UNIVESIDADE ESTADUAL PAULISTA

Júlio de Mesquita Filho

Pós-Graduação em Ciência da Computação

Clayton Reginaldo Pereira

# Detecção de Intrusão em Redes de Computadores Utilizando Floresta de Caminhos Ótimos

UNESP

2012

**Clayton Reginaldo Pereira**

Prof. Dr. João Paulo Papa (Orientador)

Dissertação de Mestrado elaborada junto ao Programa de Pós-Graduação em Ciência da Computação - Área de Concentração em Sistemas de Computação como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

**UNESP**

**2012**

Departamento de Computação
Universidade Estadual Paulista

**Clayton Reginaldo Pereira**

Setembro de 2012

# Detecção de Intrusão em Redes de Computadores Utilizando Floresta de Caminhos Ótimos

**Banca Examinadora:**

- Prof. Dr. João Paulo Papa (Orientador)
- Prof. Dr. Aparecido Nilceu Marana (DCo/FC/Unesp)
- Prof. Dr. Moacir Pereira Ponti Júnior (ICMC/USP)

Pereira, Clayton Reginaldo
Detecção de intrusão em redes de computadores utilizando Floresta de Caminhos Ótimos / Clayton Reginaldo Pereira. - São José do Rio Preto : [s.n.], 2012.
76 f. : 17 il. ; 30 cm.

Orientador: João Paulo Papa
Dissertação (mestrado) - Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Processamento de imagens 2. Visão por computador. 3. Inteligência artificial. I. Papa, João Paulo. II. Universidade Estadual Paulista, Instituto de Biociências, Letras e Ciências Exatas. III. Título.

CDU – 004.89

Ficha catalográfica elaborada pela Biblioteca do IBILCE
Campus de São José do Rio Preto - UNESP

# Dedicatória

"Quero agradecer por tudo mas não sei como, me fizeram de tudo, me deram de tudo, inclusive a vida!!!"

São pessoas mais que especiais, uma é aquela pessoa boa, alegre, companheira, divertida, tenho um amor incondicional por você, afinal Dona Silvia, és minha mãe, minha amada MÃE.

Já a outra pessoa, é aquele espelho, aquele exemplo, fiel, justo, símbolo de honestidade, pessoa para ser admirada, ainda mais por mim, que mesmo com uma idade não de menino, faço questão de atravessar a cidade toda semana apenas para estar alguns minutos ouvindo sua sapiência, que orgulho enorme em dizer a meus amigos *"ESTAVA COM MEU PAI"*.

Meu muito obrigado meu Pai e minha Mãe, mais uma conquista, nada disso seria possível sem tê-los sempre por perto, amo muito vocês.

*A meus pais Oswaldo da Silva Pereira e Silvia Domingues Pereira.*

## Um pouco de minha história, quando poderia IMAGINAR...

Quando poderia imaginar....sonhei ser jogador, ser motorista, ser um mecânico como meu Pai, mas nada disso aconteceu.

Quando poderia imaginar....ao ficar doente em 2003, que DEUS estaria me guiando e não me castigando, foi muito difícil entender o que o Senhor preparara para mim, foram noites e noites sem dormir e não deixando mais alguém dormir também, quanta parceria e cuidados comigo, tinha um anjo a meu lado, ainda bem, anjos existem, no meu caso esse anjo se chama Keila, minha esposa, muito obrigado Senhor!

Quando poderia imaginar....que morando de aluguel e com dois filhos, eu que não queria mais saber de estudo teria a grande chance de cursar uma faculdade, e o que era apenas para me distrair e esquecer da doença, passaria a ser uma meta, um desejo.... e porque não, um GRANDE SONHO!

Quando poderia imaginar....eu que sempre me julguei tão forte, estaria trancado dentro de casa com raiva do mundo, e receberia um "tal"Padre Marcelo Aparecido Paes, amigo da família conhecido como *Padre Marcelinho*, que me desafiaria em nome do Senhor e me traria de volta ao mundo, foi ali, ali foi o primeiro passo.

