas bases de dados apresentadas anteriormente (NLS-KDD completa e NLS-KDD com 20 atributos). As abordagens KNN-20 e HIB(P25-N1)-20 alcançaram 99,69% de acurácia e a abordagem HIB(P25-N75)-20 obteve 99,30% de acurácia média. A diferença entre as estratégias foi comprovada estatisticamente, visto que os resultados dessas comparações são apresentados na Tabela 6.8. Houve diferença significativa entre as taxas de acurácia da RNA-20 em relação ao KNN-20 e da RNA-20 em relação ao HIB(P25-N1)-20. No entanto, não foi possível concluir estatisticamente que existe diferença significativa entre as abordagens RNA-20 e HIB(P25-N75)-20 em termos da taxa de acurácia.

A Figura 6.8 apresenta o gráfico *boxplot* das taxas de acurácia das abordagens RNA-20, KNN-20, HIB(P25-N1)-20 e HIB(P25-N75)-20 aplicadas na base NSL-KDD com 20 atributos. Observa-se que a abordagem RNA-20 apresentou maior variância nas taxas de acurácia. Além disso, é possível notar que o KNN-20 e o HIB(P25-N1)-20 praticamente não tiveram variância entre suas taxas de acurácia.

A abordagem KNN-20 obteve uma média de taxa de detecção de 99,69%, e de acordo com os dados apresentado na Tabela 6.11, utilizou 209,211 segundos para realizar a classificação da base de 20 atributos. O HIB(P25-N1)-20 alcançou uma média de taxa de acurácia de 99,69%, mesmo valor alcançado pelo KNN-20. De fato, o resultado do teste estatístico apresentado na Tabela 6.8 demonstra que não foram encontradas diferenças estatisticamente significativas entre essas abordagens. No entanto, o HIB(P25-N1)-20 realizou a classificação dos exemplos da base NSL-KDD com 20 atributos em 155,359 segundos, representando uma melhoria no tempo de

Tabela 6.11: Resultado dos experimentos realizados na base NSL-KDD com 20 atributos. Fonte: O autor.

		RNA-20	**KNN-20**	**HIB(P25-N1)-20**	**HIB(P25-N75)-20**
Acurácia	MD(%)	97,07	99,69	99,69	99,30
	DP	1,373	0,007	0,008	0,099
Taxa de erro	MD(%)	02,93	00,31	00,31	00,70
	DP	1,373	0,007	0,008	0,099
Taxa de sensibilidade	MD(%)	95,65	99,68	99,69	98,82
	DP	2,894	0,009	0,010	0,149
Taxa de especificidade	MD(%)	98,31	99,70	99,69	99,71
	DP	1,046	0,009	0,009	0,136
Tempo de treino	MD(s)	1882,881	0,127	2034,033	1779,344
	DP	279,641	0,014	298,731	599,368
Tempo de classificação	MD(s)	1,632	219,211	155,359	83,266
	DP	0,235	16,731	11,952	11,072

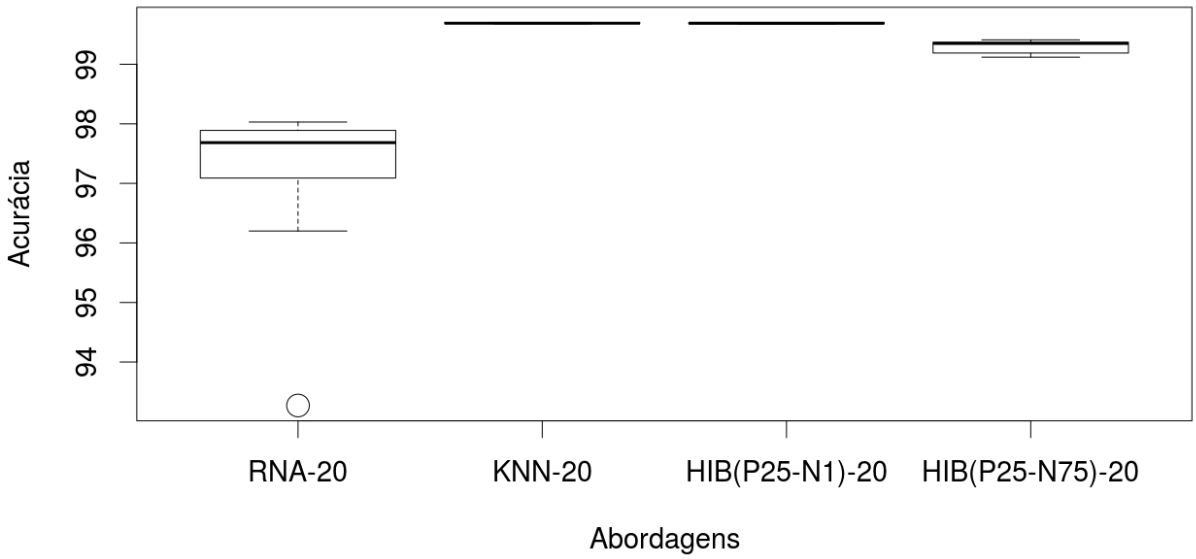


Figura 6.8: Gráfico *boxplot* das abordagens aplicados na base com 20 atributos. Fonte: O autor.

classificação de 25,74% em relação ao KNN-20.

O HIB(P25-N75)-20 alcançou uma média de taxa de acurácia de 99,30%, por consequência, demonstrou capacidade de detecção bastante próxima das taxas do KNN-20 e do HIB(P25-N1)-20. Estatisticamente, não houveram diferenças significativas entre a abordagem HIB(P25-N75)-20 e as outras duas abordagens, conforme as informações apresentadas na Tabela 6.8. É importante ressaltar a melhoria obtida pela abordagem HIB(P25-N75)-20 em relação ao KNN-20 e ao HIB(P25-N1)-20 no âmbito do tempo de processamento exigido. De acordo com a Tabela 6.11, o HIB(P25-N75)-20 classificou os exemplos da base NSL-KDD com 20 atributos em 83,266 segundos. A abordagem KNN-20 realizou a mesma tarefa em 209,211 segundos e a abordagem HIB(P25-N1)-20 classificou os exemplos da base de 20 atributos em 155,359 segundos. A melhoria é de 60,20% e 46,40% no tempo de classificação, respectivamente. Por-

tanto, em uma aplicação que prioriza a economia de recursos a abordagem HIB(P25-N75)-20 pode ser uma alternativa bastante satisfatória.

Base NSL-KDD com 12 atributos

Os resultados apresentados na Tabela 6.12 são referentes a aplicação das abordagens RNA-12, KNN-12, HIB(P1-N1)-12 e HIB(P1-N75)-12 na base NSL-KDD com 12 atributos. Assim como observado nos casos anteriores, a abordagem que utiliza apenas RNA, neste caso específico a RNA-12, obteve a pior taxa de acurácia média, com apenas 89,29% de acerto na tarefa de classificação de intrusos. As abordagens KNN-12, HIB(P1-N1)-12 e HIB(P1-N75)-12, obtiveram 97,07%, 97,54% e 96,38%, respectivamente.

De acordo com o resultado do teste estatístico de Dunn, apresentado na Tabela 6.8, só existem diferenças estatisticamente significativas entre as abordagens RNA-12 e HIB(P1-N1)-12. Este fato confirma o bom desempenho do método híbrido, uma vez que a abordagem HIB(P1-N1)-12 superou a RNA-12, algo que o KNN-12 não foi capaz devido a grande variância das taxas de acurácia média da abordagem com RNA. Além disso, a abordagem HIB(P1-N1)-12 superou o KNN-12 em termos de acurácia média. Entretanto, de acordo com o resultado do teste de Dunn (Tabela 6.8), essa diferença não foi estatisticamente significativa.

Tabela 6.12: Resultado dos experimentos realizados na base NSL-KDD com 12 atributos. Fonte: O autor.

		RNA-12	**KNN-12**	**HIB(P1-N1)-12**	**HIB(P1-N75)-12**
Acurácia	MD(%)	89,29	97,07	97,54	96,38
	DP	3,907	0,096	0,619	0,537
Taxa de erro	MD(%)	10,71	02,93	02,46	03,62
	DP	3,907	0,096	0,619	0,537
Taxa de sensibilidade	MD(%)	77,44	98,83	99,64	96,74
	DP	8,384	0,080	0,165	0,823
Taxa de especificidade	MD(%)	99,62	95,54	95,34	96,08
	DP	0,185	0,168	0,288	0,599
Tempo de treino	MD(s)	1241,846	0,295	1453,967	1346,531
	DP	188,834	0,380	308,109	203,741
Tempo de classificação	MD(s)	1,609	199,710	136,137	67,994
	DP	0,233	1,034	19,440	16,468

Ao analisar o tempo de classificação na Tabela 6.12, também pode-se observar uma melhoria no tempo necessário para as abordagens classificarem os exemplos da base de 12 atributos. A abordagem KNN-12 precisou de 199,710 segundos para realizar a classificação dos exemplos da base NSL-KDD com 12 atributos, enquanto a híbrida HIB(P1-N1)-12 realizou a classificação em 136,137 segundos, ocasionando uma melhoria de 31,83%.

A Figura 6.9 apresenta o gráfico *boxplot* das taxas de acurácia das abordagens RNA-12, KNN-12, HIB(P1-N1)-12 e HIB(P1-N75)-12. Dentre elas, observa-se que a abordagem que

apresentou maior variância nos valores da taxa de acurácia referente as 10 execuções foi a RNA-12.

