UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CASSIANO RICARDO DE OLIVEIRA PERES

ABORDAGEM COMPARATIVA ENTRE TÉCNICAS DE RECONHECIMENTO DE PADRÕES PARA CLASSIFICAÇÃO DE PACOTES DE DADOS TCP/IP EM UMA REDE DE COMPUTADORES ETHERNET

TRABALHO DE DIPLOMAÇÃO

MEDIANEIRA
2015

CASSIANO RICARDO DE OLIVEIRA PERES

ABORDAGEM COMPARATIVA ENTRE TÉCNICAS DE RECONHECIMENTO DE PADRÕES PARA CLASSIFICAÇÃO DE PACOTES DE DADOS TCP/IP EM UMA REDE DE COMPUTADORES ETHERNET

Trabalho de Conclusão de Curso apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. Dr. Paulo Lopes de Menezes.

Co-orientador: Prof. Dr. Pedro Luiz de Paula Filho.

MEDIANEIRA

2015

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Diretoria de Graduação e Educação Profissional
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

TERMO DE APROVAÇÃO

Abordagem Comparativa entre Técnicas de Reconhecimento de Padrões para Classificação de Pacotes de Dados TCP/IP em uma Rede de Computadores Ethernet

Por

Cassiano Ricardo de Oliveira Peres

Este Trabalho de Diplomação (TD) foi apresentado às 09:10 h do dia 11 de junho de 2015 como requisito parcial para a obtenção do título de Tecnólogo no Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Medianeira. O acadêmico foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

Prof. Dr. Paulo Lopes de Menezes. UTFPR – Câmpus Medianeira (Orientador)	Prof. Dr. Pedro Luiz de Paula Filho UTFPR – Câmpus Medianeira (Convidado)
Prof. Dr. Neylor Michel UTFPR – Câmpus Medianeira (Convidado)	Prof. Msc Juliano Rodrigo Lamb UTFPR – Câmpus Medianeira (Responsável pelas atividades de TCC)

- O Termo de Aprovação assinado encontra-se na Coordenação do Curso -

“*Be thankful for everything, for soon there will be nothing...”* (28 Days Later, 2002).

RESUMO

Cassiano Ricardo de Oliveira Peres. 2015. 75 f. Trabalho de Diplomação (Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas). Universidade Tecnológica Federal do Paraná – UTFPR. Medianeira. 2015.

Neste trabalho foi realizado um estudo comparativo sobre a utilização de técnicas de reconhecimento de padrões e classificação de protocolos de redes de computadores em pacotes de dados através de redes neurais artificiais (RNA) e *support vector machines* (SVM). Foram abordados aspectos relacionados à arquitetura das redes neurais, como algoritmos de treinamento, amostras de dados, funções de ativação, conceitos, entre outros aspectos. Outro ponto tido como objeto de estudo deste trabalho foram as *support vector machines*, ou máquina de suporte a vetores, sendo esta outra técnica de reconhecimento de padrões cujas características tais como, algorítmos de aprendizagem de máquina, conceitos, entre outras, foram abordadas. O estudo desenvolvido também explorou as características dos pacotes de dados analisados, apresentando as diferenças estruturais entre os pacotes, com base nos protocolos de redes de computadores contidos nos mesmos. Por fim, foram desenvolvidas alguns exemplos de redes neurais artificiais e SVMs, as quais foram utilizadas para o reconhecimento de padrões. Após a realização de testes, o desempenho das duas técnicas foram comparados com o objetivo de apresentar uma conclusão sobre qual das duas técnicas apresenta desempenho mais satisfatório no caso apresentado.

Palavras-chave: Redes Neurais Artificiais. Protocolos. Máquinas de Vetores Suporte.

ABSTRACT

Cassiano Ricardo de Oliveira Peres. 2015. 69 f. Working graduation (Degree in Technology Analysis and Systems Development). Federal Technological University of Paraná - UTFPR. Medianeira. 2015.

This work represents a comparative study on the use of techniques of pattern recognition and classification of computer networking protocols into data packets through artificial neural networks (ANN) and support vector machines (SVM). We were approached aspects related to the architecture of neural networks, such as training algorithms, data samples, activation functions, concepts, and so on. Another point taken as the object of study of this work were the support vector machines, which is another pattern recognition technique whose characteristics such as machine learning algorithms, concepts, among others, were addressed. The study also developed explored the characteristics of the data packets analyzed, showing the structural differences between the packets on the basis of computer network protocols contained therein. Finally, it was developed some examples of artificial neural networks and SVMs, which were used for the pattern recognition. After testing, the performance of the two techniques were compared in order to present a conclusion as to which of these features more satisfactory performance in the case presented.

Keywords: Artificial Neural Networks. Protocols. Support Vector Machines.

LISTA DE FIGURAS

LISTA DE SIGLAS

ABC	Artificial Bee Colony
ACO	Artificial Colony Optimization
DE	Differential Evolution
FTP	File Transfer Protocol
HTTP	Hipertext Transfer Protocol
IP	Internet Protocol
NCP	Network Control Protocol
PSO	Particle Swarm Optimization
RCP	Remote Copy
RFB	Radio Basis Function
RNA	Redes Neurais Artificiais
SOM	Self Organizing Map
SVM	Support Vector Machine
TAE	Teoria do Aprendizado Estatístico
TCP	Transfer Control Protocol
UDP	User Datagram Protocol
TAE	Teoria do Aprendizado Estatístico

SUMÁRIO

1 INTRODUÇÃO

Segundo dados estatísticos do Comitê Gestor da Internet no Brasil (2012), 45% das pessoas no país, utilizam computadores com acesso à Internet. Isso demonstra a popularização do uso de dispositivos com acesso e da navegação na rede mundial de computadores. Esse aumento na utilização da Internet gerou, e continua gerando, um significativo aumento na quantidade de dados que trafegam na rede. Porém, junto com o aumento da quantidade de informações, também aumenta a quantidade e formas de ameaças à segurança dos dados, devido à popularização e sofisticação das técnicas e ferramentas utilizadas para fins ilícitos.

Para a tomada de decisões relacionadas à segurança da informação em redes de computadores, pode ser interessante aos administradores de sistemas conhecer o padrão de dados que circulam em suas redes, pois desta forma podem ser apontados comportamentos na rede que possam representar ameaças à segurança.

Assim, a proposta deste trabalho é realizar um estudo comparando o desempenho das redes neurais artificiais (RNA) e *support vector machines* (SVM) para o reconhecimento de padrões de pacotes de dados em rede Ethernet, a qual pode servir de base para aplicações mais específicas, tais como o monitoramento e controle de tráfego de dados ou análise do comportamento de usuários de uma rede de computadores.

1.1 OBJETIVO GERAL

Desenvolver um estudo comparativo de desempenho entre redes neurais artificiais e *support vector machines*, no reconhecimento de padrões em características de pacotes de dados que trafegam em uma rede de computadores Ethernet.

1.2 OBJETIVOS ESPECÍFICOS

O objetivo geral deste trabalho é composto dos seguinte objetivos específicos:

1. Gerar amostras de dados para os processos envolvidos no reconhecimento de padrões.
2. Compor e testar diferentes topologias de RNA e SVM e identificar as que obtiverem o melhor desempenho;
3. Comparar o desempenho das melhores topologias de RNA e SVM desenvolvidas;
4. Justificar com dados estatísticos a escolha da melhor técnica de reconhecimento de padrões, baseado nos resultados dos testes realizados para o problema em questão.

1.3 JUSTIFICATIVA

Um ataque (exploração de falhas em um sistema) contra um computador, rede ou servidor pode causar sérios danos, como perda ou furto de dados, interrupção de serviços, invasões de redes entre outros problemas. Dessa forma, manter a detecção, prevenção e proteção contra ataques é algo imprescindível. Assim, a principal função de sistemas de segurança, é a sua capacidade de detectar ameaças contra a segurança da informação (IFTIKHAR, ABDULLAH, ALGHAMDI, 2009).

A utilização de técnicas de reconhecimento de padrões utilizando redes neurais artificiais pode ser importante no desenvolvimento de aplicações voltadas à segurança da informação. Pode-se detectar comportamento suspeitos tanto de usuários como de máquinas, onde suas ações podem ser consideradas anormais comparadas ao padrão de tráfego de dados de uma determinada rede. Ações como usuários tentando acessar áreas restritas, a utilização de “força bruta” testando várias senhas de forma seguida, máquinas sendo utilizadas como “escravas” por aplicações remotas enviando e recebendo dados sem o conhecimento dos seus usuários, acesso a tipos de conteúdo indevidos para o local e horário, entre outras utilidades.

Dessa forma, o desenvolvimento e utilização de aplicações voltadas à segurança utilizando técnicas de reconhecimento de padrões pode servir de fonte de informação para a tomada de decisões por parte dos gerentes e responsáveis pela segurança.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido basicamente em cinco grande sessões. A primeira trata da bibliografia referente à tecnologia das redes neurais artificiais, onde é feito um breve estudo teórico sobre o assunto.

Na segunda sessão é tratado da tecnologia das *support vector machines*. Já na terceira sessão, é abordado o estudo bibliográfico referente a protocolos e pacotes de dados de redes de computadores.

Na quarta sessão é realizada a apresentação dos resultados obtidos pelos testes efetuadas utilizando as tecnologias estudadas nas sessões anteriores.

Por fim, na quinta sessão é feita a apresentação das conclusões obtidas por meio da análise dos resultados obtidos deste trabalho.

2 DESENVOLVIMENTO

Neste capítulo será descrita a bibliografia pesquisada referente às tecnologias utilizadas no desenvolvimento do trabalho.

2.1 REDES NEURAIS ARTIFICIAIS

As redes neurais artificiais (RNA), ou simplesmente redes neurais, surgiram da motivação de que o cérebro humano processa informações de forma muito diferente da computação convencional. O cérebro humano pode ser considerado um computador altamente complexo, com processamento não-linear e paralelo. É organizado em componentes estruturais chamados neurônios, os quais realizam o processamento de dados e informações de forma muito mais eficiente do que a computação linear convencional.

De acordo com Haykin (1999), uma RNA pode ser definida como:

> Um processador paralelo massivamente distribuído, composto de unidades simples de processamento, as quais possuem uma propensão natural para armazenar conhecimento experimental e torná-lo disponível para uso. A rede neural se assemelha ao cérebro em dois aspectos:
> 1. O conhecimento é adquirido do seu ambiente através de processos de aprendizagem.
> 2. A força das conexões entre neurônios, conhecidas como pesos sinápticos, são utilizadas para armazenar o conhecimento adquirido.

De acordo com a definição de Stuart e Norvig (1995), uma rede neural é composta por um número de nós, ou unidades conectadas por ligações. Cada ligação possui um peso numérico associado, que é atualizado de acordo com o conhecimento adquirido. Algumas unidades estão conectadas com o ambiente externo à rede neural, sendo estas as unidades de entrada e saída.

2.1.1 Histórico

A era moderna das redes neurais começou com o trabalho pioneiro de McCulloch e Pitts, que propuseram o primeiro modelo matemático do neurônio artificial. Warren Sturgis McCulloch foi um psiquiatra e neurofisiologista que passou cerca de vinte anos trabalhando para desenvolver uma forma de representar eventos do sistema nervoso. Walter Harry Pitts Jr. foi um matemático prestigiado que juntou-se a McCulloch no ano de 1942. Em seu principal artigo, *A Logical Calculus of the Ideas Immanent in Nervous Activity*, escrito no ano de 1943, os autores descreveram o cálculo lógico das redes neurais, o qual uniu os estudos de neurofisiologia e lógica matemática (HAYKIN, 1999).

No ano de 1948, Norbert Wiener publicou o livro *Cybernetics*, no qual eram descritos alguns importantes conceitos para controle, comunicações e processamento estatístico de sinais. A segunda edição do livro foi publicada em 1961, com a adição de materiais sobre auto-organização e aprendizagem.

Em 1948, John Von Neumann escreveu o livro *The Computer and the Brain* no qual propôs a modelagem da performance do cérebro humano para os recursos de hardware disponíveis naquela época.

Já no ano de 1957, Frank Rosenblatt desenvolveu modelos de neurônios em hardware. Esses modelos ultimamente resultaram no conceito do Perceptron. Este foi um importante desenvolvimento conceitual que ainda tem grande utilização.

O maior impulso no desenvolvimento no tema de redes neurais ocorreu em 1949 com a publicação do livro *The Organization of Behavior*, de Donald Hebb, no qual explicitou a demonstração da regra da aprendizagem fisiológica para a modificação sináptica. Hebb propôs de forma específica que a conectividade do cérebro está continuamente mudando, de acordo com as novas tarefas que o organismo aprende. Este livro vem sendo utilizado como fonte de inspiração para o desenvolvimento de modelos computacionais de aprendizagem e sistemas adaptativos. O trabalho de Rochester, Holland, Haibt e Duda, de 1956, foi de certa forma o primeiro a utilizar a simulação computacional para testar a teoria neural baseada no postulado de aprendizagem de Hebb.

Em 1982, John Hopfield utilizou a ideia de uma função de energia para formular uma nova forma de interpretar o cálculo realizado por redes recorrentes com conexões sinápticas

simétricas. Esse tipo de rede neural ficou conhecido como Redes de Hopfield e atraiu a atenção para as redes neurais na década de 1980.

No ano de 1986, o algoritmo de *backpropagation* (retropropagação) foi proposto por Rumelhart, Hinton e Williams. No mesmo ano, o aclamado livro *Paralell Distributed Processing: Explorations int the Microstructures of Cognition* editado por Rumelhart e McClelland foi publicado. Este livro foi a maior influência para a utilização do aprendizado com retropropagação. Considera-se que o trabalho realizado por Hopfield em 1982 e o livro de McClelland e Rumelhart foram as publicações mais influentes e responsáveis pelo ressurgimento do interesse no assunto de redes neurais na década de 1980 (HAYKIN, 1999).

2.1.2 Características das redes neurais artificiais

Uma das características mais evidentes das RNAs, é a forma de processamento paralelo da informação devido à sua estrutura. Outra característica relevante é a capacidade de adquirir conhecimento por experiência e utilizá-lo para a generalização. Generalização é a capacidade da rede neural produzir resultados para dados de entrada que não foram observados no treinamento. Essas duas capacidades tornam possíveis a solução de problemas considerados intratáveis na computação linear.

Haykin (1999), aponta algumas vantagens da utilização de redes neurais artificiais, sendo algumas destas:

Adaptabilidade: Redes neurais artificiais podem adaptar seus pesos sinápticos de acordo com as mudanças no ambiente externo (entradas). Basicamente, uma rede neural treinada para um tipo de ambiente, pode ser treinada novamente para responder às menores mudanças que ocorrerem.

Resposta evidente: No contexto da classificação de padrões, uma RNA pode ser projetada não apenas para fornecer informações sobre um padrão em particular, mas também confiança para a tomada de decisão. Isto é, pode ser utilizada para rejeitar padrões ambíguos, e dessa forma, aumentar o desempenho de classificação da rede neural.

Informação contextual: O conhecimento adquirido é representado pelo estado de ativação da rede neural. Cada neurônio é potencialmente afetado pela atividade global de

todos os outros neurônios da RNA. Consequentemente, a informação contextual é tratada com naturalidade pela RNA.

Tolerância a falhas: Redes neurais, quando implementadas em hardware, tem a capacidade de serem tolerantes a falhas, ou capazes de efetuar processamento robusto quando sua performance diminuir devido a condições adversas de operação.

Analogia neurobiológica: O projeto de uma RNA é motivado pela analogia com o cérebro, o qual é uma prova viva de que a tolerância a falhas e processamento paralelos não são apenas fisicamente possíveis mas como também rápidos e poderosos. Neurobiólogos enxergam as RNA como ferramentas de pesquisa para a interpretação dos fenômenos neurológicos. De outra forma, engenheiros as veem como fonte de novas ideias para a solução de problemas muito complexos, que as técnicas clássicas são capazes de resolver.

2.1.3 O cérebro humano

O sistema nervoso humano pode ser visto como um sistema de três estágios, no qual o centro do sistema é o cérebro que recebe e interpreta os estímulos recebidos pelos nervos. Os receptores nervosos convertem estímulos externos em impulsos elétricos que fornecem informações ao cérebro. Por fim, as células efetoras convertem impulsos elétricos que são gerados pelo sistema nervoso em respostas discerníveis para os sistemas de saída.