Quando poderia imaginar....que minha amada avó, a famosa Dona Nega, com seus 75 anos de idade na época iria ser pessoa presente e forte ao auxiliar nos cuidados de meus filhos enquanto eu fazia o que todos achavam uma loucura, um curso superior.

Quando poderia imaginar....que faria amizades fantásticas, que perderiam finais de semana para me ensinar Pascal, C, Java, pessoas que em momentos de fraqueza minha, pensamentos em desistir, conseguiriam fazer com que eu entendesse o necessário e seguisse para mais um passo, quantas horas de trabalho no meu primeiro ano de faculdade em 2005 com Leandro Bodo, Stefânia Parpinelli e André Jacon.

Quando poderia imaginar....que em 2006 brotaria em mim um desejo enorme de aprender, ao ponto de trabalhar por um período de graça, sem receber, apenas para me acrescentar mais e mais, que teria hoje essa "gana"pela pesquisa, por buscar entender como

funciona, como posso aplicar idéias minhas em determinadas coisas....quando poderia imaginar.

Quando poderia imaginar....que em 2010 seria aluno de mestrado *UNESP*, o mais velho de um grupo de pessoas maravilhosas que, coordenados pelo professor "Papa"buscam a todo momento novas maneiras de aplicarmos nosso produto de pesquisa, que em diversas vezes não está ligada diretamente a nossa área, mas aprendi também que o sabor do desafio é muito bom, porém, o prazer da missão cumprida é mais gratificante!

Hoje próximo ao passo mais importante da minha vida entendo...entendo que não é preciso parar para imaginar, que o tempo de DEUS é o que importa e simplesmente, temos que fazer o melhor a cada dia, nunca imaginei estar aqui, hoje totalmente CURADO da epilepsia, busco cada dia mais, conheci novos amigos que me apóiam muito, Rodrigo Mizobe, Luis Algusto Martins, Luis Claudio Sugi Afonso, e claro, o *incontroverso* amigo Kelton Augusto Pontara da Costa dentre outros nessa minha jornada, tenho agora novas metas e aprendi, hoje não imagino, eu planejo.....eu executo!

**Clayton Reginaldo Pereira.**

"Cada um que passa em nossa vida passa sozinho, pois cada pessoa é única, e nenhuma substitui outra. Cada um que passa em nossa vida passa sozinho, mas não vai só, nem nos deixa sós. Leva um pouco de nós mesmos, deixa um pouco de si mesmo. Há os que levam muito; mas não há os que não levam nada. Há os que deixam muito; mas não há os que não deixam nada. Esta é a maior responsabilidade de nossa vida e a prova evidente que nada é ao acaso."

(Antoine De Saint-Exupery)

# Agradecimentos

Gratidão é uma sensação tão agradável...nesta minha jornada nada fácil, muitas pessoas me ajudaram, me deram força de diferentes formas, porém, acredito que tenha além de agradecer, que pedir desculpas também.

Desculpas a minha esposa Keila que sempre fez valer sua jura feita no altar *"na alegria e na tristeza, na saúde e na doença"* sempre a meu lado, porém, em busca do meu objetivo me tornei ausente muitas vezes em momentos importantes a ela, mas sei também que vê esse momento como uma conquista "NOSSA", obrigado por estar a meu lado, Te Amo.

Desculpas também a meus filhos Lucas e Leticia por priva-los de brincadeiras com o papai em vários momentos, amo demais vocês que como sua mãe, são presentes que o Senhor me deu, mas sei também que orgulham-se de seu pai, e tudo o que faço hoje, é pensando em vocês meus amores!

Desculpas a meus irmão Everton (Ní) e Anderson (Dande) por ter me tornado uma pessoa ausente na vida de vocês, por não me comportar como aquele irmão mais velho, aquele que está sempre consciente caso precise de uma palavra amiga, mas creio que estejam com muito orgulho do que esse irmão mais velho, sistemático e chato alcançou.

Desculpas claro, a meu grande professor de graduação, orientador dessa dissertação, amigo nas horas alegres e de dificuldades, João Paulo Papa, pessoa que aprendi a gostar e admirar muito pelo homem que é, pelo companheirismo, pela grande paciência e claro, por ter acreditado em mim, és não só um orientador, mas sim...um amigo, um grande irmão!