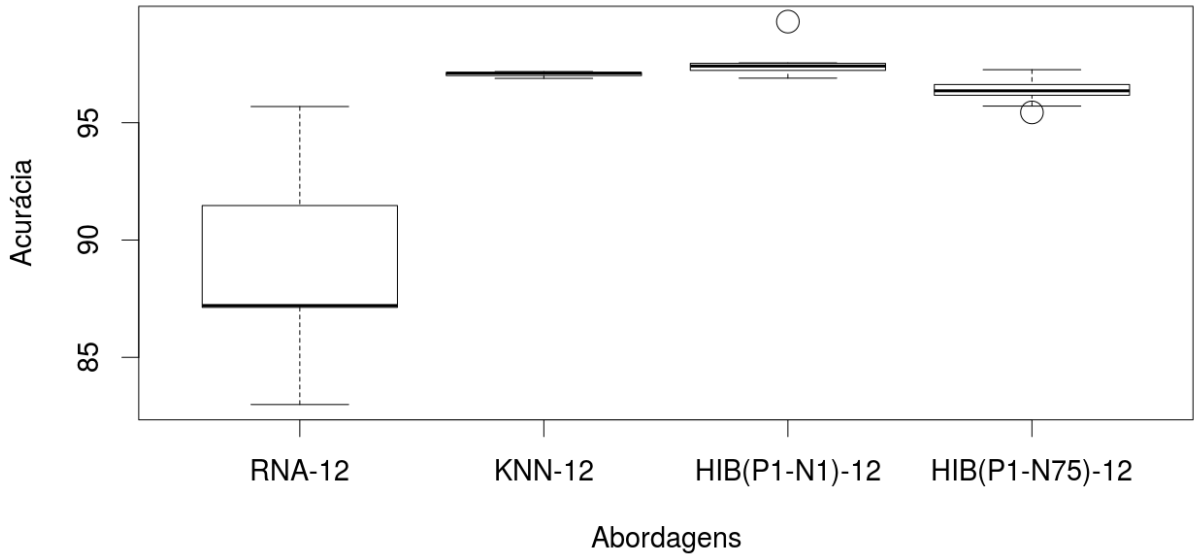


Figura 6.9: Gráfico *boxplot* das abordagens aplicadas na base com 12 atributos. Fonte: O autor.

A segunda abordagem híbrida aplicada na base NSL-KDD com 12 atributos foi a HIB(P1-N75)-12. De acordo com a tabela de resultados, esta obteve uma acurácia média de 96,38%. Portanto, superior a abordagem RNA-12 neste quesito, que conforme supracitado foi de 89,29%. As abordagens KNN-12 e HIB(P1-N1)-12 obtiveram taxas de acurácia média de 97,07% e 97,54%, respectivamente. No entanto, após a aplicação do pós teste estatístico, não foram encontradas diferenças significativas entre a abordagem híbrida HIB(P1N75)-12 em relação às outras abordagens aplicadas para essa base.

Ao avaliar as abordagens em relação ao tempo de classificação, observa-se na Tabela 6.12 que a abordagem HIB(P1-N75)-12 levou 67,994 segundos para realizar a classificação, o que caracteriza uma melhoria de 65,95% em relação ao tempo do KNN-12 e de 50,05% em relação ao tempo do HIB(P1-N1)-12.

Base NSL-KDD com 6 atributos

A Tabela 6.13 expõe os resultados obtidos com a aplicação das abordagens RNA-6, KNN-6, HIB(P25-N1)-6 e HIB(P1-N75)-6 na base NSL-KDD com 6 atributos. Observa-se que a abordagem somente com Redes Neurais Artificiais (RNA-6) novamente apresentou uma taxa de acurácia média inferior das demais abordagens, com valor igual a 89,42%. O KNN-6 apresentou taxa acurácia média de 96,94%, média inferior a da abordagem híbrida HIB(P25-N1)-6, que alcançou uma acurácia média de 97,11%. No entanto, de acordo com a avaliação estatística realizada e apresentada na Tabela 6.8, não foram encontradas diferenças significativas entre as abordagens.

Tabela 6.13: Resultado dos experimentos realizados na base NSL-KDD com 6 atributos. Fonte: O autor.

		RNA-6	**KNN-6**	**HIB(P25-N1)-6**	**HIB(P1-N75)-6**
Acurácia	MD(%)	89,42	96,94	97,11	95,56
	DP	3,647	0,295	0,206	0,822
Taxa de erro	MD(%)	10,58	03,06	02,89	04,44
	DP	3,647	0,295	0,206	0,822
Taxa de sensibilidade	MD(%)	79,24	98,97	99,37	96,11
	DP	7,914	0,584	0,358	1,467
Taxa de especificidade	MD(%)	98,27	95,18	95,15	95,09
	DP	0,371	0,214	0,206	0,461
Tempo de treino	MD(s)	1216,531	0,102	1337,700	1229,730
	DP	287,891	0,013	471,925	246,686
Tempo de classificação	MD(s)	1,537	207,905	156,564	83,384
	DP	0,201	19,850	16,556	28,208

A abordagem composta pela RNA (RNA-6) continuou a apresentar a maior variância entre os valores de taxa de acurácia, conforme pode ser observado na Figura 6.10. A figura apresenta o gráfico *boxplot* das taxas de acurácia obtidas por cada uma das abordagens aplicadas na base NSL-KDD com 6 atributos.

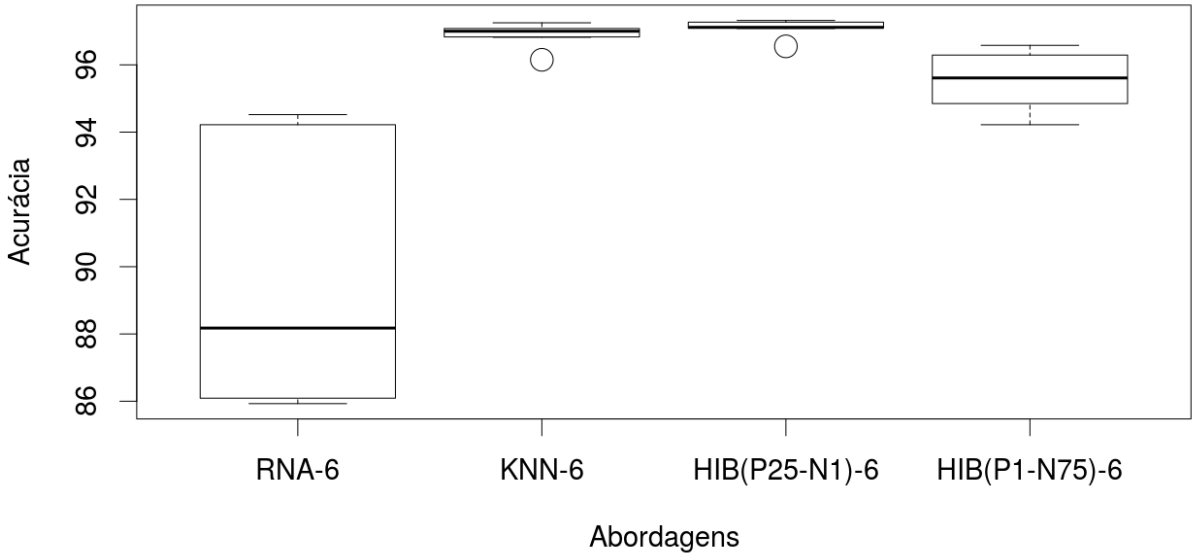


Figura 6.10: Gráfico *boxplot* das abordagens aplicadas na base de 6 atributos. Fonte: O autor.

Considerando o desempenho em termos do tempo necessário para classificação dos exemplos, o KNN-6 realizou a classificação em 207,915 segundos e o HIB(P25-N1)-6 em 156,564 segundos. Dessa maneira, a utilização da abordagem híbrida HIB(P25-N1)-6 otimizou o tempo necessário para classificação dos exemplos em relação ao KNN-6 em 24,70%.

A segunda abordagem híbrida aplicada na base NSL-KDD com 6 atributos foi a HIB(P1-N75)-6. Ela apresentou uma taxa de acurácia média de 95,56%. Ao compará-la com as demais

abordagens aplicadas, observa-se que seu desempenho em termos de acurácia média superou apenas a RNA-6 (89,42%). Todavia, esta abordagem proporciona uma interessante melhoria no processamento necessário para classificação. De acordo com a Tabela 6.13, a abordagem HIB(P1-N75)-6 desempenhou a tarefa de classificação em 83,384 segundos. Desse modo, a utilização da abordagem híbrida HIB(P1-N75)-6 representa uma melhoria de 59,89% no tempo de processamento em relação ao KNN-6 (207,915 segundos) e de 46,74% em relação ao HIB(P25-N1)-6 (156,564 segundos).

Os seis atributos que compõem a base em questão podem ser facilmente obtidos no processo de captura de pacotes em rede, atributos cuja captura é instantânea ou com pouco processamento, podendo assim ser realizado em tempo real. Em vista disso, mesmo sem ter alcançado taxas de acurácia média superiores as das abordagens aplicadas nas demais bases, as abordagens RNA-6, KNN-6, HIB(P25-N1)-6 e HIB(P1-N75)-6 podem ter um destaque maior em aplicações que exigem detecção em tempo real.

6.5.2 Resultados e análise das abordagens híbridas

Esta seção apresenta especificamente os resultados e análise da aplicação de cada abordagem híbrida nas diferentes bases de dados definidas neste trabalho (Seção 5.5.4). Sendo assim, a Tabela 6.14 dispõe os resultados do pós-teste Dunn, especificamente para estes casos.

O gráfico apresentado na Figura 6.11 permite a visualização das abordagens híbridas em relação a sua acurácia e de tempo de classificação. Observa-se que as abordagens HIB(P25-N1)-C, HIB(P25-N1)-30 e HIB(P25-N1)-20 se agrupam em uma região que indica uma alta taxa de acurácia média e alto tempo de processamento. Por outro lado, as abordagens HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20 se concentram em uma região que também sinaliza altas taxas de acurácia média, mas com redução aparente do tempo de processamento. De modo contrário, as abordagens aplicadas nas bases NSL-KDD com 12 e com 6 atributos não estão contidas nas regiões que sugerem bom desempenho.

As abordagens aplicadas na base NSL-KDD com 12 atributos (HIB(P1-N1)-12 e HIB(P1-N75)-12) e na base NSL-KDD com 6 atributos (HIB(P25-N1)-6 E HIB(P25-N75)-6) obtiveram taxas de acurácia média inferiores às demais abordagens. De fato, observa-se na Tabela 6.14 que foram encontradas diferenças estatisticamente significativas entre essas abordagens e as demais. Além disso, também não houveram melhorias satisfatórias no tempo de classificação, conforme pode ser observado na Tabela 6.15.

Esse fato é consequente da estrutura do método híbrido, no qual a primeira técnica aplicada no método (RNA – vide Seção 5.2) não possui boa acurácia para as bases NSL-KDD com 12 e com 6 atributos. Desse modo, uma quantidade maior de exemplos são submetidos para o KNN, a qual compreende na técnica mais custosa computacionalmente aplicada no método Híbrido. Portanto, quanto mais exemplos são submetidos para o KNN, maior será o tempo de

Tabela 6.14: Resultado reduzido do teste estatístico Dunn para comparação das abordagens entre as bases. Comparações que tiveram diferenças significativas são mostradas em negrito. Fonte: O autor.