O entendimento mais prático do cérebro humano se deu graças ao trabalho de Ramón y Cajal em 1911, que introduziu a ideia de neurônios como estruturas componentes do cérebro. É estimado que o córtex cerebral tenha aproximadamente 10 bilhões de neurônios e 60 trilhões de sinapses ou ligações sinápticas (SHEPHERD e KOCH, 1990).

As sinapses são as unidades funcionais e estruturais que fazem a interação entre neurônios. O tipo mais comum de sinapse é a sinapse química, na qual um processo pré-sináptico libera uma substância transmissora que se espalha pela região de junção sináptica entre neurônios e age no processo pós-sináptico. Então, a sinapse converte o sinal elétrico pré-sináptico em um sinal químico, transformando de volta em um sinal pós-sináptico (Shepherd e Koch, 1990). Uma sinapse pode ser inibitória ou excitatória, mas nunca ambas em um

neurônio receptor. Em uma rede neural artificial, valores de entrada excitatórios são positivos, e inibitórios são negativos.

No diagrama da Figura 1 é apresentado o sistema nervoso e as etapas entre a recepção do estímulo externo pelos receptores, o processamento do estimulo pelo cérebro (rede neural) e o sinal de saída, a resposta emitida pelas células efetoras.

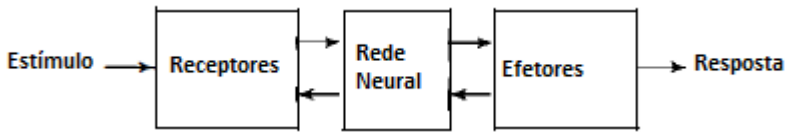

Figura 1 – Tratamento de um estímulo externo ao sistema nervoso.
Fonte:Adaptado de Haykin (1999).

Os neurônios, como o exemplo apresentado na Figura 2, são as estruturas básicas que compõem o cérebro humano e, em geral, são compostos de três partes fundamentais, sendo os dendritos que recebem o sinal de entrada, o corpo da célula que é responsável pela interpretação do sinal, e o axônio que leva o sinal de saída para as terminações sinápticas, para transmitir o sinal para outras células (HAYKIN, 1999).

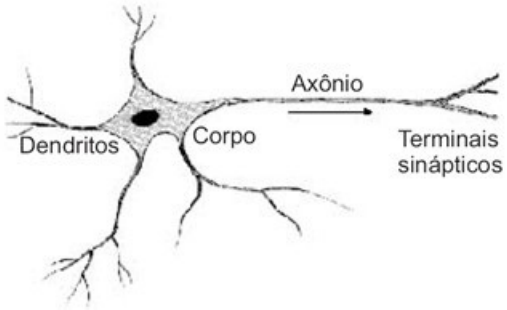

Figura 2 – Neurônio humano e suas principais estruturas.
Fonte:Adaptado de Ferneda (2006).

O cérebro humano possui organizações anatômicas de grande e pequena escala, e diferentes funcionalidades para cada, desde as de baixo nível, até as de alto nível. Na Figura 3 é exibido uma hierarquia entre os níveis de organização anatômica.

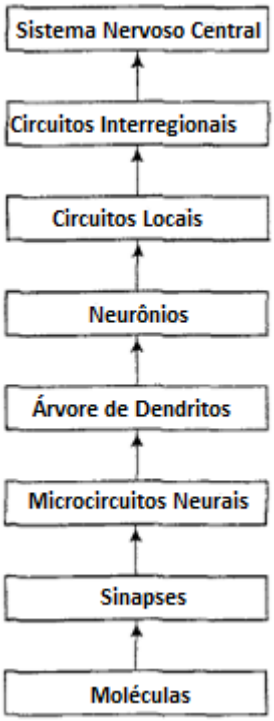

Figura 3 – Níveis de hierarquia anatômica do sistema nervoso humano.
Fonte: Adaptado de Haykin (1999).

As sinapses representam o nível mais fundamental, dependendo de moléculas e íons para agirem. Após as sinapses, há microcircuitos neurais, árvores de dendritos e os neurônios. Um microcircuito neural é um conjunto de sinapses organizado em padrões de conectividade para gerar uma operação funcional.

O neurônio possui várias unidades de dendritos. O próximo nível de complexidade da hierarquia representa os circuitos locais, composto de neurônios com propriedades similares ou diferentes, responsáveis pelas operações de uma determinada área do cérebro. O nível seguinte é dos circuitos inter-regionais, compostos de caminhos, colunas e mapas topográficos que envolvem múltiplas regiões localizadas em diferentes partes do cérebro. Os mapas topográficos são organizados para responder a informação sensorial recebida. O último nível

de complexidade é o sistema nervoso central, composto de mapas topográficos e outros circuitos inter-regionais

2.1.3.1 O neurônio artificial

Um neurônio é a unidade fundamental de processamento da informação em uma rede neural. São identificados três elementos principais de um modelo de neurônio artificial:

Conjunto de sinapses ou ligações sinápticas: cada uma delas é caracterizada por valor, chamado peso ou força. O valor do peso sináptico é multiplicado pelo valor da entrada, e varia entre valores positivos e negativos, representando respectivamente os sinais excitatórios e inibitórios de um neurônio biológico.

Junção de soma dos sinais de entrada: após estes serem multiplicados pelos pesos sinápticos. Essa operação é descrita pela utilização de um combinador linear.

Função de ativação: objetivo de limitar a amplitude da saída de um neurônio. A função de ativação também é chamada de *squashing function*, ou função de limitação, que limita o intervalo de amplitude do sinal de saída do neurônio para valores finitos. De forma geral, a amplitude do intervalo da saída de um neurônio é escrita como o conjunto unitário fechado [0,1] ou de forma alternativa [-1,1].

O modelo neural, Figura 4 também inclui um valor aplicado externamente, o *bias*, representado por *bias*. O bias atua aumentando ou diminuindo a entrada da função de ativação, dependendo do valor ser positivo ou negativo, respectivamente.

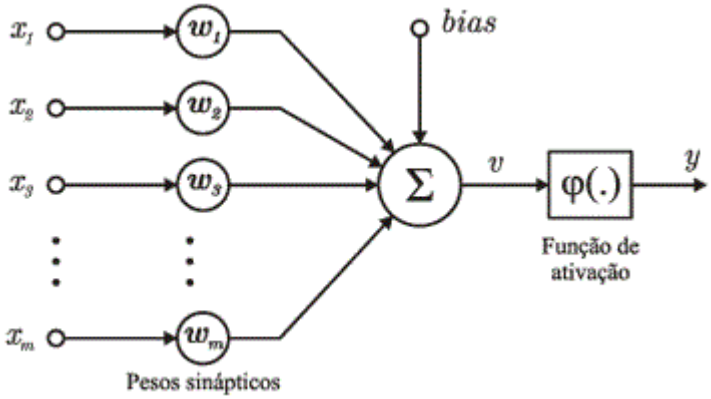

Figura 4 – Neurônio artificial.
Fonte: Adaptado de Moreto e Rolim (2010).

Em termos matemáticos, um neurônio *k* pode ser descrito com o seguinte par de equações (1.1) e (1.2):

$$u = \sum_{j=1}^{m} w_{kj} x_j \tag{1.1}$$

e

$$y = \varphi(u + bias) \tag{1.2}$$

onde $x_1, x_2, x_3, ..., x_m$ são os sinais de entrada, $w_1, w_2, w_3, ..., w_m$ são os pesos sinápticos do neurônio, v é a saída do combinador linear dos sinais de entrada, $bias$ é o valor do bias, $\varphi(.)$ é a função de ativação do neurônio.

2.1.3.2 Funções de ativação

A função de ativação do neurônio define o valor de saída após o processamento dos valores de entrada. As três funções a seguir são as mais utilizadas:

Threshold function **(Função de Limiar)**: Nesse tipo de função de ativação, tem-se a equação:

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

(1.3)

Em engenharia, essa forma de função de limiar é chamada frequentemente de *heaviside function*. Dessa forma, a saída de um determinado neurônio *k* empregando a função de limiar é expressada da seguinte forma:

$$y_k = \begin{cases} 1 & se \quad v_k \geq 0 \\ 0 & se \quad v_k < 0 \end{cases} \tag{1.4}$$

onde v_k é o campo induzido local (potencial de ativação) do neurônio, sendo este:

$$v_k = \sum_{j=1}^{m} w_{kj} x_j + bias_k \tag{1.5}$$

Na Figura 5 é apresentado o gráfico gerado pela função de ativação *threshold*:

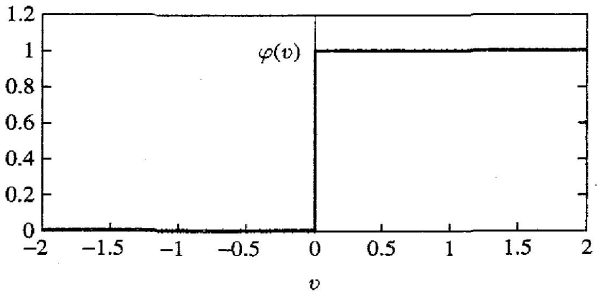

Figura 5 – Gráfico da função de ativação *threshold*.
Fonte: Haykin (1999).

O neurônio não linear é baseado no modelo de McCulloch-Pitts (1943), o qual a saída do neurônio obtém valor 1 se o campo local induzido é não-negativo, e 0 em caso contrário. Essa característica descreve a propriedade *all-or-none* (tudo ou nada) do modelo de McCulloch-Pitts.

***Piecewise-Linear Function* (Função linear de rampa)**: Para esta função de ativação, tem-se a equação (1.6):

$$\varphi(v) = \begin{cases} 1, & v \geq +\frac{1}{2} \\ v, & +\frac{1}{2} \quad v > -\frac{1}{2} \\ 0, & v \leq -\frac{1}{2} \end{cases} \tag{1.6}$$

onde o fator de amplificação dentro da região de operação é assumido para ser a unidade. As duas situações seguintes podem ser vistas como formas especiais da função linear de rampa:

- Um combinador linear é originado se a região de operação é mantida sem executar em saturação (quando os parâmetros livres da rede neural tentem ao infinito).
- A função *piecewise-linear* reduz a uma função de limiar se o fator de amplificação da região linear é infinitamente grande.

  Na Figura 6 é apresentado o gráfico gerado pela função *piecewise-linear.*

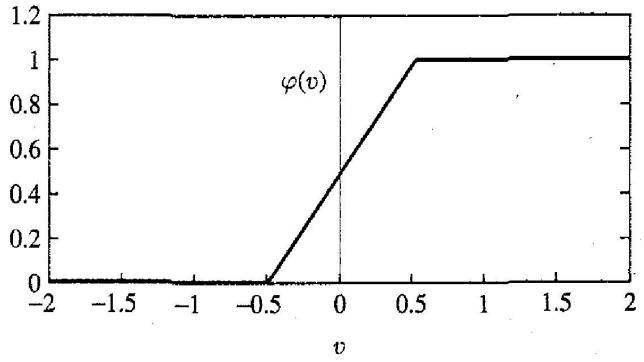

Figura 6 – Gráfico da função de ativação *piecewise-linear*.
Fonte: Haykin (1999).

Função sigmoide: Essa função de ativação é a mais comumente utilizada, e também foi utilizada nas redes neurais desenvolvidas nesse trabalho. É definida como uma função estritamente crescente. Um exemplo de função sigmoide é a função logística, definida pela equação (1.7):

$$\varphi(v) = \frac{1}{1 + exp(-av)} \tag{1.7}$$

onde *a* é o parâmetro de inclinação da função sigmoide. Com a variação do parâmetro *a* obtêm-se funções sigmoides com diferentes inclinações, como o apresentado na Figura 7. De fato, a inclinação na origem é igual a *a*/4, e no seu limite quando se aproxima do infinito, ela se torna uma função de limiar simples.

Enquanto que função de limiar assume valores de 0 ou 1, a função sigmoide pode assumir vários valores no intervalo entre 0 e 1. Outra característica é que a função sigmoide é diferenciável, ao contrário da função de limiar que não o é.

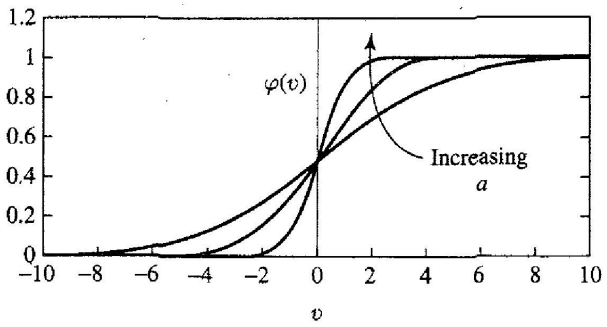

Figura 7 – Gráfico da função de ativação sigmóide.
Fonte: Haykin (1999).

2.1.4 Regra Delta Generalizada

As redes neurais artificiais que trabalham com o algoritmo *backpropagation* utilizam uma variação da regra delta padrão, conhecida como regra delta generalizada, sendo esta mais apropriada para redes neurais multicamadas. A regra padrão implementa um gradiente descendente no quadrado das somas dos erros em funções de ativação lineares.

A regra delta generalizada é utilizada quando a rede neural apresenta uma função de ativação semi-linear, sendo esta uma função diferenciável e não decrescente. Uma função comumente utilizada nesse caso é a sigmoide.

Um problema na utilização do *backpropagation*, é que podem ser necessárias muitas repetições no conjunto de treinamento, resultando em um aumento no tempo de treino. Se for encontrado um mínimo local, o erro do conjunto de treinamento não diminui e para em um valor maior que o aceitável. Uma forma utilizada para aumentar a taxa de aprendizagem é a utilização de um termo chamado *momentum*, através de uma modificação na regra delta generalizada. O *momentum* é uma constante que determina o efeito das mudanças passadas nos pesos sinápticos.

As equações seguintes são utilizadas para calcular os ajustes dos pesos (1.8) e atualizar os pesos sinápticos (1.9) com a regra delta generalizada, após o cálculo dos erros de cada neurônio.

$$\Delta w_{kj}(n+1) = \alpha w_{kj}(n) + \eta \delta_j y \tag{1.8}$$

e,

$$w(n+1) = w(n) + \Delta w_{kj}(n) \tag{1.9}$$

onde, Δw_{kj} é a variação os pesos das conexões com a camada anterior, α é a constante *momentum* (quando α = 0, a função se torna uma regra delta padrão), η representa a taxa de aprendizado, δ_j é o erro da unidade neural, y_j é a saída da unidade neural (LNCC,2014).

2.1.5 Arquiteturas de RNA

A forma na qual os neurônios estão dispostos está totalmente ligada ao algoritmo utilizado para o treinamento da rede neural. Os algoritmos de treinamento utilizados no desenvolvimento das redes neurais podem ser referenciados como sendo estruturados. Em geral, são identificadas três classes de arquiteturas de redes neurais fundamentais, as *single layer feedforward networks*, *Multilayer networks* e as *recurrent networks*.

As redes neurais *Single-Layer Feedforward Networks* (Redes neurais de camada única com alimentação para frente), como o nome sugere, possuem os neurônios organizados em camadas, sendo a camada de entrada que recebe os dados a serem processados, e a camada de saída, responsável pelo processamento, mas não o contrário. Ou seja, a rede possui propagação da informação apenas no sentido da entrada para a saída, sem formar ciclos.

Apesar de possuir duas camadas de neurônios, apenas a camada de saída é contada, pois não a processamento da informação nos neurônios da camada de entrada. Na Figura 8 é representada um modelo de rede neural de camada única.

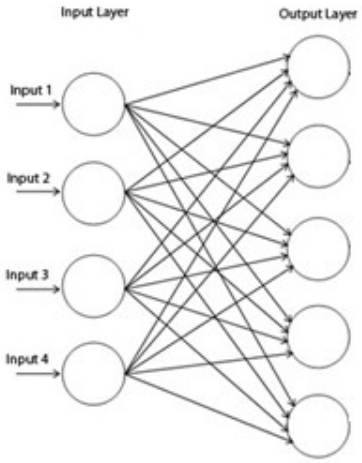

Figura 8 – Rede neural de camada única.
Fonte: Adaptado de Wang, Whu e Zao (2013).