Sou grato a todos que passaram em minha vida durante esse meu percurso, não estaria aqui sem qualquer um de vocês, pessoas que serei grato pelo resto de minha vida, não posso claro deixar e agradecer ao Senhor meu DEUS, pois sem ele, nada disso estaria acontecendo, obrigado meus amores, meus amigos e meu "DEUS".

Quero agradecer também o imenso incentivo e apoio recebido pela FAPESP durante meu período de mestrado, por meio da bolsa #2010/02045-3.

# Abstract

Intrusion Detection Systems have used even more techniques based on artificial intelligence aiming to increase their accuracy. Techniques such as neural networks and suport vetor machines are examples of approaches which have been widely used for this purpose. However, the complexity for learning new attacks is very high, imparing their retraining in real time. Since a new classifier called Optimum-Path Forest - OPF has been recently proposed, and has shown similar recognition rates to those state-of-the-art pattern recognition techniques, but much faster for training, this work aims to evaluate OPF in the context of intrusion detection in computer networks. Experiments on different databases have shown that OPF may be an appropriate tool to detect instrusion in computer networks, since it can learn new attacks faster than some of the other techniques employed in this work.

# Resumo

Sistemas de detecção de intrusão tem utilizado cada vez mais técnicas baseadas em inteligência artificial com o objetivo de aumentar sua precisão. Técnicas como redes neurais artificiais e máquinas de vetores de suporte são exemplos de abordagens que tem sido amplamente empregadas para este fim. Entretanto, a complexidade dessas técnicas para o aprendizado de novos ataques é muito alta, inviabilizando o seu retreinamento em tempo real. Dado que um novo classificador de padrões denominado Floresta de Caminhos Ótimos (*Optimum-Path Forest* - OPF) foi recentemente proposto, e obteve taxas de reconhecimento similares àquelas obtidas por técnicas estado-da-arte em reconhecimento de padrões, porém mais rápido para o treinamento, este trabalho objetiva a avaliação do classificador OPF no contexto de detecção de intrusão em redes de computadores. Experimentos em diferentes bases de dados mostraram que o classificador OPF pode ser uma ferramenta adequada para detectar intrusões em redes de computadores, dado que o mesmo pode aprender novos ataques mais rapidamente que algumas das técnicas empregadas no trabalho.

# Sumário

# 1 Introdução

O número de pessoas interessadas em obter acesso não autorizado a informações presentes em redes de computadores vem crescendo exponencialmente nos últimos anos juntamente com o diferente número de ataques [Kompella e Varghese 2004, Levchenko 2004]. Abordagens de contenção de invasões mais conhecidas e amplamente utilizadas, tais como antivírus e *firewalls*, por exemplo, são de extrema importância para evitar ataques em uma rede de computadores. Entretanto, devido a grande variedade de códigos maliciosos que surgem diariamente, essas ferramentas tem a sua sensibilidade afetada, pois tratam-se de ferramentas que possuem a necessidade de sua configuração manual, o que poderia aumentar consideravelmente as chances de um erro nesse processo por falta de conhecimento. Assim, sua constante atualização torna-se mais necessária para minimizar o risco de invasões, muito embora não resolva de fato, o problema. Um erro de configuração em um *firewall*, por exemplo, pode diminuir a sua eficácia, ou até mesmo inviabilizar o seu uso.

Devido a diversificação dos ataques, bem como suas complexidades, empresas têm aumentado seus investimentos em estudos para a elaboração de sistemas de detecção de intrusões mais eficientes, utilizando técnicas de inteligência artificial. Tais sistemas possuem como principal objetivo monitorar redes de computadores com o intuito de identificar e aprender novos tipos de ataques, automatizando também o processo de monitoramento evitando, assim, que o administrador da rede fique periodicamente analisando o seu tráfego [Whitman e Mattord 2004].