Comparações	Acurácia dos métodos	p-value
HIB(P25-N1)-C - HIB(P25-N75)-C	99,70 - 99,29	8.329375e-02
HIB(P25-N1)-C - HIB(P25-N75)-30	99,70 - 99,29	6.507520e-02
HIB(P25-N1)-C - HIB(P25-N75)-20	99,70 - 99,30	6.386860e-02
HIB(P25-N1)-C - HIB(P1-N1)-12	**99,70 - 97,54**	**1.234159e-03**
HIB(P25-N1)-C - HIB(P1-N75)-12	**99,70 - 96,38**	**2.433734e-06**
HIB(P25-N1)-C - HIB(P25-N1)-6	**99,70 - 97,11**	**1.045881e-04**
HIB(P25-N1)-C - HIB(P1-N75)-6	**99,70 - 95,56**	**1.755021e-07**
HIB(P25-N75)-C - HIB(P25-N1)-30	99,29 - 99,70	1.628114e-01
HIB(P25-N75)-C - HIB(P25-N75)-20	99,29 - 99,30	9.183473e-01
HIB(P25-N75)-C - HIB(P1-N75)-12	**99,29 - 96,38**	**3.818017e-03**
HIB(P25-N75)-C - HIB(P25-N1)-6	**99,29 - 97,11**	**4.696308e-02**
HIB(P25-N75)-C - HIB(P1-N75)-6	**99,29 - 95,56**	**7.158863e-04**
HIB(P25-N1)-30 - HIB(P25-N1)-20	99,70 - 99,69	8.166086e-01
HIB(P25-N1)-30 - HIB(P25-N75)-20	99,70 - 99,30	1.293856e-01
HIB(P25-N1)-30 - HIB(P1-N1)-12	**99,70 - 97,54**	**4.078249e-03**
HIB(P25-N1)-30 - HIB(P25-N1)-6	**99,70 - 97,11**	**4.051314e-04**
HIB(P25-N75)-30 - HIB(P25-N1)-20	99,29 - 99,69	2.224726e-01
HIB(P25-N75)-30 - HIB(P25-N75)-20	99,29 - 99,30	9.876386e-01
HIB(P25-N75)-30 - HIB(P1-N75)-12	**99,29 - 96,38**	**5.614805e-03**
HIB(P25-N75)-30 - HIB(P1-N75)-6	**99,29 - 95,56**	**1.064890e-03**
HIB(P25-N75)-20 - HIB(P1-N75)-12	**99,30 - 96,38**	**5.832999e-03**
HIB(P25-N75)-20 - HIB(P1-N75)-6	**99,30 - 95,56**	**1.110897e-03**

classificação.

Após análise dos resultados estatísticos, pode-se comprovar que em termos de acurácia média as abordagens híbridas aplicadas nas bases NSL-KDD completa, NSL-KDD com 30 e NSL-KDD com 20 atributos, são de fato superiores às abordagens híbridas aplicadas nas bases NSL-KDD com 12 e com 6 atributos.

Analisando apenas as abordagens híbridas aplicadas nas bases NSL-KDD completa, com 30 e com 20 atributos (as quais foram consideradas eficientes), observa-se que estas alcançaram taxas de acurácia média similares. O resultado estatístico apresentado na Tabela 6.14, ratifica que não existem diferenças significativas entre os resultados destas abordagens no critério acurácia.

Devido ao fato não haver diferenças estatisticamente significativas entre as abordagens híbridas aplicadas nas bases NSL-KDD completa, NSL-KDD com 20 e com 30 atributos, considerou-se o desempenho em termos de tempo de classificação, para identificar a melhor abordagem.

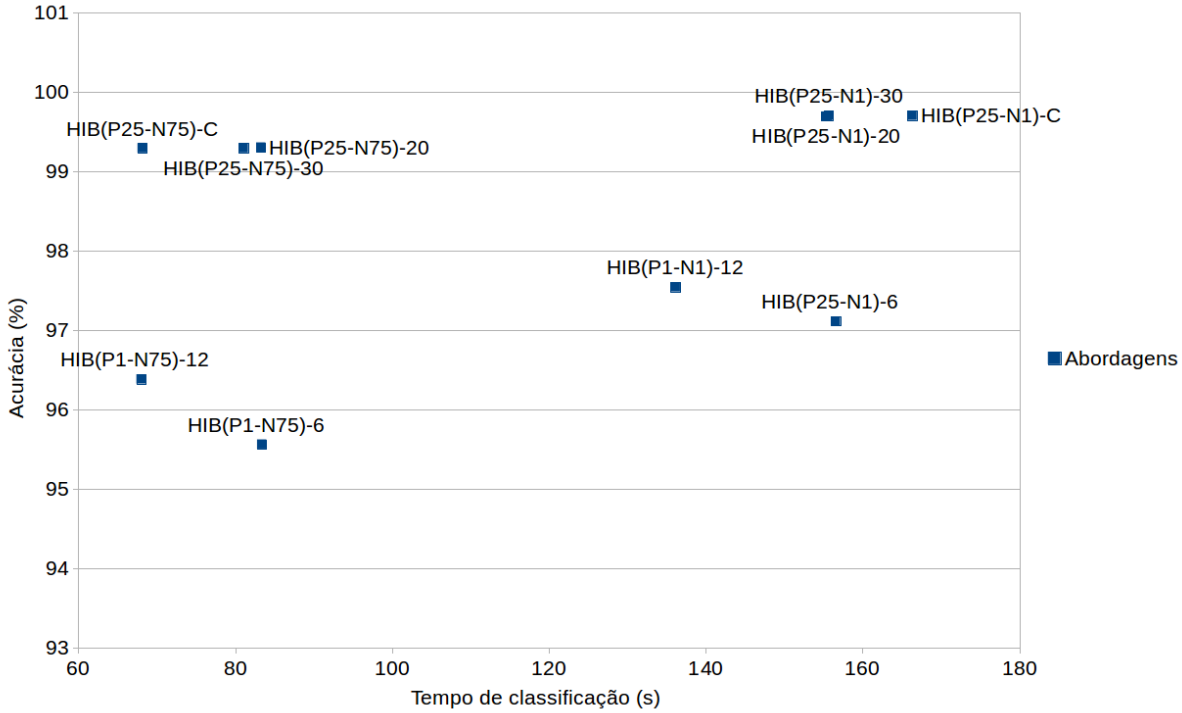


Figura 6.11: Gráfico dos resultados da aplicação dos métodos híbridos nas 5 bases. Fonte: O autor.

Conforme pode ser observado na Tabela 6.15, as abordagens HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20 submeteram aproximadamente 38961, 47555,2 e 48135,8 exemplos para o algoritmo KNN, respectivamente. Sendo estes os menores números dentre as abordagens levadas em consideração nesta comparação. Tendo em vista que o KNN é a técnica mais custosa utilizada no método Híbrido, quanto maior a quantidade de exemplos submetidos ao KNN, maior será o tempo de classificação. Por conseguinte, as abordagens HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20 apresentaram também os menores tempos de processamento.

Tabela 6.15: Resultados obtidos pelas abordagens híbridas nas 5 bases. Fonte: O autor.

Abordagens	**Acurácia (%)**	**Tempo de classificação (s)**	**Exemplos -> KNN**
HIB(P25-N1)-C	99,70	166,326	83794,7
HIB(P25-N75)-C	99,29	68,133	38961,5
HIB(P25-N1)-30	99,70	155,684	84243,9
HIB(P25-N75)-30	99,29	81,038	47555,2
HIB(P25-N1)-20	99,69	155,359	84005,9
HIB(P25-N75)-20	99,30	83,266	48135,8
HIB(P1-N1)-12	97,54	136,137	81362,2
HIB(P1-N75)-12	96,38	67,994	35713,9
HIB(P25-N1)-6	97,11	156,564	94576,2
HIB(P1-N75)-6	95,56	83,384	50898,8

Neste sentido foi também realizada uma avaliação estatística em relação ao tempo de classificação nas abordagens híbridas. Para isso, primeiramente foi realizada a avaliação da normalidade dos dados através do método Shapiro-Wilk (Shapiro & Wilk, 1965). O teste de normalidade apresentou *p-value* de $4.011e-06$, indicando que os resultados de tempo não seguem uma distribuição normal. Decorrente ao fato, para a comparação das médias entre as abordagens, foi utilizado o método não paramétrico para dados não pareados Kruskal-Wallis (Kruskal & Wallis, 1952). O resultado obtido foi um $p-value$ de $2.82e-09$ indicando diferenças estatisticamente significativas entre as abordagens híbridas.

Com o objetivo de encontrar quais abordagens se diferem estatisticamente, foi realizado o pós-teste Dunn (Everitt et al., 2001) com nível de significância de 95%. Os resultados desta análise são apresentados na Tabela 6.16.

Tabela 6.16: Resultado do teste estatístico Dunn para comparação das abordagens híbridas em termos de tempo de classificação. Comparações que tiveram diferenças significativas são mostradas em negrito. Fonte: O autor.

Comparações	**Tempos de classificações**	**p-value**
HIB(P25-N1)-C - **HIB(P25-N75)-C**	166,326 - **68,133**	**7.371971e-07**
HIB(P25-N1)-C - HIB(P25-N1)-30	166,326 - 155,684	4.270557e-01
HIB(P25-N1)-C - **HIB(P25-N75)-30**	166,326 - **81,038**	**7.985824e-05**
HIB(P25-N1)-C - HIB(P25-N1)-20	166,326 - 155,359	2.495415e-01
HIB(P25-N1)-C - **HIB(P25-N75)-20**	166,326 - **83,266**	**2.050591e-04**
HIB(P25-N75)-C - HIB(P25-N1)-30	**68,133** - 155,684	**3.871146e-05**
HIB(P25-N75)-C - HIB(P25-N75)-30	68,133 - 81,038	3.181036e-01
HIB(P25-N75)-C - HIB(P25-N1)-20	**68,133** - 155,359	**1.403681e-04**
HIB(P25-N75)-C - HIB(P25-N75)-20	68,133 - 83,266	2.113596e-01
HIB(P25-N1)-30 -**HIB(P25-N75)-30**	155,684 - **81,038**	**1.573693e-03**
HIB(P25-N1)-30 - HIB(P25-N1)-20	155,684 - 155,359	7.107067e-01
HIB(P25-N1)-30 - **HIB(P25-N75)-20**	155,684 - **83,266**	**4.351741e-03**
HIB(P25-N75)-30 - HIB(P25-N1)-20	**81,038** - 155,359	**5.347430e-03**
HIB(P25-N75)-30 - HIB(P25-N75)-20	81,038 - 83,266	7.392129e-01
HIB(P25-N1)-20 - **HIB(P25-N75)-20**	155,359 - **83,266**	**1.340087e-02**

De acordo com o pós-teste estatístico de Dunn (Tabela 6.16), não foram encontradas diferenças estatisticamente significativas entre as abordagens HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20. Por outro lado, tais métodos apresentaram diferença estatística em relação as demais abordagens híbridas. Deste modo, dentre as configurações híbridas avaliadas quantitativa e estatisticamente, os melhores desempenhos em termos de acurácia e tempo de classificação foram obtidas pelas abordagens híbridas HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20.