Já as redes neurais *Multilayer Feedforward Networks* (Redes neurais de múltiplas camadas) se distinguem por possuírem uma ou mais camadas intermediarias, (ou ocultas) cujas unidades de processamento são chamadas de neurônios ocultos ou unidades ocultas. (CHURCHLAND e SEJNOWSKI, 1992).

As saídas dos neurônios da primeira camada oculta são os sinais de entrada para a segunda camada e assim por diante. Na da Figura 9 é representada uma rede neural multicamadas do tipo totalmente conectada, do inglês *fully connected*, na qual todos os neurônios da camada anterior estão conectados com os neurônios da camada seguinte. Quando uma rede neural não é do tipo totalmente conectada, ela é denominada parcialmente conectada. Na Figura 9, é representado o modelo de uma rede neural artificial multicamadas.

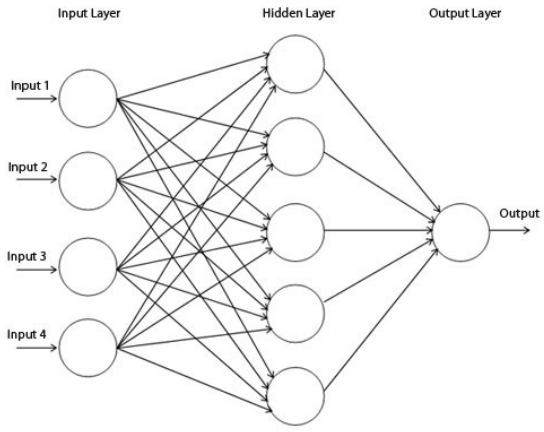

Figura 9 – Rede neural multicamadas.
Fonte: Adaptado de Wang, Whu e Zao (2013).

Por fim, as redes recorrentes, do inglês *recurrent networks* possuem a característica de ter pelo menos um *loop* de *feedback* (ou retorno), como por exemplo, uma rede neural de camada única tendo suas saídas ligadas com a sua própria entrada (Figura 10).

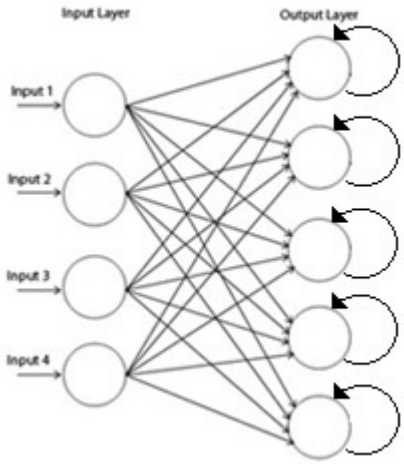

Figura 10 – Rede neural recorrente.
Fonte: Adaptado de Wang, Whu e Zao (2013).

2.1.6 Redes neurais multicamadas e o algoritmo *backpropagation*

As redes neurais artificiais (ou perceptron) multicamadas com alimentação para a frente tem seu nome vem do inglês *Multilayer feedforward networks* e são compostas de um conjunto de unidades sensores que compõe a camada de entrada, uma ou mais camadas ocultas de processamento de informação, e uma camada de saída. O sinal de entrada é propagado através da rede apenas na direção das camadas seguintes, por isso também são conhecidas como *feedforward multilayer perceptron* (MLP).

As MLP são frequentemente utilizadas com sucesso para resolver diversos problemas difíceis através do aprendizado supervisionado, onde um algoritmo muito popular, conhecido como *error backpropagation algorithm* (algoritmo de retro-propagação do erro). Esse algoritmo é baseado na regra de aprendizado de correção de erro, o qual pode ser visto como uma adaptação de outro algoritmo bastante popular, o LMS (*least mean square*), utilizado em RNA de única camada.

O algoritmo de aprendizagem *backpropagation* consiste em dois passos principais através das camadas de neurônios da RNA: uma propagação para frente, e uma retro-propagação. Na propagação, um padrão de entrada é apresentado à RNA em conjunto com sua respectiva saída desejada e é propagado camada à camada, no sentido da entrada para a saída.

Após o processamento das entradas, os pesos sinápticos são fixados e um conjunto contendo valores de saída é gerado. Os valores de saída gerados pela RNA são comparados aos valores de saída desejados, e um valor de erro é gerado. Na retropropagação, o sinal de erro é transmitido através das camadas da RNA no sentido da saída para a entrada, e os pesos sinápticos são atualizados, para gerarem valores saídas cada vez mais próximas dos valores desejados.

O erro gerado pela RNA é conhecido como SSE, *Sum of Squared Errors*, do inglês soma dos erros quadráticos, o qual determina a performance da RNA de acordo com a soma dos quadrados dos erros gerados pelo neurônio artificial no processamento dos valores de entrada (MATHWORKS, 2014).

As redes neurais artificiais multicamadas possuem três características principais:

Função de ativação não linear: Uma forma muito utilizada de não-linearidade é a não-linearidade sigmoide, definida pela função logística da Equação (2.0):

$$y_i = \frac{1}{1 + exp(-v_j)} \tag{2.0}$$

onde v_j é o campo local induzido (a soma de todos os pesos sinápticos mais o bias) do neurônio *j*, e y_j é a saída do neurônio.

Camadas ocultas: Estas camadas (excluindo as camadas de entrada e saída) permitem à RNA aprender tarefas complexas extraindo progressivamente mais características dos padrões de entrada.

Grande grau de conectividade: determinado pelas sinapses da rede neural. Uma mudança na conectividade da RNA exige mudanças na quantidade conexões sinápticas ou em seus pesos.

É baseado na combinação dessas três características junto com a habilidade de aprender com a experiência através do treinamento que a RNA possui seu poder de processamento da informação. Porém, essas características também causam algumas limitações à RNA, como as citadas a seguir:

- A alta conectividade, e a presença da não-linearidade distribuída fazem a analise teórica de uma RNA multicamadas ser difícil de ser implementada.
- A utilização de camadas ocultas torna difícil a visualização do processo de aprendizagem. De forma implícita, o processo de aprendizagem deve decidir quais características do padrão de entrada devem ser apresentadas aos neurônios das camadas ocultas.
- O processo de aprendizagem se torna complexo pois a quantidade de funções e a quantidade de alternativas para a representação da entrada se torna muito grande (HINTON, 1989).

O termo *backpropagation* apareceu depois do ano de 1985, quando foi popularizado através da publicação do livro *Parallel Distributed Processing* de Rumelhart e McClelland (1986).

As arquiteturas de RNA desenvolvidas neste trabalho são do tipo multicamadas e também são conhecidas como redes neurais totalmente conectadas, pois todos os neurônios da camada anterior estão conectados com todos os das camadas seguintes. Segundo Parker (1987), podem ser identificados dois tipos de sinais em uma rede neural multicamadas, como o representado na Figura 11.

Sinal de função: É um sinal de entrada para a rede neural e é propagado no sentido da entrada para a saída da RNA e produz um sinal de saída, na última camada da rede neural. É chamado de sinal de função por duas razões, e a primeira é por que presume-se que seja utilizada uma função matemática para gerar o resultado de saída. O segundo motivo é que cada neurônio da rede neural por onde passa o sinal de função, o sinal é calculado como uma função de entradas e pesos sinápticos aplicados ao neurônio. O sinal de função também chamado de sinal de entrada.

Sinais de erro:Um sinal de erro é originado da saída de um neurônio e é propagado no sentido da saída para a entrada, camada a camada.

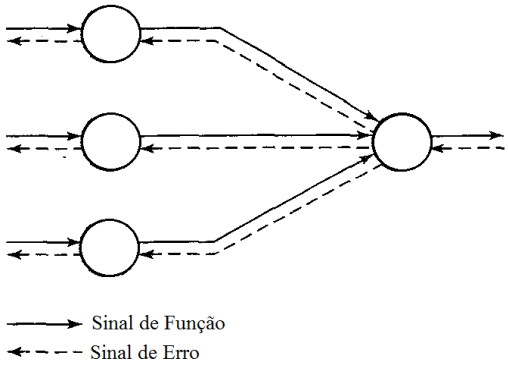

Figura 11 – Sentido da propagação dos sinais em uma RNA.
Fonte: Adaptado de Haykin (1999).

De forma geral, pode-se definir o algoritmo *backpropagation* em cinco passos principais:

1. **Inicialização**: onde são atribuídos valores iniciais aleatórios às ligações sinápticas
2. **Apresentação de amostras de treinamento**: no qual são apresentados os valores de entrada, as épocas de treinamento e os valores de saída desejados.
3. C**álculo da saída no sentido para a frente**: Os valores da entrada são propagados para as camadas da rede neural e os valores de saída são calculados/
4. **Calculo do erro**: É gerado um sinal de erro após o cálculo da saída e a comparação com a saída desejada. Esse erro é retropropagado para a rede, permitindo o ajuste dos pesos sinápticos
5. **Iteração**: Repetição dos passos 3 e 4 até que se atinja o valor de erro desejado, ou quando o número de repetições chega ao fim.

2.1.7 Tipos de redes neurais artificiais

Adaline Network: Criada por Bernard Widrow em 1959, seu nome deriva de Adaptative Linear Element. Foi inicialmente desenvolvida para reconhecer padrões binários, que ao ler um bit em um conjunto de bits, ela pode prever o próximo (Uhrig, 1995).

Madaline Network: Também criada por Widrow, foi a primeira rede neural artificial a ser aplicada em um problema real. Seu nome vem de Multiple Adaptative Linear Element (Introduction to Artificial Neural Networks, 1995).

Hopfield Network: A rede Hopfield aplica um princípio chamado de armazenamento de informações como atratores dinamicamente estáveis. A recuperação da informação acontece via um processo dinâmico de atualização dos estados dos neurônios, sendo que, o neurônio a ser atualizado é escolhido randomicamente (SILVA, 2003).

Kohonen´s Feature Map Network,* ou *Self Organizing Map: As características de auto organização da rede Kohonen mapeia as conexões entre os neurônios de entrada de forma semelhante ao cérebro humano (Introduction to Artificial Neural Networks, 1995).

Adaptative Resonance Theory (ART): As redes baseadas na teoria de ressonância adaptativa são capazes de organizar os códigos de reconhecimento estável em tempo real, em resposta às sequências arbitrarias de padrões de entrada (Uhrig, 1995).

2.1.8 Outros algoritmos de treinamento

Redes neurais artificiais são utilizadas para detectar tendências que são muito complexas para serem detectadas por seres humanos.

Normalmente são utilizadas para encontrar padrões de classificação e extraí-los. Os principais algoritmos de treinamento são os de minimização, que nesse grupo, existem os algoritmos de minimização local e global.

Também existem os algoritmos evolutivos, tais como os algoritmos de evolução diferencial, (DE), otimização de enxame de partículas (PSO), colônia artificial de abelhas (ABC) e otimização de colônias de formigas (ACO). O algoritmo mais popular para treinamento, é o de *Backpropagation* por ser considerado um dos mais eficientes em desempenho.

2.1.9 Técnica de validação cruzada

Segundo Silva (2003), a essência da aprendizagem da rede neural artificial utilizando *backpropagation*, é realizar um mapeamento reajustando os pesos sinápticos. Para isso, é utilizada a clássica técnica da validação cruzada, muito útil durante o treinamento para determinar a capacidade de generalização da rede neural. Os dados para treinamento da rede são divididos em dois conjuntos distintos:

Conjunto de treinamento ou estimação: Utilizado para treinar a rede neural. Nesse conjunto, são apresentados os padrões de entrada com suas respectivas saídas desejadas.

Conjunto de Validação: Geralmente subdividido em dois conjuntos, sendo estes o conjunto de validação, no qual é apresentada uma quantidade de padrões de entrada com suas saídas desejadas, porém, sendo padrões de entrada não vistos pela rede durante o treinamento; e o conjunto de teste que apresenta entradas também não vistas anteriormente pela rede, mas sem apontar a saída. A quantidade de padrões de teste fica entre 10 e 25 por cento das amostras disponíveis.

Após o treinamento da rede com uma quantidade de épocas pré-definido (uma época significa a apresentação de todos os padrões de treinamento disponíveis), o treino interrompido e a rede é testada com os dados de validação e teste. O processo é repetido até que a rede neural apresente um valor de erro aceitável para o problema em questão.

2.2 SUPPORT VECTOR MACHINES

De acordo com a definição de Lorena e Carvalho (2007), SVM é uma técnica de aprendizado que vem recebendo crescente atenção da comunidade de Aprendizado de Máquina (AM). Os resultados da aplicação dessa técnica são comparáveis e muitas vezes superiores aos obtidos por outros algoritmos de aprendizado, como as Redes Neurais Artificiais (RNAs). Exemplos de aplicações de sucesso podem ser encontrados em diversos domínios, como na categorização de textos, na análise de imagens e em bioinformática.

As SVMs são embasadas pela teoria de aprendizado estatístico, desenvolvida por Vapnik (1995). Essa teoria estabelece uma série de princípios que devem ser seguidos na obtenção de classificadores com boa generalização, definida como a sua capacidade de prever corretamente a classe de novos dados do mesmo domínio em que o aprendizado ocorreu.

Fundamentada na Teoria da Aprendizagem Estatística, tem o intuito de resolver problemas de classificação de padrões.

Segundo Haykin (1999), a maquina de vetores suporte é uma outra categoria das redes neurais alimentadas adiante, ou seja, redes cujas sadas dos neurônios de uma camada alimentam os neurônios da camada posterior, não ocorrendo a realimentação

Esta técnica originalmente desenvolvida para classificação binária, busca a construção de um hiperplano com superfície de decisão, de tal forma que a separação entre os exemplos seja máxima. Isso considerando padrões linearmente separáveis.

Devido à sua eficiência em trabalhar com dados de alta dimensionalidade, é reportada na literatura como uma tecnica altamente robusta, muitas vezes comparada as Redes Neurais (SUNG e MUKKAMALA, 2003) e (DING e DUBCHAK, 2001).

2.2.1 Conceitos

As SVMs foram desenvolvidas com o objetivo de serem utilizadas na classificação binária. Pode ser feita uma abordagem superficial da seguinte forma:

Separação de classes: Basicamente, busca-se encontrar o hiperplano de separação ótima entre as duas classes, maximizando a margem entre os pontos mais próximos destas classes. Os pontos situados nos limites são chamados de vetores de suporte e o meio da margem é o hiperplano de separação ótima;

Sobreposição de classes: Os pontos de dados no lado "errado" da margem discriminante são ponderados para baixo para reduzir a sua influência "*margin soft*";

Não-linearidade: Quando não é possível encontrar um separador linear, os pontos de dados são projetados em uma dimensão superior de espaço, onde os pontos de dados se tornam efetivamente linearmente separáveis.

Solução do problema: toda a tarefa pode ser formulada como um problema de otimização quadrática que pode ser resolvido por meio de técnicas conhecidas.

Um programa capaz de realizar todas estas tarefas é conhecido como *Support Vector Machine*. A Figura 12 contém uma representação de uma classificação de amostras por meio de SVM.

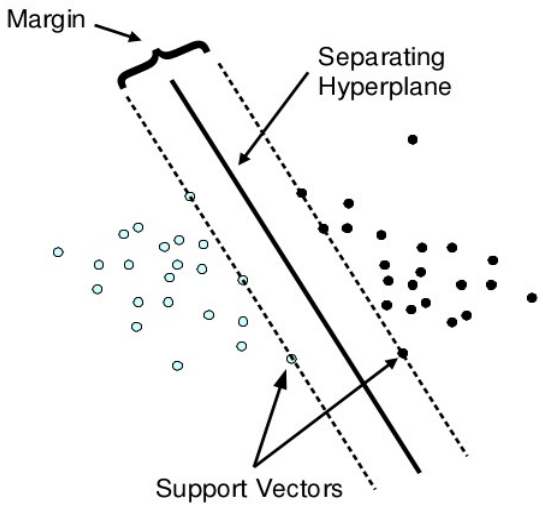

Figura 12 – Classificação de amostras com SVM.
Fonte: Meyer (2014).