Ghosh et al. [Ghosh, Wanken e Charron 1998] e Cannady [Cannady 1998] utilizaram, por exemplo, Redes Neurais Artificiais com Perceptron Multi-camadas (*Artificial Neural*

*Networks with Multilayer Perceptron* - ANN-MLP) [Haykin 1998] para a detecção de anomalias em redes de computadores. Zanero e Savaresi [Zanero e Savaresi 2004], Kayacik et al. [Kayacik, Heywood e Heywood 2007] e Lei e Ghorbani [Lei e Ghorbani 2004] empregaram Mapas Auto-Organizáveis (*Self Organizing Maps - SOM*) [Haykin 1998] para o monitoramento e detecção de intrusões. Trabalhos recentes [Chen, Hsu e Shen 2005, Li e Gu 2009, Haijun et al. 2007] utilizaram Máquinas de Vetores de Suporte (*Support Vector Machines* - SVMs) [Cortes e Vapnik 1995] para a detecção de anomalias em tráfego de pacotes em redes de computadores. Jucá et al. [Jucá, Boukerche e Sobral 2002], propuseram a utilização de uma abordagem especialista baseada no sistema imunológico humano, e Oke [Oke, Loukas e Gelenbe 2007] et al., utilizaram o classificador Bayesiano (*Bayesian Classifiers*) visando detectar ataques do tipo negação de serviço.

Todavia, muitas dessas tradicionais técnicas de reconhecimento de padrões podem não ser adequadas para o tratamento de um grande volume de dados em tempo real. Verificando o arquivo gerado por um sistema de monitoramento de tráfego de dados em uma rede de pequeno e médio porte, por exemplo, pode-se constatar a grande quantidade de informações geradas em um curto intervalo de tempo. Redes neurais possuem uma etapa de treinamento muito custosa, sendo também uma árdua tarefa definir a sua arquitetura.

Máquinas de Vetores de Suporte, embora amplamente utilizadas na literatura para endereçar problemas de reconhecimento de padrões, também possuem uma fase de aprendizado muito onerosa, nas quais os parâmetros para a construção do hiperplano separador ótimo no espaço induzido de alta dimensão são encontrados. Outra constatação pertinente é o fato que SVMs são classificadores binários, ou seja, foram projetados originalmente para resolver problemas de reconhecimento de padrões que envolvem apenas duas classes. Embora técnicas tenham sido propostas para contornar essa limitação, tais como executar várias SVMs para cada combinação binária de classes, isso faz com que a eficiência do classificador seja comprometida [Tang e Mazzoni 2006].

A quantidade de características extraídas para compor cada amostra do conjunto de dados é um outro problema frequentemente encontrado, pois várias informações tais como endereço IP de origem e destino, portas de origem e destino e permissões de acesso, dentre outras, aumentam a dimensionalidade do espaço de características a ser tratado

pelo classificador de padrões. Ainda que Shyu et al. [Shyu et al. 2003] tenham proposto utilizar a técnica de Análise de Componentes Principais (*Principal Component Analysis - PCA*) para reduzir o número de características em sistemas de detecção de intrusões em redes de computadores, o problema do intenso volume de dados ainda prepondera sobre a questão da dimensionalidade dos mesmos.

Assim, a utilização de muitas das tradicionais técnicas de reconhecimento de padrões ficam comprometidas, pois o alto custo de treinamento dos dados inviabiliza tais abordagens para a composição de sistemas de detecção de intrusão mais eficientes. Suponha uma situação na qual o número de falsos negativos, ou seja, o número de ataques que foram classificados como acessos permitidos, aumente consideravelmente. É desejável então que o administrador faça um retreinamento do sistema de modo a aumentar a sua eficácia. Entretanto, como o volume dos dados é muito grande, esse novo aprendizado necessita ser realizado com eficiência, muitas vezes em tempo real, de modo que esse novo ataque seja prontamente identificado.