6.6 Considerações sobre o Método Híbrido em Relação a Trabalhos Similares

Mediante os dados apresentados nas seções anteriores, comprova-se que o método Híbrido proposto neste trabalho supera estatisticamente o desempenho da RNA em termos de acurácia em todos as cinco bases de dados. Também que destacar que as taxas de acurácia do método Híbrido foram estatisticamente equivalentes ao desempenho do algoritmo KNN. Por fim, e não menos importante, o método Híbrido reduziu o tempo de processamento em relação ao algoritmo KNN.

Após tais contribuições alcançadas pelo método proposto neste trabalho, retoma-se a análise da literatura (Seção 4.2) no intuito de promover breves comparações em relação a trabalhos que utilizaram técnicas distintas e a mesma base de dados.

No que concerne a acurácia, para facilitar os comparativos, retoma-se alguns indicadores, tais como as melhores taxas de acurácia, evidenciando-se as configurações utilizadas do método híbrido: 99,70% para base NSL-KDD completa, 99,70% para a base com 30 atributos, 99,69% para a base com 20 atributos, 97,54% para a base com 12 atributos e 97,11% para a base com 6 atributos.

Wang et al. (2010) desenvolveram o método **FC-ANN**, utilizando RNA e clusterização *fuzzy* para detectar intrusões. Os experimentos foram realizados sobre a base KDD CUP 99, da qual derivou-se a base NSL-KDD utilizada no presente trabalho. A abordagem FC-ANN obteve uma taxa de acurácia de 96,71%, superando os métodos que usou para comparação: RNA que obteve 96,65% de acurácia e *Naive Bayes* (Russell & Norvig, 2010) que obteve 96,11% de acurácia. Tais números indicam que todas as configurações do método Híbrido proposto neste trabalho superaram o método FC-ANN, assim como os citados classificadores RNA e *Naive Bayes*, no que se refere a acurácia.

Tsai & Lin (2010) propuseram uma abordagem híbrida (TANN) para detecção de intrusão utilizando clusterização e o algoritmo KNN. Para a experimentação também foi utilizada a base KDD CUP 99. Os resultados demonstraram que a taxa de acurácia da abordagem TANN foi de 99,01% e superou os métodos KNN e Support Vector Machine (SVM) (Mitchell, 1997) que obtiveram 93,87% e 94,98% de acurácia, respectivamente. O método Híbrido apresentado neste trabalho supera, no quesito acurácia, a abordagem TANN, quando consideradas as configurações com base completa, 30 atributos e 20 atributos. Cabe ressaltar que o método TANN foi realizado com a base completa, podendo-se confirmar que o método proposto neste trabalho obteve melhores índices de acurácia. Tendo-se em evidência os trabalhos utilizados pelos autores Tsai & Lin (2010) em sua comparação, KNN e SVM, todas as configurações híbridas do presente trabalho superaram tais métodos no parâmetro acurácia.

Já os autores Singh et al. (2015) conceberam um método de detecção de intrusão baseado

na rede neural Online Sequential Extreme Learning Machine (OS-ELM) (Liang et al., 2006). Os experimentos foram realizados com a base NSL-KDD, a mesma base utilizada no presente trabalho, e obteve-se uma acurácia de 98,66%, superior as abordagens que utilizaram para comparação (RNA com acurácia de 94,89% e *Naive Bayes* com acurácia de 87,29%). O método Híbrido definido neste trabalho supera, no quesito acurácia, o modelo de rede neural proposto pelos autores autores Singh et al. (2015), quando consideradas as configurações com base completa, 30 atributos e 20 atributos. A exemplo do caso anterior, cabe ressaltar a rede neural Online Sequential Extreme Learning Machine utilizou todos os atributos da base NSL-KDD. Tendo-se em evidência os trabalhos utilizados pelos autores Tsai & Lin (2010) em sua comparação, RNA e *Naive Bayes*, todas as configurações híbridas do presente trabalhos superaram tais métodos no parâmetro acurácia.

Retomando a linha de pesquisa desenvolvida em parceria entre os laboratórios LaPSeC da UNIOESTE e DMC-NS da UFSC, um importante trabalho foi a definição de uma abordagem de detecção de intrusão, utilizando monitoramento do fluxo de dados da rede, análise semântica e uma RNA *multilayer perceptron* (Lima, 2005). A acurácia alcançada por tal método foi 94,36%, sendo superada por todas as configurações do método Híbrido apresentado neste trabalho. Assim demonstra-se adicionalmente que o presente trabalho constitui um importante avanço dentro desta linha de pesquisa.

Neste mesmo sentido, o sistema de detecção de intrusão baseado em sistemas imunológicos artificiais e agentes móveis (Machado, 2005b) apresentou uma importante necessidade de melhoria, a utilização de métodos de detecção com maior acurácia na caixa de análise. Neste sentido, o método Híbrido proposto neste trabalho pode suprir esta deficiência , e assim, apresenta-se mais uma importante contribuição do presente trabalho para a linha de pesquisa na qual está inserida.

6.7 Considerações Finais

Neste capítulo foram apresentados e discutidos os experimentos e resultados obtidos.

No próximo capítulo apresenta-se as conclusões deste trabalho e serão apresentas sugestões para a realização de trabalhos futuros nesta linha de pesquisa.

Capítulo 7

Conclusão

Com a grande expansão da Internet e o crescimento das informações que são manipuladas em sistemas computacionais interligados pela rede, surgiram dificuldades para se manter a segurança dos sistemas. Os *Intrusion Detection System* (IDS) são mecanismos que tem por objetivo evitar que ocorram intrusões em redes de computadores.

As abordagens de detecção utilizando métodos híbridos de Inteligência Artificial, criadas pela combinação ou integração de múltiplas técnicas de classificação, vem alcançando resultados mais satisfatórios que a utilização das técnicas de maneira individual (Kumar et al., 2010). Este trabalho enquadra-se na linha de pesquisa Métodos de Inteligência Artificial aplicados na Segurança de Redes e resultou em um método Híbrido de detecção de intrusão combinando as técnicas de Redes Neurais Artificiais (RNA) e *k-Nearest Neighbors* (KNN).

O método Híbrido, assim como o delineamento experimental, foram desenvolvidos de acordo com as fases do processo de *Knowledge Discovery in Databases* (KDD) (Fayyad, 1996). A primeira etapa consistiu inicialmente na seleção da base de dados pública NSL-KDD (Tavallaee et al., 2009), a qual é fortemente conceituada e aplicada na área de detecção de intrusão. A partir da escolha da base, foram aplicadas técnicas de pré-processamento, especificamente a seleção de atributos, com a finalidade de identificar a contribuição de cada atributo da base para o processo de classificação. Como resultado deste processo, criaram-se quatro novas sub-bases derivadas, sendo três compostas de 30, 20 e 12 atributos respectivamente, tomando como fator de escolha a taxa de ganho de informação individual de cada atributo. Os atributos da quarta sub-base foram selecionados por serem facilmente obtidos no processo de captura de pacotes em rede, com a intenção de criar um modelo que fosse mais facilmente utilizado em tempo real.

Dentre as configurações utilizadas do método híbrido, as melhores taxas de acurácia foram: 99,70% para base NSL-KDD completa, 99,70% para a base com 30 atributos, 99,69% para a base com 20 atributos, 97,54% para a base com 12 atributos e 97,11% para a base com 6 atributos. Tais taxas de acurácia foram estatisticamente superiores em relação a RNA e foram equivalentes ao KNN.

No que concerne ao tempo, dentre as configurações do método híbrido os melhores re-

sultados foram: 68,13 segundos para a base completa, 81,03 segundos para a base com 30 atributos, 83,27 segundos para a base 20 atributos, 67,99 segundos para a base com 12 atributos e 83,38 segundos para a base com 6 atributos. Isso demonstra que o método híbrido apresentou uma melhoria significativa no tempo de classificação quando comparado com o método KNN.

Após a análise dos resultados, conclui-se que o método Híbrido proposto neste trabalho supera estatisticamente o desempenho da RNA em termos de acurácia em todos as cinco bases de dados. Além disso, as taxas de acurácia do método híbrido foram equivalentes ao desempenho do algoritmo KNN, o que foi ratificado nos testes estatísticos realizados. Também cabe destacar que o método Híbrido reduziu o tempo de processamento em relação ao algoritmo KNN.

Com relação aos métodos similares abordados na Seção 4.2 (Estado da Arte), ficou demonstrado que o método Híbrido definido neste trabalho superou todos no quesito acurácia, considerando-se os métodos passíveis de comparação (Seção 6.6). Este trabalho também apresenta importante contribuição para a evolução da linha de pesquisa em segurança computacional desenvolvida em parceria entre os laboratórios LaPSeC da UNIOESTE e DMC-NS da UFSC.

Por fim, tais resultados confirmam a hipótese elaborada neste trabalho, de que que a utilização de um método híbrido, composto por uma RNA e pelo algoritmo KNN, alcançaria desempenho superior em relação às abordagens individuais destas técnicas.

Cabe ressaltar ainda, que dentre as configurações híbridas avaliadas quantitativa e estatisticamente, os melhores desempenhos em termos de acurácia e tempo de classificação foram obtidos pelas abordagens híbridas HIB(P25-N75)-C, HIB(P25-N75)-30 e HIB(P25-N75)-20.