2.2.2 Características das *Support Vector Machines*

Para Smola e Scholkopf (2002), algumas das principais características das SVMs são:

Capacidade de generalização: Os classificadores gerados por uma SVM em geral alcançam bons resultados de generalização. A capacidade de generalização de um classificador é medida baseado na sua eficiência na classificação de dados que não pertençam ao conjunto utilizado em seu treinamento. Na geração de preditores por SVM, busca-se evitar *overfitting*, onde o preditor se torna muito especializado no conjunto de treinamento, obtendo baixo desempenho quando confrontado com novos padrões.

Robustez em grandes dimensões: As SVMs possuem robustez diante de objetos de grandes dimensões, tais como imagens. Comumente ocorre *overfitting* nos classificadores gerados por outros métodos de inteligência artificial sobre esses tipos de dados.

Convexidade da função objetivo: A aplicação das SVMs implica na otimização de uma função quadrática, que possui apenas um mínimo global. Esta é uma vantagem sobre, por exemplo, as Redes Neurais Artificiais, em que há apresençaca de mínimos locais na função objetivo a ser minimizada.

Teoria bem definida: As SVMs possuem uma base teórica bem estabelecida dentro da Matemática e Estatística. Estes resultados foram apresentados por Vapnik e Chernovenk através da Teoria de Aprendizado Estatístico, proposta por estes autores nas décadas de 60 e 70 (Vapnik, 1995). As SVMs surgiram como resultado direto do emprego dos princípios apresentados nestes estudos. Apesar de sua teoria ser relativamente antiga, as primeiras aplicações práticas das SVMs são recentes, e datam da década de 90 (Hearst et al., 1998).

2.2.3 Teoria do aprendizado estatístico

De acordo com o exemplo de Smola e Schölkopf (2002), admitindo ser *f* um classificador e *F* o conjunto de todos os classificadores que um determinado algoritmo de aprendizado de máquina pode gerar. Esse algoritmo, durante o processo de aprendizado, utiliza um conjunto de treinamento *T* , $\hat{f}$ composto de n pares $(\mathbf{x_i}, y_i)$, para gerar um classificador particular $\hat{f} \in F$.

Observando o conjunto de treinamento da Figura 13, o objetivo do processo de aprendizado é encontrar um classificador que separe os dados das classes “círculo” e “triângulo”. As funções ou hipóteses consideradas são apresentadas na Figura 13 por meio de bordas , também denominadas fronteiras de decisão, traçadas entre as classes.

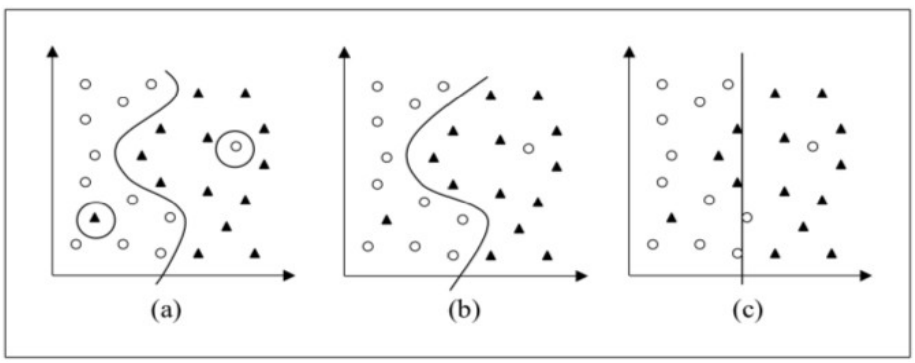

Figura 13 – Classificação de amostras em SVM lineares.
Fonte: Smola e Schölkopf (2002)

No primeiro exemplo da Figura 13(a) tem-se uma hipótese que classifica corretamente todos os exemplos do conjunto de treinamento, incluindo dois possíveis ruídos. Por ser muito específica para o conjunto de treinamento, essa função apresenta elevada suscetibilidade a cometer erros quando confrontada com novos dados. Esse caso representa a ocorrência de um supertreinamento do modelo aos dados de treinamento.

Outro classificador proposto poderia desconsiderar pontos pertencentes a classes opostas que estejam muito próximos entre si, mostrado na ilustração da Figura 13(c), representando essa alternativa. Essa hipótese considerada, porém, possui muitos erros, mesmo para casos que podem ser considerados simples. Tem-se assim a ocorrência de um subajustamento, pois o classificador não é capaz de se ajustar mesmo aos exemplos de treinamento.

Uma alternativa intermediária entre as duas funções descritas é representado no exemplo da Figura 13(b). Esse preditor tem complexidade intermediária e classifica corretamente grande parte dos dados, sem se fixar demasiadamente em qualquer ponto individual.

A Teoria de Aprendizado Estatístico (TAE) estabelece condições matemáticas que auxiliam na escolha de um classificador particular f a partir de um conjunto de dados de treinamento.

Essas condições levam em conta o desempenho do classificador no conjunto de treinamento e a sua complexidade, com o objetivo de obter um bom desempenho também para novos dados do mesmo domínio.

2.2.4 SVM lineares

As SVM lineares surgiram devido à utilização direta dos resultados obtidos através da TAE, sendo divididos em duas classes. A primeira formulação, mais simples, lida com problemas linearmente separáveis. Já a segunda trata da obtenção de fronteiras não lineares, por meio de uma extensão das SVMs lineares.

2.2.4.1 SVM com margens rígidas

De acordo com Lorena e Carvalho (2007), as SVMs lineares com margens rígidas definem fronteiras lineares a partir de dados linearmente separáveis. Sendo T um conjunto de treinamento com n dados $\mathbf{x}_i \in \mathrm{X}$ e seus respectivos rótulos $y_i \in \mathrm{Y}$, onde X é o espaço dos dados e Y = {−1, +1}. Então T é linearmente separável se for possível separar os dados das classes +1 e −1 por um hiperplano.

Classificadores que separam os dados por meio de um hiperplano são denominados lineares. A equação de um hiperplano é apresentada na Equação 2.1 , em que w · x é o produto escalar entre os vetores *w* e *x*, $w \in X$ é o vetor normal ao hiperplano descrito.

As restrições são impostas para garantir que não haja dados de treinamento entre as margens de separação das classes. Por esse motivo, a SVM obtida possui também a nomenclatura de SVM com margens rígidas, Equação (2.1).

$$f(x) = wx + b = 0 \tag{2.1}$$

De forma semelhante às RNAs, o erro obtido é o quadrático, cuja solução se baseia em uma ampla teoria matemática. Sendo a função objetivo minimizada convexa e os pontos que satisfazem as restrições formam um conjunto convexo, esse problema possui um único mínimo global.

Problemas desse tipo podem ser solucionados com a introdução de uma função Lagrangiana, que engloba as restrições à função objetivo, associadas a parâmetros denominados multiplicadores de Lagrange α_i , Equação (2.2)

$$L(w, b, \alpha) = \frac{1}{2}||w||^2 - \sum_{i=1}^{n} \alpha(y_i(w \cdot x_i + b) - 1) \tag{2.2}$$

2.2.4.2 SVM com margens suaves

Normalmente é difícil encontrar aplicações cujos dados sejam linearmente separáveis. Isso se deve a diversos fatores, como a presença de ruídos e *outliers* (valores atípicos) nos dados ou à própria natureza do problema, que pode ser não linear. Dessa forma, as SVMs lineares de margens rígidas são estendidas para lidar com conjuntos de treinamento mais gerais. Esse processo é feito com a introdução de variáveis de folga ξ_i, para todo $i = 1, \ldots, n$. Essas variáveis relaxam as restrições impostas ao problema de otimização primal, que gera a Equação (2.3):

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \xi \geq 0. \forall_i = 1....n \tag{2.3}$$

A aplicação desse procedimento suaviza as margens do classificador linear e permitindo também a que ocorram alguns erros de classificação. Por esse motivo, as SVMs obtidas também podem ser referenciadas como SVMs com margens suaves. Um erro no conjunto de treinamento é indicado por um valor de ξ_i maior que 1. Logo, a soma dos ξ_i representa um limite no número de erros de treinamento.

2.2.5 SVM não-lineares

De acordo com Müller *et al.* (2001), as SVMs lineares tem bom desempenho na classificação de conjuntos de dados considerados linearmente separáveis, ou que possuam uma distribuição aproximadamente linear, sendo que a versão de margens suaves tolera a presença de alguns ruídos e *outliers*. Porém, há muitos casos em que não é possível dividir de forma satisfatória os dados de treinamento através de um hiperplano. Na Figura 14 é apresentado um caso em que o uso de uma fronteira curva poderia obter melhores resultados na separação das classes, sendo (a) conjunto de dados não-linear (b) fronteira não-linear no espaço de entradas e (c) fronteira linear no espaço de características.

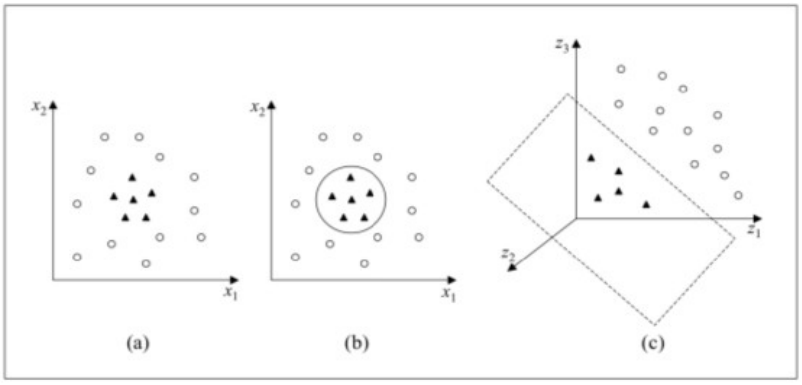

Figura 14 – Utilização de fronteira curva em separação de classes.
Fonte: Müller *et al.* (2001).

As SVMs tratam de problemas não lineares mapeando o conjunto de treinamento de seu espaço original, sendo referenciado como de entradas, para um novo espaço de maior dimensão, denominado espaço de características (*feature space*).

Seja $\Phi : X \rightarrow \Im$ um mapeamento, em que X é o espaço de entradas e $\Im$ denota o espaço de características. A escolha apropriada de Φ faz com que o conjunto de treinamento mapeado em possa ser separado por uma SVM linear.

O uso desse procedimento, de acordo com Haykin (1999), é baseado no teorema de Cover. Sendo um conjunto de dados não linear no espaço de entradas X, esse teorema afirma que X pode ser transformado em um espaço de características $\Im$, onde há alta probabilidade dos dados serem linearmente separáveis.

Para isso duas condições devem ser satisfeitas. A primeira é que a transformação seja não linear, enquanto a segunda é que a dimensão do espaço de características seja suficientemente alta.

2.2.6 Funções de *Kernel*

Segundo Gonçalves (2010), as funções de *kernel* possuem o objetivo de projetar os vetores com características de entrada em um espaço de características de alta dimensão, para a classificação de problemas que se encontram em espaços não linearmente separáveis. Esse processo é realizado pois, devido à medida em que se aumenta o espaço da dimensão

do problema, aumenta também a probabilidade desse problema se tornar linearmente separável em relação a um espaço de baixa dimensão. Porém, para obter uma boa distribuição para esse tipo de problema é necessário um conjunto de treinamento com um elevado número de instâncias.

Na Figura 15 é apresentado o processo de transformação de um problema não linearmente separável em um problema linearmente separável através do aumento da dimensão, onde é feito um mapeamento por uma função de *kernel F(x).*

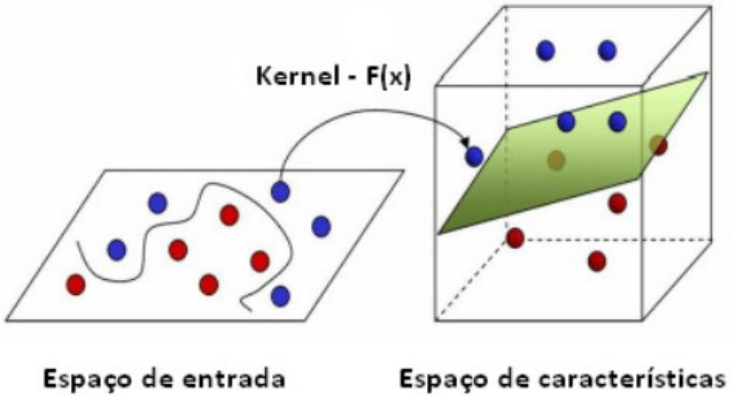

Figura 15 – Transformação de um problema não-linearmente separável para linearmente separável.
Fonte: Rebelo (2008).

Uma função para ser definida como função de *kernel*, deve obedecer a seguinte teoria de Hilbert RKHS (*Reproducing Kernel Hilbert Space*), como demonstrado na Equação 2.4:

$$K(x_i, x_j) = \langle \Phi(x_i \cdot \Phi(y_i)) \rangle \quad (2.4)$$

onde φ(.) deverá pertencer a um domínio onde seja possível realizar o calculo do produto interno.

Segundo Gonçalves (2010), essas funções também satisfazem as condições do Teorema de Mercer , o qual determina que uma determinada função é definida como sendo de *kernel*, se a matriz K for positivamente definida (valores maiores que zero), onde K é obtida pela Equação 2.5:

$$K = K_{ij} = \kappa(x_i, x_j) \tag{2.5}$$

Essas funções de *kernel* também são conhecidas como *Kernels de Mercer.* Há vários tipos de *kernels* que podem ser utilizadas, porém, as mais conhecidas são apresentadas na Tabela 1:

Tabela 1 – Funções de *Kernel*

Tipo de *Kernel*	Função $\kappa(x_i, x_j)$	Tipo do Classificador
Polinomial	$(\langle x_i \cdot x_j \rangle + 1)^p$	Máquina de aprendizagem polinomial
Gaussiano ou RBF	$exp\left(-\frac{\|\|x_i - x_j\|\|^2}{2\sigma^2}\right)$	Rede RBF
Sigmoidal	$tanh(\beta \langle x_i \cdot x_j \rangle) + \beta_1$	Perceptron de duas camadas

2.2.6.1 *Kernel* Gaussiano

De acordo com Almeida (2007), o *kernel* RBF é bastante utilizado para solucionar de problemas de aprendizagem, também sendo usado como padrão em muitas bibliotecas de linguagens de programação que utilizam o algoritmo SVM.

Na máquina RBF, ao contrário do *kernel* linear, é possível resolver problemas não linearmente separáveis, por meio do mapeamento para um espaço de maior dimensão. Existem dois parâmetros que variam em busca de um melhor resultado para o aprendizado do classificador, sendo: γ (gamma) e C (custo). No *kernel* RBF o número de funções radiais e os seus respectivos centros são definidos pelos vetores suporte obtidos. Quando o valor de gamma é muito pequeno, o modelo é muito limitado e não pode captar a complexidade dados. A região de influência de qualquer vetor de suporte selecionado incluiria todo o

conjunto de treinamento. O modelo resultante irá comportar-se de forma semelhante a um modelo linear com um conjunto de hiperplanos que separam os centros de alta densidade de qualquer par de duas classes.

A Figura 16 contém a demonstração da aplicação de uma máquina RBF, onde é possível ver um aglomerado de pontos que representam as instâncias das classes, e a cor de fundo indica a qual classe pertence determinado conjunto.

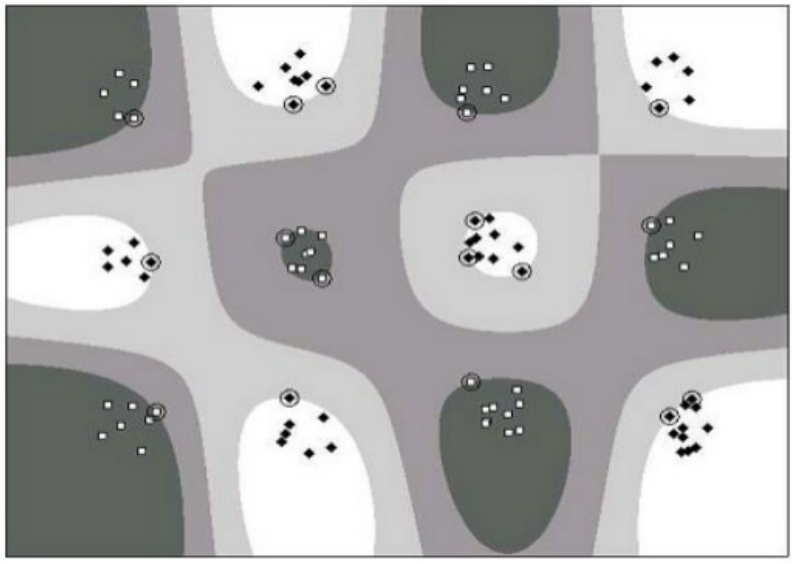

Figura 16 – Separação de classes com utilização de *kernel* RBF.
Fonte: Almeida (2007).