A partir dos resultados e contribuições alcançadas, novos trabalhos podem ser vislumbrados, dentre os quais citam-se:

- Comparar o método Híbrido proposto neste trabalho com outros técnicas tradicionais de Inteligência Artificial e com técnicas híbridas levantadas na bibliografia;
- Realizar experimentos em ambiente e tempo real, para verificar o desempenho do método Híbrido em relação as abordagens individuais de RNA e KNN;
- Avaliar a redução de dimensionalidade, através da seleção de atributos realizada neste trabalho para detecção de intrusão em soluções de Internet of Things (IoT), dado que tais soluções exigem métodos de detecção que se adaptem as suas limitações de recursos;
- Propor novas abordagens híbridas, especificamente utilizando métodos *Fuzzy*, técnicas evolutivas e algoritmos neuro-evolutivos.

Referências Bibliográficas

Aburomman, A. A. & Reaz, M. B. I. (2017). A novel weighted support vector machines multi-class classifier based on differential evolution for intrusion detection systems, *Information Sciences* .

Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-based learning algorithms, *Machine learning* **6**(1): 37–66.

Asaka, M., Okazawa, S., Taguchi, A. & Goto, S. (1999). A method of tracing intruders by use of mobile agents, *INET'99 Proceedings*.

Ashfaq, R. A. R., Wang, X.-Z., Huang, J. Z., Abbas, H. & He, Y.-L. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system, *Information Sciences* **378**: 484–497.

Azad, H. B., Mekhilef, S. & Ganapathy, V. G. (2014). Long-term wind speed forecasting and general pattern recognition using neural networks, *IEEE Transactions on Sustainable Energy* **5**(2): 546–553.

Bace, R. & Mell, P. (2001). Nist special publication on intrusion detection systems, *Technical report*, BOOZ-ALLEN AND HAMILTON INC MCLEAN VA.

Baluja, S. (1994). Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning, *Technical report*, Carnegie-Mellon University Pittsburgh, Department Of Computer Science.

Barbosa, A. S. & Moraes, L. F. M. (2000). Sistemas de detecção de intrusão, *Seminários Ravel - CPS760: Laboratório de Redes de Alta Velocidade, UFRJ* .

Bernardes, M. C. (1999). Avaliação do uso de agentes móveis em segurança computacional, *São Carlos* .

Blake, C. & Merz, C. J. (1998).

Bodnar, Lucas, M. (2013). Atualização de um sistema de detecção de intrusão para uma plataforma atual em jade. Monografia (Bacharel em Sistemas de Informação), UFSC (Universidade Federal de Santa Catarina), Florianópolis, Brasil.

Boukerche, A., Machado, R. B., Jucá, K. R., Sobral, J. B. M. & Notare, M. S. (2007). An agent based and biological inspired real-time intrusion detection and security model for computer network operations, *Computer Communications* **30**(13): 2649–2660.

Campello, R. S. & Weber, R. F. (2001). Sistemas de detecçao de intrusao, *Minicurso procedente do 19º Simpósio Brasileiro de Redes de Computadores* .

Chen, M.-H., Chang, P.-C. & Wu, J.-L. (2016). A population-based incremental learning approach with artificial immune system for network intrusion detection, *Engineering Applications of Artificial Intelligence* **51**: 171–181.

Chua, L. O. & Yang, L. (1988). Cellular neural networks: Applications, *IEEE Transactions on circuits and systems* **35**(10): 1273–1290.

Costa, R, M. (2014). Implantação de um sistema de detecção de intrusão baseado em sistema imunológico artificial para um ambiente de nuvem computacional. Monografia (Bacharel em Sistemas de Informação), UFSC (Universidade Federal de Santa Catarina), Florianópolis, Brasil.

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification, *IEEE transactions on information theory* **13**(1): 21–27.

Crosbie, M. & Spafford, G. (1995). Defending a computer system using autonomous agents, *West Lafayetee: Dept. of Computer Sciences* pp. 12–16.

Dalton, J. & Deshmane, A. (1991). Artificial neural networks, *IEEE Potentials* **10**(2): 33–36.

Danh, N. T., Phien, H. N. & Gupta, A. D. (1999). Neural network models for river flow forecasting, *water SA* **25**(1): 33–39.

Dasarathy, B. V. (1994). Minimal consistent set (mcs) identification for optimal nearest neighbor decision systems design, *IEEE Transactions on Systems, Man, and Cybernetics* **24**(3): 511–517.

De Castro, L. N. & Timmis, J. (2002). *Artificial immune systems: a new computational intelligence approach*, Springer Science & Business Media.

Denning, D. E. (1987). An intrusion-detection model, *IEEE Transactions on software engineering* (2): 222–232.

Dhillon, H. S., Huang, H. & Viswanathan, H. (2017). Wide-area wireless communication challenges for the internet of things, *IEEE Communications Magazine* **55**(2): 168–174.

Eesa, A. S., Brifcani, A. M. A. & Orman, Z. (2013). Cuttlefish algorithm–a novel bio-inspired optimization algorithm, *International Journal of Scientific and Engineering Research* **4**(9): 1978–86.

Eesa, A. S., Orman, Z. & Brifcani, A. M. A. (2015). A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems, *Expert Systems with Applications* **42**(5): 2670–2679.

Encryption, A. (2016). Aes encryption.
Disponível em: *http://aesencryption.net/*

Engelbrecht, A. P. (2007). *Computational intelligence: an introduction*, John Wiley & Sons.

Ester, M., Kriegel, H.-P., Sander, J. & Xu, X. (1996). Density-based spatial clustering of applications with noise, *Int. Conf. Knowledge Discovery and Data Mining*, Vol. 240.

Everitt, B. S., Dunn, G. et al. (2001). *Applied multivariate data analysis*, Vol. 2, Wiley Online Library.

Fausett, L. & Fausett, L. (1994). *Fundamentals of neural networks: architectures, algorithms, and applications*, number 006.3, Prentice-Hall,.

Fayyad, U. M. (1996). Data mining and knowledge discovery: Making sense out of data, *IEEE Expert: Intelligent Systems and Their Applications* **11**(5): 20–25.

Fayyad, U., Piatetsky-Shapiro, G. & Smyth, P. (1996). From data mining to knowledge discovery in databases, *AI magazine* **17**(3): 37.

Feistel, H. (1973). Cryptography and computer privacy, *Scientific american* **228**: 15–23.

Ferrero, C. A. (2009). *Algoritmo kNN para previsão de dados temporais: funções de previsão e critérios de seleção de vizinhos próximos aplicados a variáveis ambientais em limnologia*, PhD thesis, Universidade de São Paulo.

Fisher, R. A. (1992). Statistical methods for research workers, *Breakthroughs in Statistics*, Springer, pp. 66–70.

Goodfellow, I., Bengio, Y., Courville, A. & Bengio, Y. (2016). *Deep learning*, Vol. 1, MIT press Cambridge.

Goodrich, M. T. & Tamassia, R. (2013). *Introdução à segurança de computadores*, Bookman, Porto Alegre.

Govindarajan, M. & Chandrasekaran, R. (2011). Intrusion detection using neural based hybrid classification methods, *Computer networks* **55**(8): 1662–1671.

Gunupudi, R. K., Nimmala, M., Gugulothu, N. & Gali, S. R. (2017). Clapp: A self constructing feature clustering approach for anomaly detection, *Future Generation Computer Systems* **74**: 417–429.

Guo, C., Zhou, Y., Ping, Y., Zhang, Z., Liu, G. & Yang, Y. (2014). A distance sum-based hybrid method for intrusion detection, *Applied intelligence* **40**(1): 178–188.

Hall, M. A. (1999). *Correlation-based Feature Selection for Machine Learning*, PhD thesis, The University of Waikato.

Hattori, K. & Takahashi, M. (1999). A new nearest-neighbor rule in the pattern classification problem, *Pattern recognition* **32**(3): 425–432.

Haykin, S. S. (2001). *Redes neurais: Princípios e Práticas*, Bookman.

Hoque, N., Kashyap, H. & Bhattacharyya, D. (2017). Real-time ddos attack detection using fpga, *Computer Communications* .

Ilgun, K. (1993). Ustat: A real-time intrusion detection system for unix, *Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on*, IEEE, pp. 16–28.

Karegowda, A. G., Manjunath, A. & Jayaram, M. (2010). Comparative study of attribute selection using gain ratio and correlation based feature selection, *International Journal of Information Technology and Knowledge Management* **2**(2): 271–277.

Kim, G., Lee, S. & Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection, *Expert Systems with Applications* **41**(4): 1690–1700.

Kovács, Z. L. (2002). *Redes neurais artificiais*, Editora Livraria da Fisica.

Kristjanpoller, W. & Minutolo, M. C. (2015). Gold price volatility: A forecasting approach using the artificial neural network–garch model, *Expert Systems with Applications* **42**(20): 7245–7251.

Kruskal, W. H. & Wallis, W. A. (1952). Use of ranks in one-criterion variance analysis, **47**(260): 583–621.

Kumar, G., Kumar, K. & Sachdeva, M. (2010). The use of artificial intelligence based techniques for intrusion detection: a review, *Artificial Intelligence Review* **34**(4): 369–387.

Kuncheva, L. I. & Bezdek, J. C. (1998). Nearest prototype classification: Clustering, genetic algorithms, or random search?, *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* **28**(1): 160–164.

Leite, A. & Girardi, R. (2017). A hybrid and learning agent architecture for network intrusion detection, *Journal of Systems and Software* .

Lent, R. (2004). *Cem bilhões de neurônios: conceitos fundamentais de neurociência*, Atheneu.

Liang, N.-Y., Huang, G.-B., Saratchandran, P. & Sundararajan, N. (2006). A fast and accurate online sequential learning algorithm for feedforward networks, *IEEE Transactions on neural networks* **17**(6): 1411–1423.

Lima, I. V. M. d. (2005). Uma abordagem simplificada de detecção de intrusão baseada em redes neurais artificiais.

Lin, W.-C., Ke, S.-W. & Tsai, C.-F. (2015). Cann: An intrusion detection system based on combining cluster centers and nearest neighbors, *Knowledge-based systems* **78**: 13–21.

Lippmann, R. (1987). An introduction to computing with neural nets, *IEEE Assp magazine* **4**(2): 4–22.

Ma, T., Wang, F., Cheng, J., Yu, Y. & Chen, X. (2016). A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks, *Sensors* **16**(10): 1701.

Machado, A. B. M. (2005a). *Neuroanatomia funcional*, Atheneu.

Machado, R. B. (2005b). *Uma abordagem de detecção de intrusão baseada em sistemas imunológicos artificiais e agentes móveis*, Master's thesis, UNIVERSIDADE FEDERAL DE SANTA CATARINA.

Manzoor, I., Kumar, N. et al. (2017). A feature reduced intrusion detection system using ann classifier, *Expert Systems with Applications* **88**: 249–257.

McHugh, J. (2000). Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory, *ACM Transactions on Information and System Security (TISSEC)* **3**(4): 262–294.

Minsky, M. & Papert, S. (1969). Perceptrons: Anlntroduction to computational geometry.

Miorandi, D., Sicari, S., De Pellegrini, F. & Chlamtac, I. (2012). Internet of things: Vision, applications and research challenges, *Ad Hoc Networks* **10**(7): 1497–1516.

Mitchell, T. M. (1997). Machine learning, *McGraw Hill Science/Engineering/Math* p. 432.

Mukherjee, B., Heberlein, L. T. & Levitt, K. N. (1994). Network intrusion detection, *IEEE network* **8**(3): 26–41.

Ng, A. Y., Jordan, M. I. & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm, *Advances in neural information processing systems*, pp. 849–856.

Ni, J., Zhang, K., Lin, X. & Shen, X. (2017). Securing fog computing for internet of things applications: Challenges and solutions, *IEEE Communications Surveys & Tutorials* .

Northcutt, S., Cooper, M., Fearnow, M. & Frederick, K. (2001). *Intrusion Signatures and Analysis*, New Riders, New Jersey.

Norton, M. J. (1999). Knowledge discovery in databases, *Library Trends* **48**(1): 9.

Osanaiye, O., Cai, H., Choo, K.-K. R., Dehghantanha, A., Xu, Z. & Dlodlo, M. (2016). Ensemble-based multi-filter feature selection method for ddos detection in cloud computing, *EURASIP Journal on Wireless Communications and Networking* **2016**(1): 130.

Panda, M., Abraham, A., Das, S. & Patra, M. R. (2011). Network intrusion detection system: A machine learning approach, *Intelligent Decision Technologies* **5**(4): 347–356.

Porras, P. A. & Neumann, P. G. (1997). Emerald: Event monitoring enabling response to anomalous live disturbances, *Proceedings of the 20th national information systems security conference*, pp. 353–365.

Preneel, B. (1999). The state of cryptographic hash functions, *Lectures on Data Security*, Springer, pp. 158–182.

Quinlan, J. R. (1986). Induction of decision trees, *Machine learning* **1**(1): 81–106.

Quinlan, J. R. (1993). C4.5: Programming for machine learning, *Morgan Kauffmann* **38**.

Quinlan, J. R. (1996). Learning decision tree classifiers, *ACM Computing Surveys (CSUR)* **28**(1): 71–72.

Quinlan, J. R. (2014). *C4. 5: programs for machine learning*, Elsevier.

Raman, M. G., Somu, N., Kirthivasan, K., Liscano, R. & Sriram, V. S. (2017). An efficient intrusion detection system based on hypergraph-genetic algorithm for parameter optimization and feature selection in support vector machine, *Knowledge-Based Systems* **134**: 1–12.

Raman, M. G., Somu, N., Kirthivasan, K. & Sriram, V. S. (2017). A hypergraph and arithmetic residue-based probabilistic neural network for classification in intrusion detection systems, *Neural Networks* **92**: 89–97.

Rivest, R. (1992). The md5 message-digest algorithm.

Rivest, R. L., Shamir, A. & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems, *Communications of the ACM* **21**(2): 120–126.

Rojas, R. (2013). *Neural networks: a systematic introduction*, Springer Science & Business Media.

Rosa, W. A. (2017). Avaliação de métodos de inteligência computacional para detecção de intrusão. Monografia (Bacharel em Ciência da Computação), UNIOESTE (Universidade Estadual do Oeste do Paraná), Foz do Iguaçu, Brazil.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain., *Psychological review* **65**(6): 386.

Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1985). Learning internal representations by error propagation, *Technical report*, California Univ San Diego La Jolla Inst for Cognitive Science.

Russell, S. J. & Norvig, P. (2010). *Artificial intelligence: A Modern Approach*, Pearson Education.

Schurgot, M. R., Shinberg, D. A. & Greenwald, L. G. (2015). Experiments with security and privacy in iot networks, *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, IEEE, pp. 1–6.

Shapiro, S. S. & Wilk, M. B. (1965). An analysis of variance test for normality (complete samples), *Biometrika* **52**(3/4): 591–611.

Shon, T. & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection, *Information Sciences* **177**(18): 3799–3821.

Short, R. & Fukunaga, K. (1981). The optimal distance measure for nearest neighbor classification, *IEEE transactions on Information Theory* **27**(5): 622–627.

Silva, S. F., Anjos, C. A. R., Cavalcanti, R. N. & dos Santos Celeghini, R. M. (2015). Evaluation of extra virgin olive oil stability by artificial neural network, *Food chemistry* **179**: 35–43.

Singh, R., Kumar, H. & Singla, R. (2015). An intrusion detection system using network traffic profiling and online sequential extreme learning machine, *Expert Systems with Applications* **42**(22): 8609–8624.

Specht, D. F. (1990). Probabilistic neural networks, *Neural networks* **3**(1): 109–118.

Squire, L., Berg, D., Bloom, F. E., Du Lac, S., Ghosh, A. & Spitzer, N. C. (2012). *Fundamental neuroscience*, Academic Press.

Staniford-Chen, S., Tung, B., Schnackenberg, D. et al. (1998). The common intrusion detection framework (cidf), *Proceedings of the information survivability workshop*.

Steinwart, I. & Christmann, A. (2008). *Support vector machines*, Springer Science & Business Media.

Stolfo, J., Fan, W., Lee, W., Prodromidis, A. & Chan, P. K. (2000). Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection, *Results from the JAM Project by Salvatore* pp. 1–15.

Tanenbaum, A. S. et al. (2003). Computer networks, 4-th edition, *ed: Prentice Hall* .

Tavallaee, M., Bagheri, E., Lu, W. & Ghorbani, A. A. (2009). A detailed analysis of the kdd cup 99 data set, *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*, IEEE, pp. 1–6.

Tombini, E., Debar, H., Mé, L. & Ducassé, M. (2004). A serial combination of anomaly and misuse idses applied to http traffic, *Computer Security Applications Conference, 2004. 20th Annual*, IEEE, pp. 428–437.

Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y. & Lin, W.-Y. (2009). Intrusion detection by machine learning: A review, *Expert Systems with Applications* **36**(10): 11994–12000.

Tsai, C.-F. & Lin, C.-Y. (2010). A triangle area based nearest neighbors approach to intrusion detection, *Pattern recognition* **43**(1): 222–229.

Tsai, C.-W., Lai, C.-F. & Vasilakos, A. V. (2014). Future internet of things: open issues and challenges, *Wireless Networks* **20**(8): 2201–2217.

Vieira, K., Schulter, A., Westphall, C. & Westphall, C. (2010). Intrusion detection for grid and cloud computing, *IEEE IT ProfessionalV i e i r a* **12**(4): 38–43.

Wang, G., Hao, J., Ma, J. & Huang, L. (2010). A new approach to intrusion detection using artificial neural networks and fuzzy clustering, *Expert Systems with Applications* **37**(9): 6225–6232.

Wang, X., Zhang, C. & Zheng, K. (2016). Intrusion detection algorithm based on density, cluster centers, and nearest neighbors, *China Communications* **13**(7): 24–31.

Wood, M. & Erlinger, M. A. (2007). Intrusion detection message exchange requirements.

Xuan, Y., Shin, I., Thai, M. T. & Znati, T. (2010). Detecting application denial-of-service attacks: A group-testing-based approach, *IEEE Transactions on parallel and distributed systems* **21**(8): 1203–1216.

Zodik, G. (2015). Future technologies supporting the convergence of mobile, wearables, and iot, *Mobile Software Engineering and Systems (MOBILESoft), 2015 2nd ACM International Conference on*, IEEE, pp. 129–130.

Zurada, J. M. (1992). *Introduction to artificial neural systems*, Vol. 8, West St. Paul.

Apêndice A

Resultados dos Experimentos

Tabela A.1: Matrizes de confusão obtidas pela execução dos experimentos nas bases NSL-KDD completa.

Configuração		**Predição**	
RNA-C		**Intrusivo**	**Não Intrusivo**
	Intrusivo	54977,2 (VP)	3649,9 (FN)
	Não Intrusivo	977,6 (FP)	66365,3 (VN)
KNN-C		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58440,3 (VP)	186,7 (FN)
	Não Intrusivo	191,1 (FP)	67151,9 (VN)
HIB(P25-N1)-C		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58445,9 (VP)	181,3 (FN)
	Não Intrusivo	195,6 (FP)	67147,2 (VN)
HIB(P25-N75)-C		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	57928,3 (VP)	697 (FN)
	Não Intrusivo	202,1 (FP)	67142,6 (VN)

Tabela A.2: Matrizes de confusão obtidas pela execução dos experimentos na base NSL-KDD com 30 atributos.

Configuração		**Predição**	
RNA-30		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	56677,4 (VP)	1949,6 (FN)
	Não Intrusivo	1258,9 (FP)	66084,1 (VN)
KNN-30		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58439,7 (VP)	187,3 (FN)
	Não Intrusivo	193 (FP)	67150 (VN)
HIB(P25-N1)-30		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58442,9 (VP)	184,1 (FN)
	Não Intrusivo	199 (FP)	67144 (VN)
HIB(P25-N75)-30		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	57910,6 (VP)	716,4 (FN)
	Não Intrusivo	170,8 (FP)	67172,2 (VN)

Tabela A.3: Matrizes de confusão obtidas pela execução dos experimentos na base NSL-KDD com 20 atributos.

Configuração		**Predição**	
RNA-20		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	56077,9 (VP)	2549,2 (FN)
	Não Intrusivo	1139,6 (FP)	66203,3 (VN)
KNN-20		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58437,1 (VP)	189,9 (FN)
	Não Intrusivo	200,5 (FP)	67142,5 (VN)
HIB(P25-N1)-20		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58442,9 (VP)	184,1 (FN)
	Não Intrusivo	206,8 (FP)	67136,2 (VN)
HIB(P25-N75)-20		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	57937 (VP)	690 (FN)
	Não Intrusivo	197,5 (FP)	67145,5 (VN)

Tabela A.4: Matrizes de confusão obtidas pela execução dos experimentos nas bases NSL-KDD com 12 atributos.