Segundo Almeida (2007), outra razão para a utilização do *kernel* RBF é o baixo número de parâmetros (dois) que influenciam o resultado do aprendizado. No *kernel* polinomial são usados mais parâmetros que o RBF: C (custo), γ (gamma) e o grau (degree). Por exemplo, se C assumisse 11 como valor, γ assumisse o valor de 10 no *kernel* RBF, resultaria em uma combinação de valor 110, adicionando-se o grau com valor 5, a combinação teria valor de 550, aumentando de forma considerável o desempenho e complexidade do classificador.

2.3 PROTOCOLOS DE REDES DE COMPUTADORES

2.3.1 Histórico

Segundo Bonaventure (2011), quando os primeiros computadores foram desenvolvidos, em meados da Segunda Guerra Mundial, eles eram máquinas muito caras e isoladas. No então, depois de vinte anos, os preços gradualmente caíram, e então, começaram a surgir os primeiros experimentos buscando a conexão entre as máquinas.

Em 1969, o Departamento de Defesa dos Estados Unidos (DoD) iniciou o desenvolvimento da Arpanet, que foi o protótipo da atual Internet. A Arpanet foi desenvolvida com finalidade militar e foi continuada até metade da década de 1980. Na França, Louis Pouzin desenvolveu a rede Cyclades. Durante a década de 1970, muitos pesquisadores se empenharam em desenvolver formas de conexão entre computadores, e também começou a surgir o interesse das industrias de computadores e de comunicações.

A indústria partiu para uma aproximação diferente para o projeto das redes LAN (*Local Area Network*). Muitas tecnologias para redes LAN surgiram, entre elas a Token Ring e a Ethernet. Na década de 1980, a necessidade de interconectar muitos computadores incentivou fabricantes a desenvolverem seus próprios protocolos, tais como o XNS da Xerox, NetBIOS da Microsoft, DECNet, da DEC, o SNA da IBM e a Appletalk da Apple. Na comunidade de desenvolvimento, a Arpanet foi descontinuada dando lugar ao TCP/IP desenvolvido dentro do BSD Unix.

Universidades que já utilizavam sistemas Unix adotaram facilmente o protocolo TCP/IP, e vendedores de máquinas com sistema Unix tais como Sun Silicon Graphics incluíram o TCP/IP nas suas variantes do Unix. De forma paralela a International Standardization Organization (ISO) com suporte de alguns governos, trabalhou no desenvolvimento de um conjunto aberto de protocolos de rede. Ao final, o TCP/IP de fato se tornou o protocolo padrão, não sendo restrito à comunidade de pesquisa.

2.3.2 Pacotes de dados

Para Torres (2001), pacotes de dados são formados por conjunto de bytes que carregam informações que o auxiliam a chegar a seu destino, tais como, entre outras características, os endereços IP de origem e destino, as portas de entrada e saída, tamanho do pacote e seu conteúdo.

A Figura 17 apresenta um esquema com os campos de um pacote de dados TCP/IP.

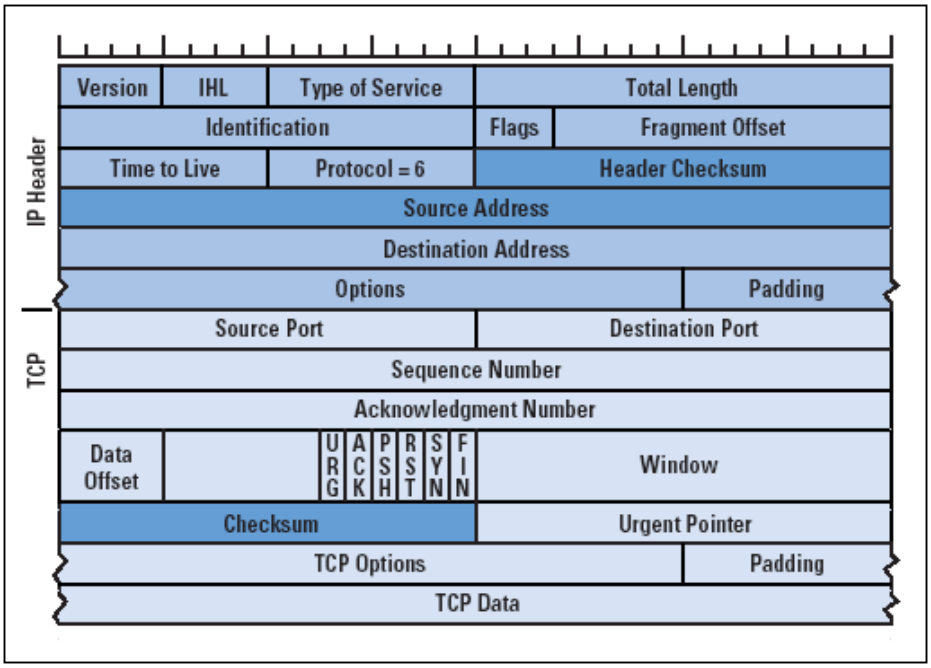

Figura 17 – Esquema de pacote de dados TCP/IP.
Fonte: Huston (2004).

Version: armazena a versão do protocolo a que o datagrama pertence, na versão 4 possui tamanho de 4 bits (0100).

IHL, Internet Header Lenght: representa o comprimento do cabeçalho da Internet, com o número de palavras de 32 bits no cabeçalho IPv4.

Type of service: especifica uma preferência para como os datagramas poderiam ser manuseados assim que circulariam pela rede. Na prática, o campo ToS não foi largamente implementado.

Total Length: campo de dezesseis bits do IPv4, define todo o tamanho do datagrama, incluindo cabeçalho e dados, em bytes de oito bits. O datagrama de tamanho mínimo é de vinte bytes, e o máximo é 64 Kb.

Identification: campo de identificação de 16 bits. Este campo é usado principalmente para identificar fragmentos identificativos do datagrama IP original.

Flags: campo utilizado para identificar e controlar fragmentos, o campo é composto por três bits.

Offset: Campo constituído por treze bits, o *offset* permite que o receptor do datagrama determine o local do fragmento do pacote IP original.

TTL, *Time to Live*: determina o tempo de vida do pacote e é composto por oito bits. O campo tem a funcionalidade de evitar que o pacote fique percorrendo em loop sem encontrar o destino final, evitando que o pacote persista. Sem o campo TTL, haveriam muitas requisições na Internet, o que causaria uma enorme lentidão da rede, ou até mesmo, o travamento de roteadores. O campo TTL limita a vida útil do pacote em segundos, onde cada roteador que o pacote atravessa, é decrementado o valor do TTL.É iniciadoem 128 e quando o valor chega a 0, o pacote é descartado. O TTL é considerado como uma contagem de *hops* (saltos).

Protocol: campo que possui oito bits e é neste campo que é realizado a definição de qual é o protocolo usado na porção de dados de um pacote IP. Existe a possibilidade de ser o TCP, mas há também o UDP e outros. A numeração que se aplica a todos protocolos da Internet é definida na RFC 1700.

Checksum: responsável por detectar inconsistência no datagrama IP, realizando uma checagem cíclica de todos os campos de um datagrama e identificar se nos *hops* (saltos), onde o datagrama percorreu, se não houve nenhuma falha.

Campos de origem e destino (*source* e *destination address*): campos que contém os endereços de IP de origem e destino do pacote.

Options: o campo de opções começa com um pequeno campo de 8 bits que permite saber quais opções são usados no pacote. As opções são listados na tabela de opções de TCP.

Padding: possui bits variáveis e é utilizado para determinar o limite do cabeçalho em até 32 bits. O campo deve ser preenchido com zeros para ajustar o seu tamanho, até o final.

Source* e *destination port: Determinam os numeros da porta de origem e destino do pacote. São esses numeros que identificam o protocolo de rede do pacote transmitido.

Sequence Number: Determina a ordem de sequência dos pacotes enviados para serem organizados no computador de destino, e apresentados na ordem correta.

ACK: Quando o bit ACK (*acknowledgement* ou reconhecimento) é definido, este segmento está servindo como um reconhecimento (além de outras atribuições possíveis) e este campo contém o número de seqüência da próxima origem esperando para enviar um pacote.

Data Offset: Especifica o número de palavras de dados no cabeçalho TCP de 32 bits. Em outras palavras, este valor é igual a quatro vezes o número de bytes no cabeçalho, que deve ser sempre um múltiplo de quatro. Ele é chamado de "desvio de dados", uma vez que indica quantas palavras de 32 bits do início dos dados é deslocado a partir do início do segmento de TCP.

Window: Indica o número de octetos de dados que remetente deste segmento está disposto a aceitar a partir do destino de uma só vez. Isso normalmente corresponde ao tamanho atual do buffer alocado para aceitar dados para esta conexão.

Checksum: Campo utilizado para verificar a integridade do pacote após a transmissão.

Urgent Pointer: Ponteiro que determina o nível de urgência, ou prioridade n transmissão do pacote.

Dados: Contém os bits com as informações de fato transportados pelo pacote.

2.3.3 A Internet

A Internet como conhecemos não surgiu em um momento único. No ano de 1908, Nikola Tesla havia predito uma tecnologia que surgiria em determinado momento, na qual “um homem de negócios em Nova York diria suas instruções e imediatamente elas apareceriam em seu escritório em Londres ou em qualquer outro lugar”, ou permitiria um acesso global a “figuras, palavras, desenhos ou imagens”.

Também não surgiu de forma repentina com a estrutura que conhecemos. As primeiras pesquisas na área surgiram nas décadas de 1950 e 1960. Sendo a mais notável, o desenvolvimento da Arpanet, pelo Departamento de Defesa dos Estados Unidos

O protocolo de controle de rede NCP, o primeiro a ser aplicado sobre a Arpanet, e foi utilizado como protocolo de comunicação simétrica host a host. Essa pilha de protocolos ficava nas próprias máquinas e incluía a hierarquia de camadas dos protocolos para a implementação de protocolos mais complexos. Com o desenvolvimento do NCP, os usuários da rede puderam iniciar o desenvolvimento de novas aplicações. Este protocolo futuramente seria substituído pelo protocolo TCP/IP. (KLEINROCK, 2010).

2.3.4 O protocolo TCP/IP

Segundo a Fielding et. al. (1999), em uma visão geral, o termo “TCP/IP” pode significar muitas coisas relacionadas com os protocolos específicos do TCP e do IP. Isso pode incluir outros protocolos, aplicações, e todo o meio da rede. Alguns exemplos desse protocolos são: UPD, ARP, e ICMP. Exemplos dessas aplicações são: TELNET, FTP, e RCP.

O protocolo de controle de transmissão TCP, descrito na RFC 1180, oferece um serviço diferente do UDP. TCP oferece uma conexão de fluxo de bytes orientada, em vez de uma entrega de datagrama sem conexão de serviço. O TCP garante a entrega, enquanto que UDP não.

O TCP é usado por aplicativos de rede que exigem entrega garantida e não pode ser comprometido com a realização de time-outs e retransmissões. As duas aplicações de rede mais comuns que usam TCP são FTP e o TELNET. Outras aplicações de rede TCP populares incluem sistema X-Window, RCP (cópia remota). Mas essa maior capacidade do TCP não é sem custo: requer mais CPU e largura de banda de rede. Os módulos internos do TCP são muito mais complicadas do que as de um módulo UDP.

2.3.5 O modelo de camadas OSI

No início do desenvolvimento das redes de computadores, as soluções em sua maioria eram proprietárias, onde as tecnologias eram utilizadas apenas por seus fabricantes. Isso impossibilitava a comunicação entre sistemas diferentes. Para facilitar a intercomunicação entre sistemas diferentes, a *International Standars Organization*, ISSO, desenvolveu um modelo de referência chamado OSI, *Open Systems Interconnection*, para que os fabricantes de sistemas pudessem ter um modelo para seus protocolos.

O modelo OSI é dividido em sete camadas, como o representado na Figura 18, no qual durante a transmissão de dados, cada camada pega as informações enviadas pela camada superior, acrescenta informações (cabeçalho), e repassa à camada inferior. Esse processo é conhecido como encapsulamento. Na camada de transporte, número 4, os dados enviados pela aplicação são divididos em pacotes, na camada de enlace, número 2, os pacotes são divididos

em quadros. No recebimento dos pacotes, este processo é feito de maneira inversa. A Figura 18 contém a representação da pilha de protocolos do modelo de camadas OSI.

Figura 18 – Camadas do modelo OSI.
Fonte: Torres (2001).

2.3.5.1 Camada de aplicação

A camada de aplicação é responsável por fazer a interface entre o protocolo de comunicação e o aplicativo que requisitou ou receberá a informação através da rede. Por exemplo, um arquivo baixado da rede será recebido por um aplicativo FTP, e este entrará em contato com a camada de aplicação para efetuar esse pedido.

2.3.5.2 Camada de apresentação

Esta camada é responsável por converter o formato do dado recebido pela camada de aplicação, para um formato utilizado na transmissão do dado, ou seja, um formato entendido pelo protocolo utilizado. Essas marcações indicam se houve falhas na transmissão dos pacotes, solicitando o reenvio, a partir da última marcação recebida pelo computador receptor.

2.3.5.3 Camada de sessão

A camada de sessão permite a comunicação entre aplicações que estejam sendo executadas em computadores distintos, estabelecendo uma sessão de comunicação. Na qual, as aplicações definem de que forma a transmissão de dados deve ser feita, e coloca marcações nos dados a serem transmitidos. Essas marcações são responsáveis pela indicação de possíveis falhas no recebimento das informações enviadas. Com base na última marcação recebida pelo computador de destino, este solicita o reenvio das informações, a partido desta marcação final.

2.3.5.4 Camada de transporte

A camada de transporte divide em pacotes os dados recebidos da camada de sessão, que serão transmitidos pela rede, ou repassados para a camada de rede. No destinatário, a camada de transporte é responsável por remontar os pacotes recebidos pela camada de rede, e enviá-los à camada de sessão. Também nesta camada é feito o controle de fluxo, onde os pacotes são colocados em ordem, no caso de terem chegados fora de ordem, correção de erros, enviando ao transmissor uma informação de reconhecimento, *ackonwledge*, informando se o pacote foi recebido com sucesso.

Esta camada separa as camadas de nível de aplicação (aplicação, apresentação e sessão) e as camadas de nível físico (rede, enlace de dados e física).

2.3.5.5 Camada de rede

A camada de rede é responsável por fazer o endereçamento dos pacotes, convertendo os endereços lógicos em endereços físicos, permitindo assim, que os pacotes cheguem ao seu destino correto. Essa camada também determina quais rotas os pacotes devem seguir para

chegar ao seu destino, no caso de haver mais de uma possibilidade de rotas entre a origem e o destino. As rotas são determinadas com base em parâmetros como prioridade e condições de trafego.

2.3.5.6 Camada de enlace de dados

A camada de enlace de dados, também conhecida como camada de link de dados, transforma os pacotes de dados recebidos da camada de rede, adicionando a ele informações como o endereço da placa de rede de origem e destino (*mac address*), dados de controle, o conteúdo do pacote, entre outros. O quadro criado pela camada de enlace é enviado para a camada física e transformado em sinais elétricos que serão enviados pelos cabos de rede.

2.3.5.7 Camada física

A camada física transforma os quadros recebidos da camada de enlace em sinais compatíveis com o meio por onde os dados devem ser transmitidos. No caso de serem transmitidos por meio elétrico, cabos, os dados binários (0 e 1) são convertidos em sinais elétricos. No caso de ser um meio óptico (fibra óptica), os dados são convertidos em sinais luminosos representando 0 e 1. Os processos dessa camada são executados pela placa de rede do dispositivo que possui acesso à rede.