Configuração		**Predição**	
RNA-12		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	45399,8 (VP)	13228 (FN)
	Não Intrusivo	259,7 (FP)	67082,5 (VN)
KNN-12		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	57941,2 (VP)	685,8 (FN)
	Não Intrusivo	3007 (FP)	64336 (VN)
HIB(P1-N1)-12		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58415,9 (VP)	211,1 (FN)
	Não Intrusivo	3141 (FP)	64202 (VN)
HIB(P1-N75)-12		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	56715,4 (VP)	1912,8 (FN)
	Não Intrusivo	2642,5 (FP)	64699,3 (VN)

Tabela A.5: Matrizes de confusão obtidas pela execução dos experimentos na base NSL-KDD 6 atributos.

Configuração		**Predição**	
RNA-6		**Intrusivo**	**Não Intrusivo**
	Intrusivo	46457,3 (VP)	12170 (FN)
	Não Intrusivo	1162,4 (FP)	66180,3 (VN)
KNN-6		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58020,9 (VP)	606,1 (FN)
	Não Intrusivo	3249,9 (FP)	64093,1 (VN)
HIB(P25-N1)-6		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	58255,2 (VP)	371,8 (FN)
	Não Intrusivo	3267,8 (FP)	64075,2 (VN)
HIB(P1-N75)-6		**Não Intrusivo**	**Não Intrusivo**
	Intrusivo	56343,1 (VP)	2283,8 (FN)
	Não Intrusivo	3309,5 (FP)	64033,6 (VN)

Apêndice B

Resultados do Teste Estatístico

Tabela B.1: Resultado do teste estatístico Dunn para comparação entre grupos. Comparações que tiveram diferenças significativas são mostradas em negrito.

Comparações	Acurácia das Configurações	p-value
RNA-C - **KNN-C**	**96,33 - 99,70**	**1.537779e-04**
RNA-C - **HIB(P25-N1)-C**	**96,33 - 99,70**	**1.968701e-04**
RNA-C - HIB(P25-N75)-C	96,33 - 99,29	6.539991e-02
RNA-C - RNA-30	96,33 - 97,45	7.822897e-01
RNA-C - **KNN-30**	**96,33 - 99,70**	**3.064512e-04**
RNA-C - **HIB(P25-N1)-30**	**96,33 - 99,70**	**7.187291e-04**
RNA-C - HIB(P25-N75)-30	96,33 - 99,29	8.367584e-02
RNA-C - RNA-20	96,33 - 97,07	8.584615e-01
RNA-C - **KNN-20**	**96,33 - 99,69**	**2.352914e-03**
RNA-C - **HIB(P25-N1)-20**	**96,33 - 99,69**	**1.961901e-03**
RNA-C - HIB(P25-N75)-20	96,33 - 99,30	8.521012e-02
RNA-C - RNA-12	**96,33 - 89,29**	**4.330952e-02**
RNA-C - KNN-12	96,33 - 97,07	7.890994e-01
RNA-C - HIB(P1-N1)-12	96,33 - 97,54	6.792517e-01
RNA-C - HIB(P1-N75)-12	96,33 - 96,38	3.635884e-01
RNA-C - RNA-6	**96,33 - 89,42**	**3.541810e-02**
RNA-C - KNN-6	96,33 - 96,94	6.231746e-01
RNA-C - HIB(P25-N1)-6	96,33 - 97,11	9.108678e-01
RNA-C - HIB(P1-N75)-6	96,33 - 95,56	1.621941e-01
KNN-C - HIB(P25-N1)-C	99,70 - 99,70	9.513481e-01
KNN-C - HIB(P25-N75)-C	99,70 - 99,29	7.392223e-02
KNN-C - RNA-30	**99,70 - 97,45**	**6.186246e-04**
KNN-C - KNN-30	99,70 - 99,70	8.934611e-01
KNN-C - HIB(P25-N1)-30	99,70 - 99,70	7.482532e-01
KNN-C - HIB(P25-N75)-30	99,70 - 99,29	5.808891e-02

KNN-C - RNA-20	**99,70 - 97,07**	**3.850516e-04**
KNN-C - KNN-20	99,70 - 99,69	5.269189e-01
KNN-C - HIB(P25-N1)-20	99,70 - 99,69	5.596859e-01
KNN-C - HIB(P25-N75)-20	99,70 - 99,30	5.643573e-02
KNN-C - RNA-12	**99,70 - 89,29**	**6.432771e-09**
KNN-C - KNN-12	**99,70 - 97,07**	**4.474050e-05**
KNN-C - HIB(P1-N1)-12	**99,70 - 97,54**	**1.013266e-03**
KNN-C - HIB(P1-N75)-12	**99,70 - 96,38**	**1.733171e-06**
KNN-C - RNA-6	**99,70 - 89,42**	**1.080499e-08**
KNN-C - KNN-6	**99,70 - 96,94**	**1.329663e-05**
KNN-C - HIB(P25-N1)-6	**99,70 - 97,11**	**8.136863e-05**
KNN-C - HIB(P1-N75)-6	**99,70 - 95,56**	**1.477774e-07**
HIB(P25-N1)-C - HIB(P25-N75)-C	99,70 - 99,29	8.329375e-02
HIB(P25-N1)-C - RNA-30	**99,70 - 97,45**	**7.399082e-04**
HIB(P25-N1)-C - KNN-30	99,70 - 99,70	9.238750e-01
HIB(P25-N1)-C - HIB(P25-N1)-30	99,70 - 99,70	7.861258e-01
HIB(P25-N1)-C - HIB(P25-N75)-30	99,70 - 99,29	6.507520e-02
HIB(P25-N1)-C - RNA-20	**99,70 - 97,07**	**4.993283e-04**
HIB(P25-N1)-C - KNN-20	99,70 - 99,69	5.668134e-01
HIB(P25-N1)-C - HIB(P25-N1)-20	99,70 - 99,69	5.928215e-01
HIB(P25-N1)-C - HIB(P25-N75)-20	99,70 - 99,30	6.386860e-02
HIB(P25-N1)-C - RNA-12	**99,70 - 89,29**	**7.828094e-09**
HIB(P25-N1)-C - KNN-12	**99,70 - 97,07**	**5.664754e-05**
HIB(P25-N1)-C - HIB(P1-N1)-12	**99,70 - 97,54**	**1.234159e-03**
HIB(P25-N1)-C - HIB(P1-N75)-12	**99,70 - 96,38**	**2.433734e-06**
HIB(P25-N1)-C - RNA-6	**99,70 - 89,42**	**8.821000e-09**
HIB(P25-N1)-C - KNN-6	**99,70 - 96,94**	**1.834956e-05**
HIB(P25-N1)-C - HIB(P25-N1)-6	**99,70 - 97,11**	**1.045881e-04**
HIB(P25-N1)-C - HIB(P1-N75)-6	**99,70 - 95,56**	**1.755021e-07**
HIB(P25-N75)-C - RNA-30	99,29 - 97,45	1.347803e-01
HIB(P25-N75)-C - KNN-30	99,29 - 99,70	1.059158e-01
HIB(P25-N75)-C - HIB(P25-N1)-30	99,29 - 99,70	1.628114e-01
HIB(P25-N75)-C - HIB(P25-N75)-30	99,29 - 99,29	9.209956e-01
HIB(P25-N75)-C - RNA-20	99,29 - 97,07	1.054523e-01
HIB(P25-N75)-C - KNN-20	99,29 - 99,69	2.918853e-01
HIB(P25-N75)-C - HIB(P25-N1)-20	99,29 - 99,69	2.694350e-01
HIB(P25-N75)-C - HIB(P25-N75)-20	99,29 - 99,30	9.183473e-01
HIB(P25-N75)-C - RNA-12	**99,29 - 89,29**	**6.173742e-05**

HIB(P25-N75)-C - KNN-12	**99,29 - 97,07**	**2.891834e-02**
HIB(P25-N75)-C - HIB(P1-N1)-12	99,29 - 97,54	1.752725e-01
HIB(P25-N75)-C - HIB(P1-N75)-12	**99,29 - 96,38**	**3.818017e-03**
HIB(P25-N75)-C - RNA-6	**99,29 - 89,42**	**4.720394e-05**
HIB(P25-N75)-C - KNN-6	**99,29 - 96,94**	**1.388762e-02**
HIB(P25-N75)-C - HIB(P25-N1)-6	**99,29 - 97,11**	**4.696308e-02**
HIB(P25-N75)-C - HIB(P1-N75)-6	**99,29 - 95,56**	**7.158863e-04**
RNA-30 - **KNN-30**	**97,45 - 99,70**	**1.100504e-03**
RNA-30 - **HIB(P25-N1)-30**	**97,45 - 99,70**	**2.400526e-03**
RNA-30 - HIB(P25-N75)-30	97,45 - 99,29	1.679812e-01
RNA-30 - RNA-20	97,45 - 97,07	9.244190e-01
RNA-30 - **KNN-20**	**97,45 - 99,69**	**7.758876e-03**
RNA-30 - **HIB(P25-N1)-20**	**97,45 - 99,69**	**6.585279e-03**
RNA-30 - HIB(P25-N75)-20	97,45 - 99,30	1.719317e-01
RNA-30 - RNA-12	**97,45 - 89,29**	**1.616613e-02**
RNA-30 - KNN-12	97,45 - 97,07	5.617987e-01
RNA-30 - HIB(P1-N1)-12	97,45 - 97,54	9.122309e-01
RNA-30 - HIB(P1-N75)-12	97,45 - 96,38	2.062215e-01
RNA-30 - RNA-6	**97,45 - 89,42**	**1.270952e-02**
RNA-30 - KNN-6	97,45 - 96,94	3.992453e-01
RNA-30 - HIB(P25-N1)-6	97,45 - 97,1	6.772188e-01
RNA-30 - HIB(P1-N75)-6	97,45 - 95,56	8.107170e-02
KNN-30 - HIB(P25-N1)-30	99,70 - 99,70	8.616343e-01
KNN-30 - HIB(P25-N75)-30	99,70 - 99,29	1.315064e-01
KNN-30 - RNA-20	**99,70 - 97,07**	**1.575894e-03**
KNN-30 - KNN-20	99,70 - 99,69	6.318777e-01
KNN-30 - HIB(P25-N1)-20	99,70 - 99,69	6.705464e-01
KNN-30 - HIB(P25-N75)-20	99,70 - 99,30	8.068422e-02
KNN-30 - RNA-12	**99,70 - 89,29**	**1.136519e-08**
KNN-30 - KNN-12	**99,70 - 97,07**	**8.151618e-05**
KNN-30 - HIB(P1-N1)-12	**99,70 - 97,54**	**1.842492e-03**
KNN-30 - HIB(P1-N75)-12	**99,70 - 96,38**	**4.354593e-06**
KNN-30 - RNA-6	**99,70 - 89,42**	**7.763117e-09**
KNN-30 - KNN-6	**99,70 - 96,94**	**3.158047e-05**
KNN-30 - HIB(P25-N1)-6	**99,70 - 97,11**	**1.688610e-04**
KNN-30 - HIB(P1-N75)-6	**99,70 - 95,56**	**3.054660e-07**
HIB(P25-N1)-30 - HIB(P25-N75)-30	99,70 - 99,29	1.315064e-01
HIB(P25-N1)-30 - RNA-20	**99,70 - 97,07**	**1.575894e-03**

HIB(P25-N1)-30 - KNN-20	99,70 - 99,69	7.856999e-01
HIB(P25-N1)-30 - HIB(P25-N1)-20	99,70 - 99,69	8.166086e-01
HIB(P25-N1)-30 - HIB(P25-N75)-20	99,70 - 99,30	1.293856e-01
HIB(P25-N1)-30 - RNA-12	**99,70 - 89,29**	**3.969225e-08**
HIB(P25-N1)-30 - KNN-12	**99,70 - 97,07**	**1.998839e-04**
HIB(P25-N1)-30 - HIB(P1-N1)-12	**99,70 - 97,54**	**4.078249e-03**
HIB(P25-N1)-30 - HIB(P1-N75)-12	**99,70 - 96,38**	**1.311698e-05**
HIB(P25-N1)-30 - RNA-6	**99,70 - 89,42**	**2.636345e-08**
HIB(P25-N1)-30 - KNN-6	**99,70 - 96,94**	**7.507693e-05**
HIB(P25-N1)-30 - HIB(P25-N1)-6	**99,70 - 97,11**	**4.051314e-04**
HIB(P25-N1)-30 - HIB(P1-N75)-6	**99,70 - 95,56**	**1.131354e-06**
HIB(P25-N75)-30 - RNA-20	99,29 - 97,07	1.353534e-01
HIB(P25-N75)-30 - KNN-20	99,29 - 99,69	2.404938e-01
HIB(P25-N75)-30 - HIB(P25-N1)-20	99,29 - 99,69	2.224726e-01
HIB(P25-N75)-30 - HIB(P25-N75)-20	99,29 - 99,30	9.876386e-01
HIB(P25-N75)-30 - RNA-12	**99,29 - 89,29**	**9.348309e-05**
HIB(P25-N75)-30 - KNN-12	**99,29 - 97,07**	**4.034449e-02**
HIB(P25-N75)-30 - HIB(P1-N1)-12	99,29 - 97,54	2.185282e-01
HIB(P25-N75)-30 - HIB(P1-N75)-12	**99,29 - 96,38**	**5.614805e-03**
HIB(P25-N75)-30 - RNA-6	**99,29 - 89,42**	**7.276621e-05**
HIB(P25-N75)-30 - KNN-6	**99,29 - 96,94**	**1.974879e-02**
HIB(P25-N75)-30 - HIB(P25-N1)-6	99,29 - 95,56	6.212155e-02
HIB(P25-N75)-30 - HIB(P1-N75)-6	**99,29 - 95,56**	**1.064890e-03**
RNA-20 - **KNN-20**	**97,07 - 99,69**	**5.299997e-03**
RNA-20 - **HIB(P25-N1)-20**	**97,07 - 99,69**	**4.471831e-03**
RNA-20 - HIB(P25-N75)-20	97,07 - 99,30	1.375698e-01
RNA-20 - RNA-12	**97,07 - 89,29**	**2.271442e-02**
RNA-20 - KNN-12	97,07 - 97,07	6.358270e-01
RNA-20 - HIB(P1-N1)-12	97,07 - 97,54	8.375566e-01
RNA-20 - HIB(P1-N75)-12	97,07 - 96,38	2.481555e-01
RNA-20 - RNA-6	**97,07 - 89,42**	**1.824094e-02**
RNA-20 - KNN-6	97,07 - 96,94	4.729928e-01
RNA-20 - HIB(P25-N1)-6	97,07 - 97,11	7.580863e-01
RNA-20 - HIB(P1-N75)-6	97,07 - 95,56	1.031884e-01
KNN-20 - HIB(P25-N1)-20	99,69 - 99,69	9.618168e-01
KNN-20 - HIB(P25-N75)-20	99,69 - 99,30	2.354972e-01
KNN-20 - RNA-12	**99,69 - 89,29**	**2.162416e-07**
KNN-20 - KNN-12	**99,69 - 97,07**	**7.196209e-04**

KNN-20 - HIB(P1-N1)-12	**99,69 - 97,54**	**1.219398e-02**
KNN-20 - HIB(P1-N75)-12	**99,69 - 96,38**	**5.884760e-05**
KNN-20 - RNA-6	**99,69 - 89,42**	**1.651377e-07**
KNN-20 - KNN-6	**99,69 - 96,94**	**2.864501e-04**
KNN-20 - HIB(P25-N1)-6	**99,69 - 97,11**	**1.355294e-03**
KNN-20 - HIB(P1-N75)-6	**99,69 - 95,56**	**6.384358e-06**
HIB(P25-N1)-20 - HIB(P25-N75)-20	99,69 - 99,30	2.193949e-01
HIB(P25-N1)-20 - RNA-12	**99,69 - 89,29**	**1.837541e-07**
HIB(P25-N1)-20 - KNN-12	**99,69 - 97,07**	**6.256006e-04**
HIB(P25-N1)-20 - HIB(P1-N1)-12	**99,69 - 97,54**	**1.042547e-02**
HIB(P25-N1)-20 - HIB(P1-N75)-12	**99,69 - 96,38**	**4.895595e-05**
HIB(P25-N1)-20 - RNA-6	**99,69 - 89,42**	**1.463350e-07**
HIB(P25-N1)-20 - KNN-6	**99,69 - 96,94**	**2.328364e-04**
HIB(P25-N1)-20 - HIB(P25-N1)-6	**99,69 - 97,11**	**1.139961e-03**
HIB(P25-N1)-20 - HIB(P1-N75)-6	**99,69 - 95,56**	**5.063613e-06**
HIB(P25-N75)-20 - RNA-12	**99,30 - 89,29**	**9.758257e-05**
HIB(P25-N75)-20 - KNN-12	**99,30 - 97,07**	**4.158868e-02**
HIB(P25-N75)-20 - HIB(P1-N1)-12	99,30 - 97,54	2.216134e-01
HIB(P25-N75)-20 - HIB(P1-N75)-12	**99,30 - 96,38**	**5.832999e-03**
HIB(P25-N75)-20 - RNA-6	**99,30 - 89,42**	**7.351636e-05**
HIB(P25-N75)-20 - KNN-6	**99,30 - 96,94**	**2.042369e-02**
HIB(P25-N75)-20 - HIB(P25-N1)-6	99,30 - 97,11	6.327788e-02
HIB(P25-N75)-20 - HIB(P1-N75)-6	**99,30 - 95,56**	**1.110897e-03**
RNA-12 - KNN-12	89,29 - 97,07	8.834888e-02
RNA-12 - **HIB(P1-N1)-12**	**89,29 - 97,54**	**1.055913e-02**
RNA-12 - HIB(P1-N75)-12	89,29 - 96,38	2.985967e-01
RNA-12 - RNA-6	89,29 - 89,42	9.454809e-01
RNA-12 - KNN-6	89,29 - 96,94	1.486063e-01
RNA-12 - HIB(P25-N1)-6	89,29 - 97,11	6.179256e-02
RNA-12 - HIB(P1-N75)-6	89,29 - 95,56	5.921711e-01
KNN-12 - HIB(P1-N1)-12	97,07 - 97,54	4.656275e-01
KNN-12 - HIB(P1-N75)-12	97,07 - 96,38	5.747478e-01
KNN-12 - RNA-6	97,07 - 89,42	7.671373e-02
KNN-12 - KNN-6	97,07 - 96,94	8.456199e-01
KNN-12 - HIB(P25-N1)-6	97,07 - 97,11	8.936261e-01
KNN-12 - HIB(P1-N75)-6	97,07 - 95,56	2.862251e-01
HIB(P1-N1)-12 - HIB(P1-N75)-12	97,54 - 96,38	1.577834e-01
HIB(P1-N1)-12 - RNA-6	**97,54 - 89,42**	**8.102377e-03**

HIB(P1-N1)-12 - KNN-6	97,54 - 96,94	3.134633e-01
HIB(P1-N1)-12 - HIB(P25-N1)-6	97,54 - 97,11	5.812476e-01
HIB(P1-N1)-12 - HIB(P1-N75)-6	97,54 - 95,56	6.007638e-02
HIB(P1-N75)-12 - RNA-6	96,38 - 89,42	2.640532e-01
HIB(P1-N75)-12 - KNN-6	96,38 - 96,94	7.431734e-01
HIB(P1-N75)-12 - HIB(P25-N1)-6	96,38 - 97,11	4.596132e-01
HIB(P1-N75)-12 - HIB(P1-N75)-6	96,38 - 95,56	6.890101e-01
RNA-6 - KNN-6	89,42 - 96,94	1.288171e-01
RNA-6 - HIB(P25-N1)-6	89,42 - 97,11	5.216304e-02
RNA-6 - HIB(P1-N75)-6	89,42 - 95,56	5.489246e-01
KNN-6 - HIB(P25-N1)-6	96,94 - 97,11	7.269848e-01
KNN-6 - HIB(P1-N75)-6	96,94 - 95,56	4.281828e-01
HIB(P25-N1)-6 - HIB(P1-N75)-6	97,11 - 95,56	2.147020e-01