2.3.6 O modelo de camadas TCP/IP

O protocolo TCP/IP é atualmente o protocolo mais utilizado em LAN, as redes locais. Isso é devido em grande parte à popularização da Internet, já que o TCP/IP foi o protocolo desenvolvido para ser utilizado por ela. Atualmente, até os sistemas operacionais que possuem

protocolos de redes proprietários, tais como o Windows NT com o NetBEUI, ou o Netware como IPX/SPX, possuem suporte ao TCP/IP (TORRES,2001).

Uma grande vantagem do TCP/IP em relação aos outros protocolos, é de ser um protocolo roteável, ou seja, foi planejado já tendo em vista redes de computadores em longas distâncias, onde podem haver vários caminhos para os pacotes de dados seguirem para chegar ao seu destino.

Outra característica responsável pela sua ampla utilização, é o fato dele possuir arquitetura aberta, o que significa que qualquer fabricante pode utilizá-lo e modifica-lo conforme suas necessidades sem precisar pagar direitos autorais a ninguém. Dessa forma, ele acabou se tornando um protocolo universal, possibilitando a comunicação entre todos os sistemas sem dificuldades.

Em termos de arquitetura, o TCP/IP é um protocolo de quatro camadas. Na Figura 19 é apresentada uma comparação entre camadas do modelo TCP/IP e as camadas do modelo OSI.

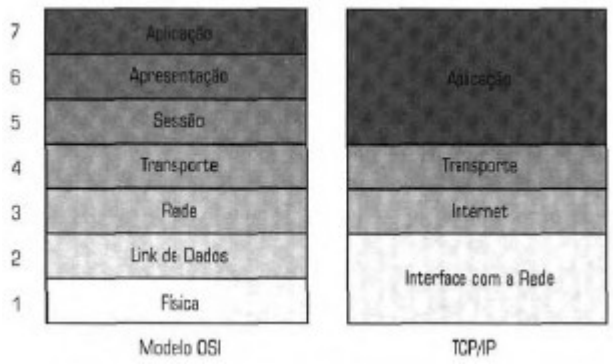

Figura 19 – Correlação entre as camadas dos modelos OSI e TCP/IP.
Fonte: Torres (2001).

O TCP/IP pode ser visto como um conjunto de protocolos, sendoos mais conhecidos são o *Transmission Control Protocol*, do inglês Protocolo de Controle de Transmissao, TCP e o *Internet Protocol*, do inglês Protocolo de Internet, IP.

2.3.6.1 Camada de aplicação

Esta camada é equivalente ás camadas 5,6 e 7 do modelo OSI e é responsável pela comunicação entre os aplicativos e o protocolo de transporte. Há vários protocolos na camada

de aplicação, sendo os mais conhecidos o *Hypertext Transfer Protocol* HTTP, *Simple Mail Transfer Protocol* SMTP, o *File Transfer Protocol* FTP, o *Secure Shell* SSH. Neste trabalho, os protocolos analisados foram o FTP, HTTP e SSH, explicados mais detalhadamente nos próximos tópicos.

A utilização de um número de porta permite que o protocolo de transporte, geralmente o TCP, saiba qual o tipo de conteúdo do pacote de dados, e no receptor, identificar para qual tipo de aplicação o pacote deve ser entregue. Por exemplo: O protocolo HTTP utiliza a porta 80, o que direciona o pacote para uma aplicação especifica para tratar esse pacote (TORRES, 2001).

2.3.6.2 Camada de transporte

A camada de transporte do TCP/IP é correspondente direto da camada número 4 do modelo OSI. Esta camada é a responsável por transformar os dados enviados pela camada de aplicação e transformá-los em pacotes para repassar, posteriormente, à camada de Internet.

No protocolo TCP/IP é utilizado um esquema chamado de multiplexação, o que possibilita a transmissão “simultânea” de dados por diferentes aplicações. Mas o que acontece em realidade, é a intercalação de pacotes, onde várias aplicações podem estar se comunicando com a rede no mesmo momento, mas os pacotes são enviados à rede de forma intercalada, não havendo assim, a necessidade de aguardar o encerramento de uma aplicação para iniciar outra.

Nesta camada operam dois protocolos, sendo o TCP e o *User Datagram Protocol*, o UDP. Ao contrário do TCP, que verifica se o pacote chegou ao destino (se não chegou, ele reenvia o mesmo pacote), o UDP não possui essa verificação de erro. O TCP, por possuir essa verificação, é mais utilizado na transmissão de dados, enquanto o UDP fica responsável pela transmissão de informações de controle.

2.3.6.3 Camada de internet

A camada de internet do modelo TCP/IP equivale à camada 3 do modelo OSI. Nessa camada, os pacotes são endereçados, transformando os endereços lógicos, em endereços físicos, fazendo com que estes possam chegar em seu destino. A camada de Internet também determina quais rotas o pacote deve percorrer para atingir o destino, baseado em parâmetros como condições de tráfego e níveis de prioridade.

Na transmissão de um pacote de dados, este é dividido em divido em pacotes menores chamados datagramas. Os datagramas são enviados para a camada de interface com a rede, onde são transmitidos pelo cabeamento de rede através de quadro. Esta camada não é responsável por fazer a verificação do recebimento do pacote pelo destino, cabendo isso ao protocolo TCP, da camada de transporte.

2.3.6.4 Camada de interface com a rede

Esta camada corresponde às camadas 1 e 2 do modelo OSI, sendo responsável pelo envio dos datagramas recebidos pela camada de Internet, em forma de quadros, através da rede. Na Figura 20 é mostrado o esquema completo do funcionamento do protocolo TCP/IP.

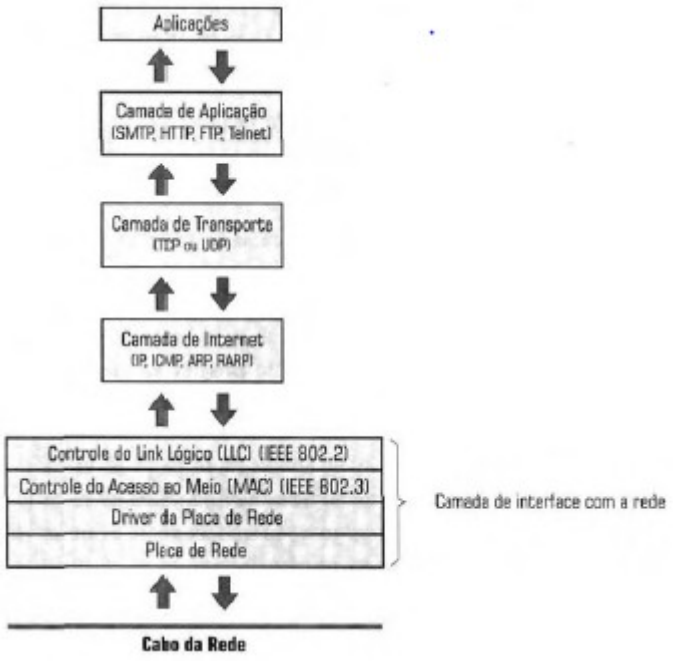

Figura 20 – Estrutura de camadas do modelo TCP/IP.

Fonte: Torres (2001).

2.3.7 O protocolo HTTP

De acordo com a definição dada por Socolofsky e Kale (1991), o protocolo de transferência de hipertexto HTTP, é um protocolo da camada de aplicação para sistemas colaborativos, distribuídos e de informação hipermídia. O HTTP tem sido amplamente utilizado pela iniciativa global da World Wide Web desde o ano de 1990. A primeira versão do HTTP, referida como HTTP 0.9, foi um protocolo simples para transporte de dados através da Internet. O HTTP 1.0, definido na RFC 1945, incrementou o protocolo permitindo mensagens contento metainformação sobre o dado transferido e modificadores da semântica de requisição e resposta.

A comunicação HTTP geralmente ocorre através de conexões TCP/IP. A porta padrão é TCP 80, mas outras portas podem ser usadas. Isso não impede o HTTP de ser implementado em cima de qualquer outro protocolo na Internet, ou em outras redes. O HTTP apenas pressupõe um transporte confiável, qualquer protocolo que fornece tais garantias pode ser usado, o mapeamento do pedido HTTP/1.1, definido na RFC 2616, e estruturas de resposta para as unidades de transporte do protocolo em questão de dados está fora do escopo desta especificação.

2.3.8 O protocolo FTP

O protocolo para transferência de arquivos FTP, RFC 765, é tão antigo quanto o TELNET, também utiliza o TCP e tem ampla interoperabilidade. A operação e aparência é tal como executar um comando TELNET em um computador remoto. Mas em vez de executar seus comandos habituais, é necessário executar uma pequena lista de comandos para listagem de diretórios e afins.

Os comandos FTP permitem efetuar a cópia de arquivos entre computadores, e especificam os parâmetros para a conexão de dados (número da porta, modo de transferência,

tipos de representação e estrutura) e a natureza da operação do sistema de arquivos (armazenamento, recuperação, anexação, exclusão de arquivos, entre outros). O cliente e o servidor FTP iniciam a conexão e a transferência de arquivos com base nos parâmetros especificados (POSTEL e REINOLDS, 1985).

2.3.9 O protocolo SSH

O protocolo SSH, definido na RFC 4251 é utilizado para o acesso remoto seguro de máquinas e outros serviços de executados em uma rede considerada insegura. O SSH é composto de três principais componentes.

1. O protocolo de camada de transporte, que fornece autenticação do servidor, confidencialidade e integridade. Também pode fornecer de forma opcional a compressão de dados. A camada de transporte normalmente é executada sobre uma conexão TCP/IP, mas também pode ser utilizada por cima de qualquer tipo de fluxo de dados.
2. O protocolo de autenticação de usuário, que autentica o cliente que acessa o servidor. Esse protocolo é executado sobre o protocolo de autenticação.
3. O protocolo de conexão fornece a multiplexação da criptográfica do túnel de comunicação em vários canais lógicos. Esse protocolo é executado sobre o protocolo de autenticação de usuário.

O cliente envia uma requisição de serviço uma vez que a conexão segura é estabelecida. Uma segunda requisição é enviada após a autenticação do usuário ser completada. Isso permite que novos protocolos sejam definidos e coexistam com os protocolos listados acima (YLONEN, LONVICK,2006).

3 DESENVOLVIMENTO DO TRABALHO

3.1 APRESENTAÇÃO DO ESTUDO DE CASO

Nesta seção serão apresentados os detalhes do experimento realizado, procedimentos e tecnologias utilizadas.

3.2 PROCEDIMENTOS

3.2.1 Configurações iniciais de ambiente

De forma inicial, foi necessário construir o ambiente de desenvolvimento, com a criação de duas máquinas virtuais com o sistema operacional Microsoft Windows 7, operando sobre a plataforma de virtualização Citrix XenServer. Esta plataforma, desenvolvida e mantida pela empresa norte-americana Citrix Systems, é um software de código aberto, utilizado para gerenciar estruturas virtuais em nuvem, servidores e desktop (CITRIX, 2013).

Após este processo, foi realizada a configuração do ambiente de desenvolvimento Java, com a instalação do ambiente de desenvolvimento integrado Eclipse. O Eclipse IDE, do inglês Integrated Development Environment, é um ambiente de desenvolvimento que foi iniciado pela IBM, e depois doado à comunidade open-source. Apesar de ser amplamente utilizado para a programação com a linguagem Java, o Eclipse fornece suporte a diversas linguagens de programação, como C, C++, PHP, entre outras (ECLIPSE FOUNDATION, 2014).

Para a simulação do tráfego de pacotes de dados foram configuradas aplicações do tipo cliente-servidor, tais como as seguintes:

Para o tráfego de dados HTTP foi utilizado o Wamp Server, sendo este um conjunto de ferramentas para aplicações web voltadas para a plataforma Microsoft Windows, utilizando o servidor web Apache2, PHP, e banco de dados MySQL (Bourdon, 2014).

A aplicação servidor utilizada para o tráfego de pacotes FTP foi o Filezilla FTP Server, sendo esta, parte da solução FTP de código aberto Filezilla. A aplicação cliente utilizada é o Filezilla Client e permite o download e upload de dados para o servidor FTP (Filezilla, 2014).

Para realizar o tráfego de dados SSH foram utilizadas as ferramentas Bitvise SSH Server e Bitvise SSH Client, que são softwares proprietários que fornece alternativas para o acesso remoto de forma segura, criptografando os dados durante o envio (Bitvise, 2014).Já a aplicação Bitvise24

SSH Client, fornece um emulador de terminal para acesso remoto, interface gráfica para tráfego de dados Secure File Transfer Protocol (SFTP), fornece uma ponte para o tráfego FTP para SFTP, e outros recursos voltados ao acesso remoto (Bitvise, 2014).

Para a implementação das RNA foi utilizada uma aplicação específica para este fim. A aplicação utilizada foi o SNNS, Stuttgart Neural Network Simulator, desenvolvido pela Universidade de Stuttgart na Alemanha.

O SNNS é um simulador para redes neurais artificiais e possui a implementação de funções de criação, treinamento e teste das RNA. Também possui um interpretador de linha de comando e uma linguagem de programação chamados batchman(MAMIER,WEILAND, 2014). Outra aplicação utilizada é o JavaNNS, Java Neural Neural Network Simulator, desenvolvido pela Universidade de Tübingen, também Alemanha.

O JavaNNS é considerado o sucessor do SNNS e possui uma interface gráfica desenvolvida na linguagem Java, além de todas as funcionalidades do SNNS. A Figura 21 contém a imagem da interface gráfica do JavaNNS.

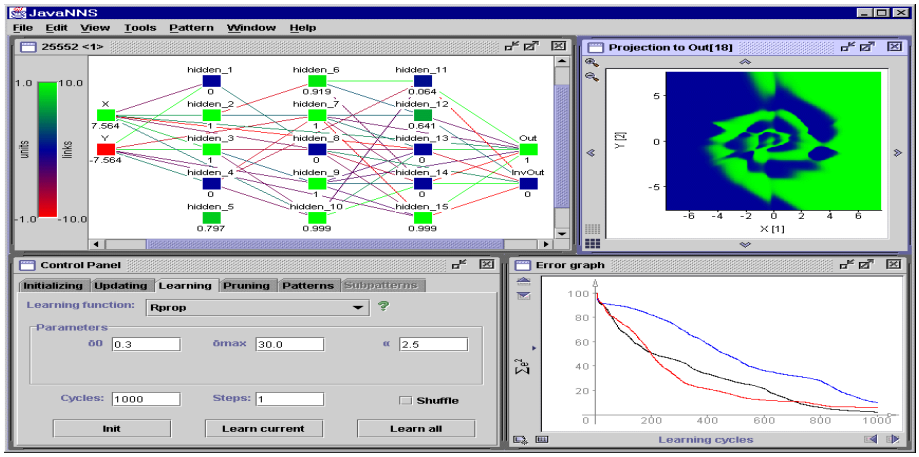

Figura 21– Interface gráfica da aplicação JavaNNS.
Fonte: Sly Technologies (2014).

3.2.2 Geração das amostras de pacotes de dados

Para gerar as amostras de dados utilizadas nos processos de treinamento, validação e testes das redes neurais, foi necessário realizar a captura dos pacotes de dados. Para isto, foi desenvolvida uma aplicação do tipo *sniffer*. Segundo Qader et al. (2010), *sniffer* é um programa executado em um computador conectado a uma rede, que passivamente recebe cópias dos pacotes que passam em seu adaptador de rede, salvando-os em arquivos para análises posteriores.

A Figura 22 contém um diagrama simplificado de uma rede de computadores com um computador conectado que possui um *sniffer* ativo, que recebe cópias dos pacotes que trafegam por esta rede. Neste diagrama, o notebook com a legenda recebe cópia do pacote enviado pelo PC1 ao PC3.

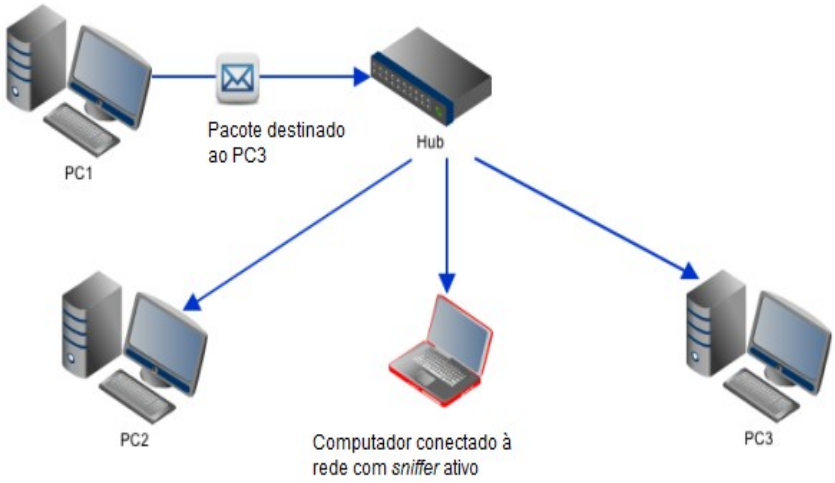

Figura 22 – Diagrama representando o funcionamento de um S*niffer*.
Fonte: Villanueva (2014).

No desenvolvimento do *sniffer*, foi utilizada a biblioteca Java JNetPcap, que fornece ferramentas para chamada de funções de aplicações nativas (instaladas na máquina) do Libpcap. O Libpcap é uma interface que opera de forma independente do sistema operacional para a captura de pacotes de dados. Fornece um conjunto de classes para o monitoramento baixo-nível da rede (GARCIA, 2014).

O JNetPcap possui também a capacidade de decodificar em tempo real os pacotes capturados, fornece uma biblioteca ampla com os protocolos de redes de computadores e utiliza uma combinação de aplicações nativas e implementações na linguagem Java para otimizar a performance na decodificação dos pacotes capturados (SlyTechnologies, 2014).

Na captura dos pacotes, a placa de rede ethernet do computador é colocada no chamado modo *promíscuo*, no qual o computador recebe todos os pacotes de dados que trafegam pela rede, não apenas aqueles endereçados a ele.

Os pacotes capturados foram separados de acordo com seu protocolo de rede. A característica relevante para determinar um protocolo neste caso foi o número da porta de origem e a porta de destino do pacote. Usualmente para pacotes HTTP, a porta utilizada é a 80, para HTTPS 443, FTP utiliza a porta 21 para mensagens de controle, e 20 para transferência dos dados em si. Por fim, o protocolo SSH utiliza a porta 22 no envio e recebimento de pacotes. No Quadro 1 é mostrado o método responsável pela classificação e captura dos pacotes com base no número de porta de saída ou destino.

```
public void nextPacket(PcapPacket packet, String user) {

    binaryPacket = formatString(packet.toHexdump().toString());

    if (packet.hasHeader(tcp) && (packet.getHeader(tcp).source() == 80 ||
            packet.getHeader(tcp).destination() == 80) ||
            (packet.hasHeader(tcp) && (packet.getHeader(tcp).source() == 443 ||
            packet.getHeader(tcp).source() == 443))) {
        httpPackets.add(binaryPacket + "\n");

    }
    else
    if (packet.hasHeader(tcp) && ( packet.getHeader(tcp).source() == 20 ||
    packet.getHeader(tcp).destination() == 20) ||
    (packet.hasHeader(tcp) && (packet.getHeader(tcp).source() == 21 ||
    packet.getHeader(tcp).destination() == 21))||
    (packet.hasHeader(tcp) && (packet.getHeader(tcp).destination() == 989 ||
    (packet.getHeader(tcp).destination() == 55323)||
    (packet.getHeader(tcp).destination() == 55324)))) {
        ftpPackets.add(binaryPacket + "\n");

    }
    else
    if(packet.hasHeader(tcp) && (packet.getHeader(tcp).source() == 22 ||
    packet.getHeader(tcp).destination() == 22)){
        sshPackets.add(binaryPacket + "\n");
    }else{
        otherPackets.add(binaryPacket + "\n");
    }
}
```

Quadro 1 – Método para captura de pacotes de dados.
Fonte: Autoria Própria (2014).

Os pacotes capturados inicialmente vêm no formato hexadecimal, com caracteres alfanuméricos que variam de 0 a 9 e A a F. Dessa forma, foi necessário convertê-los para o formato binário, com caracteres 0 e 1, para ser compatível com os dados de entrada da rede neural. Após essa conversão, os bits foram gravados em arquivos, com extensão “.pat”, compatíveis com a aplicação utilizada no desenvolvimento das redes neurais. Nas Figuras 17 e 18 são apresentados trechos de pacotes de dados capturados nos formatos binário e hexadecimal, respectivamente.

```
0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 1 1 0 0 0 1 1 0
1 0 0 0 1 0 0 0 1 1 0 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0
0 0 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 1 0
1 0 0 1 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0 0 0 1 1 0 0 1
0 0 1 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 0 1 1 1
1 1 1 1 1 0 0 1 0 0 0 0 1 0 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1 0 0
0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
1 1 0 1 0 0 1 1 0 1 0 1 1 1 1 0 0 0 0 0 0 1 0 1 0 1 0 0 0 0 0
0 0 1 0 1 0 1 0 0 0 0 0 1 0 1 0 1 1 0 0 0 1 0 1 1 0 0 1 1 1 1
1 1 1 0 0 0 0 1 0 0 1 0 1 1 1 1 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 1
```

Figura 23 – Amostra de pacotes de dados em formato binário.
Fonte: Autoria Própria (2014).

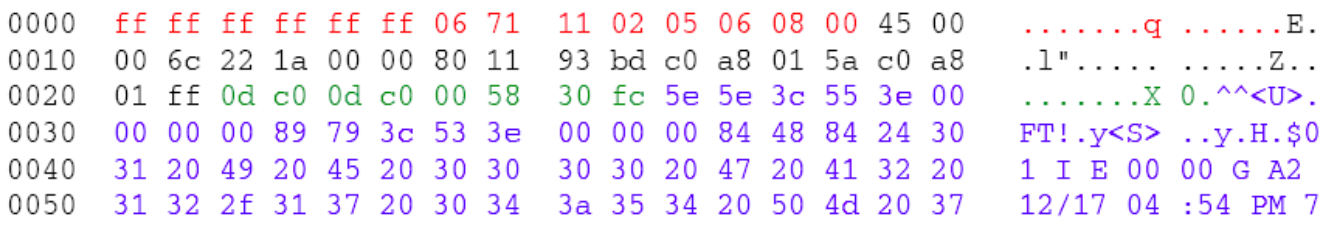

Figura 24 – Amostra de pacotes de dados em formato hexadecimal e caracteres ASCII.
Fonte: CallerID (2014).

3.2.3 Desenvolvimento das RNAs

Para a implementação das RNAs, as aplicações SNNS e JavaNNS foram utilizadas de forma conjunta. No SNNS, foram utilizados comandos para a criação das arquiteturas, e execução dos processos de aprendizagem validação e teste. O JavaNNS foi utilizado para visualizar a arquitetura montada e também para iniciar os neuronios artificiais com valores aleatórios. É necessário que os neuronios possuam um peso sináptico aleatório inicial, pois sem este não é possível executar as operações de atualização dos pesos de acordo com o aprendizado adquirido pela rede neural.

Foi verificado que, apesar do SNNS possuir funções de inicialização dos pesos sinápticos, ocorriam problemas na randomização dos valores, o que acarretava em erro na correção dos pesos. Dessa forma, foi utilizado o JavaNNS para esta etapa. O Quadro 2 contém um trecho de código fonte com as funções de criação das RNAs.

```
net_create_input_layer = "ff_bignet -p 1 1000 -p 1 "
net_create_hidden_layer = " Act_Logistic Out_Identity hidden "
net_create_output_layer = " -p 1 4 "
net_create_link = " -l 1 + 2 + -l 2 + 3 + "
net_prefix = "le_network"
net_ext = ".net"
net_number_count = 1
net_i = 400

repeat

    net_name = net_prefix+ "_" + net_number_count + net_ext
    cmd = net_create_input_layer + net_i + net_create_hidden_layer + net_create_output_layer + net_create_link + net_name

    execute(cmd)

    net_i = net_i + 400;
    net_number_count = net_number_count + 1

until(net_i > 2000)
```

Quadro 2 – Trecho de código fonte para criação das RNAs.
Fonte: Autoria Própria (2014).

Após a etapa de criação e inicialização das RNAs, são realizados os processos de treinamento, validação e teste, utilizando os arquivos de amostras de pacotes de dados, como demonstrado no Quadro 3.

```
min_validation_error = 10000
min_error_epoch = 0
max_epochs = 60

loadNet("rede.net")
loadPattern("treino.pat")
loadPattern("validacao.pat")

setSubShuffle(TRUE)
setLearnFunc("Std_Backpropagation", 0.2, 0.1)
setUpdateFunc("Topological_Order")

repeat
        setPattern("treino.pat")
        trainNet()
        train_error = SSE
        setPattern("validacao.pat")
        testNet()
        validation_error = SSE

        if (validation_error < min_validation_error) then

                min_validation_error = validation_error
                min_error_epoch = CYCLES
                saveNet("melhorRna.net")

        endif
until (train_error < 0.002 || max_epochs < CYCLES)

delPattern("treino.pat")
delPattern("validacao.pat")
```

Quadro 3 – Trecho de código com instruções de treinamento e validação das RNA. Fonte: Autoria Própria (2014).

Após a realização das etapas de treinamento e validação, a RNA que obteve o melhor resultado, o menor erro SSE, é transformada em código fonte na linguagem C e é testada com a utilização dos arquivos de amostra de dados para teste. Para essa transformação, é utilizada a ferramenta SNNS2C, onde o arquivo da RNA gerada pela aplicação SNNS, com exetnsão “.net” é utilizado como entrada de dados, e o resultado é um arquivo com extensão “.c”, com o código fonte, e mais um arquivo com extensão “.h”, com o cabeçalho contendo funções utilizadas na execução do código fonte C.

Depois da transformação, a rede neural em código C é executado tendo com entrada de dados o arquivo com as amostras para teste de reconhecimento de padrões, que tem como

resultado uma tabela contendo números variando entre 0 e 1, indicando o desempenho da rede neural no reconhecimento de cada pacote da amostra.

3.2.4 Desenvolvimento das SVMs

Para o desenvolvimento das SVMs neste trabalho, foi utilizada a aplicação LIBSVM. De acordo com Chang e Lin (2014), LIBSVM é um software integrado, de código fonte aberto, para a classificação de vetores de suporte, (C-SVC, nu-SVC), regressão (epsilon-SVR, nu-SVR) e estimativa de distribuição (uma classe SVM) e suporta classificação multi-classe.

O objetivo proposto do LIBSM é ajudar os usuários de outros domínios para usar com facilidade SVM como uma ferramenta. LIBSVM fornece uma interface simples, onde os usuários podem facilmente fazer um link com os seus próprios programas. As principais características do LIBSVM incluem:

- Implementações de SVM diferentes ;
- Eficiente na classificação multi-classe ;
- Validação cruzada para seleção de modelos ;
- Estimativas de probabilidade ;
- SVM ponderada para dados desequilibrado s;
- Código fonte nas linguagens C ++ e Java;
- Interface gráfica para demonstração da classificação da SVM e regressão .

4 RESULTADOS E DISCUSSÃO

Neste capítulo serão apresentados resultados sobre o desempenho obtido pelas RNAs e SVMs implementadas e a comparação entre o desempenho obtido por estas duas técnicas.

4.1 DESEMPENHO DAS RNAS

No seguinte cenário, cinco redes neurais artificiais foram implementadas, com o objetivo de reconhecer protocolos de redes de computadores, tendo variação no número de neurônios na sua camada oculta, sendo 400, 800, 1200, 1600 e 2000 neurônios. Todas as RNAs tiveram o mesmo número de neurônios nas camadas de entrada e saída, sendo 350 para a camada de entrada e 4 para a de saída.

Com base nos resultados apresentados na Tabela 1, observou-se que a RNA com o menor erro SSE foi a que possui 800 neurônios na camada oculta (RNA2), permitindo assim, concluir que não necessariamente a RNA com a maior quantidade de neurônios apresentará o melhor desempenho.

Tabela 1 – Desempenho das RNAs

RNA	**Número de neurônios**	**Erro SSE**
RNA 1	400	41.974201
RNA 2	800	29.05041
RNA 3	1200	1400.0000
RNA 4	1600	700.0000
RNA 5	2000	72.10769

Na Figura 11 são apresentadas as curvas de aprendizado de cada uma das RNA testadas, com suas respectivas épocas e erro SSE, o qual decresce no decorrer das épocas de treinamento.

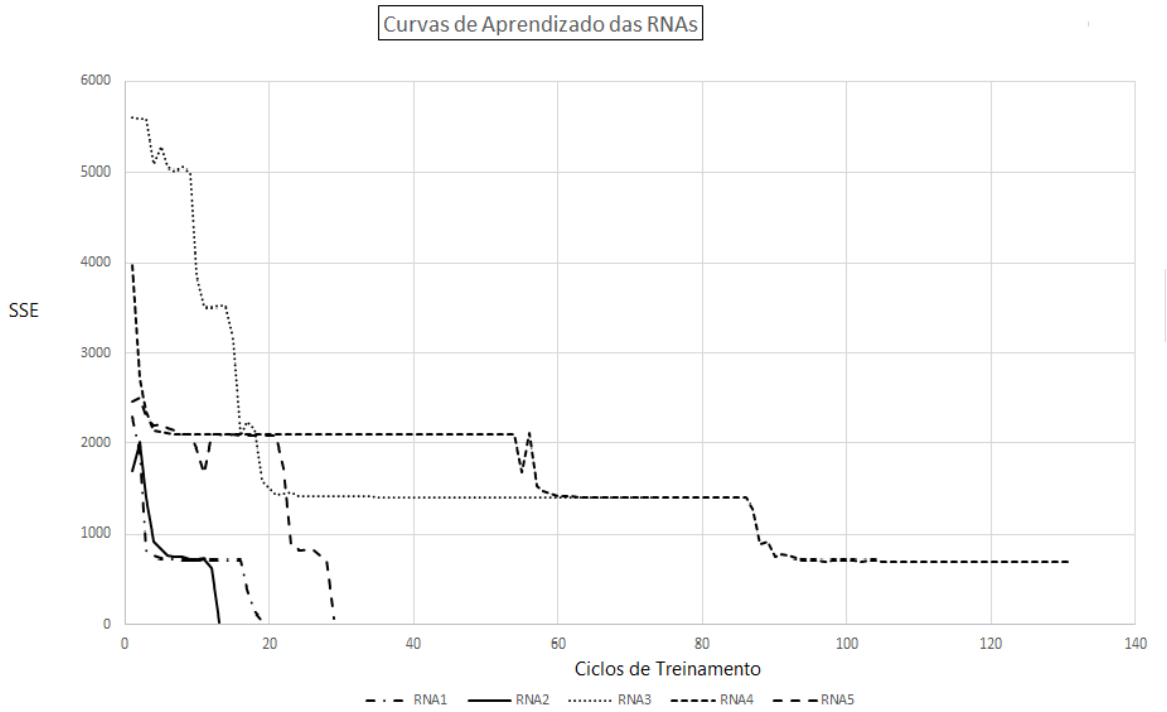

Figura 25 – Curvas de aprendizado das RNA treinadas.
Fonte: Autoria Própria (2014).

A matriz de confusão da Tabela 2 apresenta o resultado percentual de acerto das RNA no reconhecimento dos padrões, onde os valores em destaque são as taxas de acerto, e os demais valores são as taxas de erro (falsos-positivos), do protocolo associado à coluna.

Tabela 2 – Matriz de confusão da rede neural selecionada.

	HTTP	**FTP**	**SSH**	**Outros**
HTTP	29,1040131	27,10584913	35,5686723	8,221465463
FTP	63,47254661	63,47254661	35,60436407	0,922967288
SSH	0,008945716	0,00101536	94,92184434	5,068194582
Outros	0,013660312	0,055585937	0,023583702	99,90717005

Com base nos resultados da matriz de confusão, observou-se que a RNA selecionada (RNA 3) teve um razoável percentual de acerto, sendo próximo a 30% no reconhecimento de pacotes HTTP, 63,472% em pacotes FTP, 94,921% em pacotes SSH e 99,90% no reconhecimento dos demais protocolos.

Pode-se concluir que a RNA teve um desempenho não muito alto no reconhecimento de padrões em pacotes HTTP, o que possivelmente é devido à estrutura do seu cabeçalho que pode possuir características semelhantes a pacotes de outros protocolos. Nos pacotes FTP e

SSH, a RNA selecionada teve um bom desempenho no reconhecimento dos padrões, o que pode ser atribuído à estrutura destes pacotes, que se diferenciam dos demais protocolos.

Por fim, o reconhecimento dos demais pacotes, referentes aos protocolos que não foram especificados para a captura, teve quase 100% de reconhecimento devido a quantidade de demais protocolos existentes (ICMP, ARP, RARP, UDP, entre outros).

Utilizando o cálculo da acurácia, que determina em porcentagem a precisão do resultado obtido, relativo ao veradeiro resultado esperado para as amostras, pela técnica analisada. Na Equação 2.6 é representada o cálculo da acurácia:

$$Acurácia = Número\ de\ verdadeiros\ positivos\ /\ Número\ de\ amostras\ total \quad (2.6)$$

Sendo assim RNAS apresentaram o seguinte resultado de 47,5% de acurácia, o que representa um acerto na classificação correta de 142 pacotes em uma amostra de teste em um total de 300 pacotes.

4.2 DESEMPENHO DAS SVM

Ao contrário dos procedimentos para obtenção de resultados de treinamento realizados no treinamento das RNAs, tais como matriz de confusão, a ferramenta LIBSVM encapsula estes procedimentos apresentando de forma direta em gráficos os dados gerados durante o período de treinamento das SVMs. As SVMs, diferente das RNAs não é implementada em forma de topologias, mas através da configuração e teste de diferentes valores parâmetros através de um processo do tipo força-bruta, onde vários valores de parâmetros, buscando o melhor resultado.

Na Figura 26 é apresentado o gráfico com a acurácia das SVMs treinadas, com a variação dos parâmetros *gamma* e custo *C*, na utilização de uma função de *kernel* RBF.

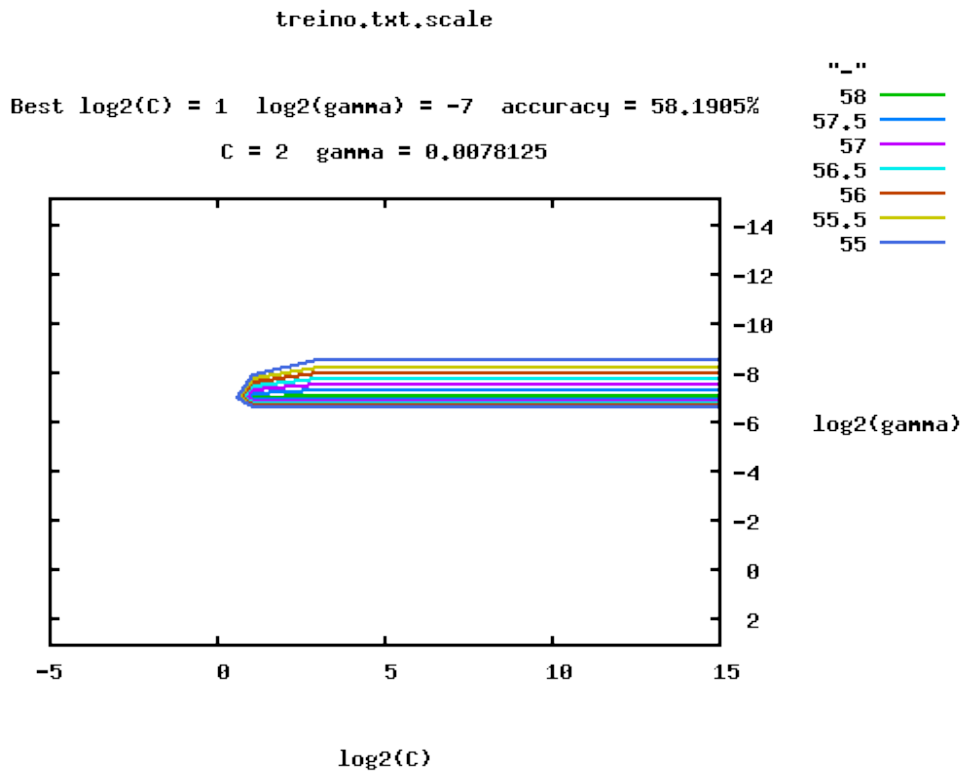

Figura 26 – Desempenho das SVMs treinadas com variação de parâmetros *gamma* e custo *C*. Fonte: Autoria Própria (2015).

De acordo com o gráfico gerado, pode-se concluir que os melhores parâmetros econtrados no processo de treinamento apresentoram um valor de *gamma* igual a 0.0078125 e um custo *C* com valor igual a 2.

Nos processo de treinamento das SVMs, o melhor valor de acurácia obtido foi de 100% em uma amostra de 300 pacotes, indicando que todas as amostras foram reconhecidas corretamente.

5 CONSIDERAÇÕES FINAIS

5.1 CONCLUSÃO

Ainda existe muito a ser explorado na aplicação de técnicas de IA no reconhecimento de dados e comportamento em redes de computadores. Estes estudos podem vir a trazer informações relevantes para diversas aplicações, tais como as voltadas à análise de tráfego de dados e segurança da informação.

Este trabalho teve a finalidade de demonstrar a capacidade de utilização de técnicas de reconhecimento de padrões, em específico as RNAs e SVMs, na análise de pacotes de dados. De acordo com os testes realizados e resultados obtidos, pode-se concluir que é viável a utilização das técnicas analisadas para este propósito

Quanto à utilização de dados na forma binária, esta foi escolhida devido à forma como os dados são recebidos pela camada física do modelo TCP/IP, que é o modelo aplicado de forma prática no tratamento dos dados que trafegam em redes de computadores. Dessa forma, com base nas análises realizadas, podem vir a ser desenvolvidos novos procedimentos e equipamentos que atuem na análise de dados diretamente na camada física.

Algumas dificuldades foram encontradas durante o desenvolvimento do trabalho, tais como a falta de aplicações para a composição dos arquivos de amostras de dados, tanto para RNAs, quando para SVMs, a falta de parâmetros para os processos de treinamento e validação das duas técnicas abordadas.

5.2 AVALIAÇÃO COMPARATIVA DOS RESULTADOS DAS SVMS E RNAS

Avaliando os resultados obtidos, pode-se notar que as SVMs obtiveram melhor resultado na acurácia, sendo este o denominador comum a ser comparado ente as duas técnicas. Porém, vale destacar que o valor de 100% na acurácia das SVMs pode indicar que houve *overfitting*, ou supertreinamento, onde ocorre da técnica, utilizada “decora” as

amostras, ficando especializada em identificar apenas estas amostras apresentadas. Os testes realizados para RNAs e SVMs indicam que ambas as soluções podem ser utilizadas de forma adequada para tarefas de classificação, regressão e previsão.

A vantagem mais relevante da utilização das SVMs é a formulação do seu processo de aprendizagem, utilizando a utilização quadrática. Isso reduz em boa parte o número de operações no modo de aprendizagem. É bem vista como solução para o tratamento de grandes conjuntos de dados, onde algoritmo SVM possui melhor desempenho e menor tempo de processamento.

5.3 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Para trabalhos futuros ou continuação deste trabalho, podem ser realizados mais estudos relacionados à construção das arquiteturas de redes neurais artificiais com a finalidade de reconhecer padrões em pacotes de dados. Também podem ser realizados estudos teóricos epráticos visando a determinação de parâmetros que otimizem os processos de treinamento, validação e teste das redes neurais.

Também podem ser realizados diversos testes de desempenho aplicando os algoritmos de SVM, buscando obter parâmetros para melhores resultados no reconhecimento de padrões. Ademais, também podem ser explorados novos estudos comparativos entre estas técnicas aboradas e muitas das técnicas de reconhecimento de padrões já existentes.

Esses estudos podem ser aplicados também no projeto e desenvolvimento de aplicações voltadas ao monitoramento e segurança da informação, como monitoramento do fluxo de dados em um determinado ambiente, controles de acesso, detecção de intrusão ou anomalias nos padrões de acesso a uma determinada rede de computadores.

REFERÊNCIAS

Almeida F. F. M. Relatório técnico: Support VectorMachine. Universidade Federal de Campina Grande, Centro de Ciência eTecnologia, 2007.

BONAVENTURE, Oliver: Computer Networking: Principles Protocols and Practice, 2011

BOURDON, Romain. Wamp Server, 2014.

CAMPOS, José; LOTUFO Ana; MINUSSI, Carlos; LOPES, Mara. Implementação de Redes Neurais Artificiais Utilizando a Linguagem de Programação Java, DINCON´10, 9 Brazilian Conference on Dynamics, Control and their Applications, Ilha Solteira, jun. 2010.

CALLERID. Ethernet Link Supplement Whozz Calling? Full-Featured Series, Lite Series and Pos Series Units, 2014

CGIbr. Comitê Gestor da Internet no Brasil: Relatório de Políticas de Internet Brasil 2011, Fundação Getúlio Vargas, 2012

CHANG, Chih-Chungh; LIN, Chih-Jen. LIBSM, A Library for Support Vector Machines, National Science Council of Taiwan, 2014.

CHURCLAND, Patricia S.; SEJNOWSKI, Terrence, I. The Computional Brain University of California at Santa Cruz, California USA, 1992.

CITRIX, About Xen Server, 2013.

DING, Chris H.; DUBCHAK, Inna. Multi-Class Protein Fold Recognition Using Support Vector Machines and Neural Networks. University of California Berkeley, CA, USA, 2001.

ECLIPSE FOUNDATION. Eclipse IDE for Java EE Developers, 2014.

ELETRONIC TECHNOLOGY DIRECTIONS TO THE YEAR 2000, 10., 1995, Adelaide. Introduction to Artificial Neural Networks, 1995.

FERREIRA, Helen P. Proposta de uma rede neural SOM para classificação de pacotes de dados que trafegam por rede de computadores. 2010. 51 f. Dissertação (Bacharelado em Ciência da Computação) – Fundação de Ensino “Eurípedes Soares da Rocha” Centro Universitário de Marília - UNIVEM, Marília, São Paulo, 2011.

FIELDING, et al. Hypertext Transfer Protocol – HTTP 1.1. Network Working Group, The Internet Society, 1999.

GARCIA, Luis M. Programming with Libpcap - Sniffing the network from our own application, Hakin9, 2008.

GONÇALVES, André R. Máquina de Vetores Suporte, Unicamp, Campinas, Brasil 2010.

HAYKIN, Simon: Neural Networks: A Comprehensive Foundation, New Jersey, USA, 1999.

HEARST, Marti A. Support Vector Machines, New Jersey, USA 1998.

HINTON, Geoffrey E. Learning Representations by Recirculation, Toronto, CA 1989.

HUSTON, Geoff. Anatomy: A Look Inside Network Address Translators, The Internet Protocol Journal - Volume 7, Number 3, 2004.

IFTIKHAR, Ahmad; ABDULLAH, Azween B.; ALGHAMDI, Abdullah S. Applicattion of a Neural Network in Detection of DOS Attacks. SIN ’09 Proceedings of the 2nd international conference on Security of Information and networks. Gazimagusa, 2009, p. 229-234, out. 2009.

KLEINROCK, Leonard. History of Communications An Early History of the Internet. Communications Magazine, IEEE, v 48. P. 26 – 36, ago. 2010.

LORENA, Ana C.; CARVALHO André C. P. L. F. Uma Introdução às Support Vector Machines, RITA, 2007

MAMIER, Günter; WIELAND, Jens. SNNS Stuttgart Neural Network Simulator, Batchman Manual, University of Stuttgart, 2014

MATHWORKS, Matlab Documentation SSE, 2014.

MCCULLOCH, Warren S.; PITTS, Walter H.; A Logical Calculus of the Ideas Immanent in Nervous Activity, Bulletin of Mathematical Biophysics, 1943.

MEYER, David. Support Vector Machines * The Interface to libsvm in package e1071. FH Technikum Wien, Austria, 2014.

MORETO, Miguel; ROLIM, Jacqueline. Análise Automática de Oscilografias em Sistemas Elétricos de Potência, Sba Controle & Automação [online], 2010.

MÜLLER, K. R.;MIKA, S.; RÄTSCH, G; TSUDA, K.; SCHÖLKOPF, B. An introduction to *kernel*-based learning algorithms. IEEE Transactions on Neural Networks, 12(2):181– 201, Março 2001.

OSOWSKI, Stanislaw; SIWEK, Krzysztof; MARKIEWICZ, Tomasz. MLP and SVM Networks – a Comparative Study. Proceedings of the 6th Nordic Signal Processing Symposium - NORSIG 2004, Espoo, Finland, 2004.

QADEER, Mohammed A.; ZAHID, Mohammad; IQBAL, Arshad; SIDDIQI, Misbahur R. Network Traffic Analysis and Intrusion Detection Using Packet Sniffer. 2nd International Conference on Communication Software and Networks, Singapore, ICCSN '10. 2010, p. 313 – 317 fev. 2010.

PARKER, David B. Optimal Algorithms for Adaptative Netoworks: Second Order Back Propagation, Second Order Direct Propagation, and Second Order Hebbian Learning, New York, USA, 1987.

REBELO, L. D. T., Avaliação Automática do Resultado Estético do Tratamento Conservador do Cancro de Mama. Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2008

REYNOLDS, Joyce, POSTEL, Jon. File Transfer Protocol FTP, Request For Comments RFC 765, Network Working Group, 1985.

SHAO, Yang; LUNETTA, Ross, S. Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points, Isprs Journal of Photogrammetry And Remote Sensing, 2012.

SHEPHERD, Gordon M.; KOCH, Christoff, Introduction to Synaptic Circuits, Oxford, 1990.

SOCOLOFSKY, Theodore J.; KALE, Claudia J. A TCP/IP Tutorial, Network Working Group, United Kingdom, jan. 1991.

SILVA, Juliana M. N. Redes Neurais Artificiais: Rede Hopfield e Redes Estocásticas. 2003 40 f. Dissertação (Pós-graduação em Ciência da Computação) – Instituto de Computação, Universidade Federal Fluminense, Niterói, 2003.

SILVA, Lília de S. Uma metodologia para detecção de ataques no tráfego de redes baseada em redes neurais. 2008. 254 f. Tese (Pós-Graduação – Doutorado em Computação Aplicada) – Instituto Nacional de Pesquisas – INPE, São José dos Campos, São Paulo, 2008.

SILVA, Renato M. Redes neurais artificiais aplicadas à detecção de intrusão em redes TCP/IP. 2005. 139 f. Dissertação (Mestrado em Engenharia Elétrica) – Departamento de Engenharia Elétrica do Centro Técnico Científica da PUC – Rio, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

SLYTECHNOLOGIES, jNetPcap 1.3 Overview, 2014.

SMOLA, A. J.; SCHÖLKOPF B. Learning with *Kernel*s. The MIT Press, Cambridge, MA, 2002.

STUART, Russel; Norvig, Peter. Atificial Intelligence A Modern Approach, Second Edition, 1995

SUNG, A. H.; MUKKAMALA, S. Identifying important features for Intrusion Detection Using Support Vector Machines and Neural Networks. Applications and the Internet, 2003. Proceedings. 2003 Symposium on, 2003.

TIC Domicílios e Empresas 2011 Pesquisa sobre o uso das tecnologias de informação e comunicação no Brasil. CGI.br Comitê Gestor da Internet no Brasil, São Paulo: 2013.

TISSOT, Hegler C.; CAMARGO, Luiz C.; POZO, Aurora T. R. Treinamento de Redes Neurais Feedforward: comparativo dos algoritmos Backpropagation e Differential Evolution. BRACIS 2012, Brazilian Conference on Intelligent Systems. Curitiba, 2012, out. 2012.

TORRES, Gabriel. Redes de Computadores Cusro Completo, 2009.

UHRIG, Robert E. Introduction to Arificial Neural Networks, Orlando FL, USA 1995

VAPNIK, Vladmir; CORTES, Corinna. Support Vector Machines, AT&T Bell Labs., Hohndel, NJ 07733, USA, 1995.

VILLANUEVA, John A. Managed File Transfer and Network Solutions Countering Packet Sniffers Using Encrypted FTP, 2014

YLONEN, Tatu; LONVICK, Chris. The Secure Shell (SSH) Authentication Protocol, Network Working Group, 2006.