Com o intuito de aliar eficácia e eficiência na tarefa de reconhecimento de padrões em grande volume de dados, Papa et al. [Papa, Falcão e Suzuki 2009] propuseram um novo classificador de padrões denominado Floresta de Caminhos Ótimos (*Optimum-Path Forest* - OPF), o qual reduz o problema de reconhecimento de padrões ao particionamento de um grafo em árvores de caminhos ótimos enraizadas por amostras protótipos. Nessa abordagem, qualquer elemento pertencente a uma dada árvore é mais fortemente conexo à sua raiz do que a qualquer outra. Essa força de conexidade é estabelecida por uma função de custo de caminho, previamente escolhida. Atualmente, duas variantes foram propostas: OPF com grafo completo [Papa, Falcão e Suzuki 2009] e OPF com grafo $k$-vizinhos mais próximos ($k$-*Nearest Neighbors* – $k$-NN) [Papa e Falcão 2008], sendo a primeira mais utilizada e difundida.

O presente trabalho propõe o desenvolvimento de um sistema de detecção de intrusão em redes de computadores baseado no classificador OPF com grafo completo. Experimentos com dois tipos de Redes neurais e Máquinas de Vetores de Suporte foram realizados com o intuito de comparar tais técnicas com o referido classificador baseado em floresta de caminhos ótimos. Vale ressaltar que este trabalho é o primeiro a empregar OPF nesse

contexto de pesquisa. O restante do trabalho é organizado como segue.

A Seção 2 apresenta dados e técnicas aplicadas em Sistemas de Detecção de Intrusos, já a Seção 3 apresenta os classificadores de padrões utilizados neste trabalho, as Seções 4 e 5 apresentam, respectivamente o classificador OPF e a metodologia utilizada nesse trabalho. A Seção 6 mostra os experimentos realizados e, finalmente, a Seção 7 discorre sobre os resultados e trabalhos futuros.

# 2 Sistemas de Detecção de Intrusos

Detectar uma intrusão significa identificar qualquer comportamento suspeito no tráfego de uma rede de computadores. Sistemas de detecção de intrusos (*Intrusion Detection Systems* - IDSs) monitoram esse tráfego detectando comportamentos e pacotes suspeitos que possam danificar ou ter acesso não autorizado a informações sigilosas que trafegam pela rede. O CERT.br (Centro de Estudos, Respostas e Tratamento de Incidentes de Segurança no Brasil)[1] juntamente com o CGI.br (Comitê Gestor de Internet no Brasil)[2], são orgãos responsáveis por tratar incidentes de segurança que envolvam redes conectadas à internet brasileira. A Figura 2.1 apresenta um levantamento recente sobre os incidentes de segurança no Brasil.

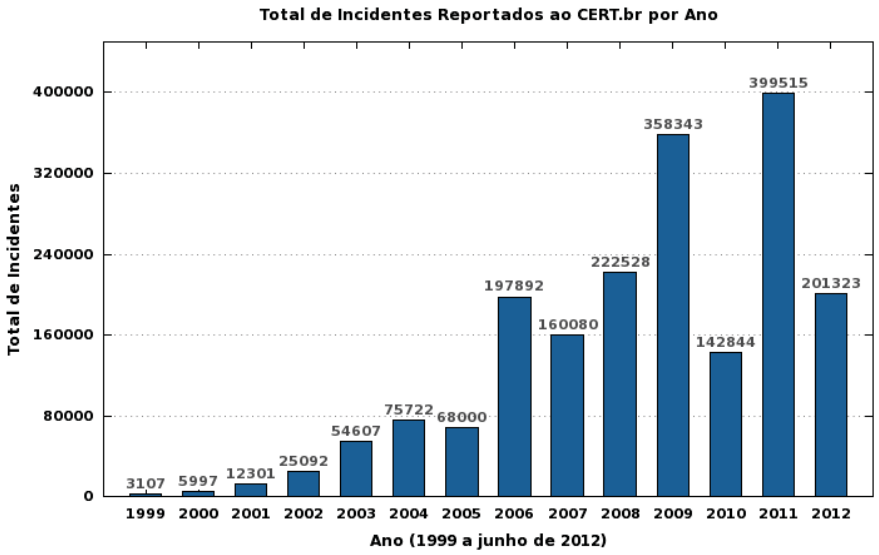



Figura 2.1: Estatística de incidentes reportados ao CERT. Fonte: (CERT.br)

[1] http://www.cert.br/stats/incidentes/
[2] http://www.cgi.br/

Sistemas de Detecção de Intrusos são aplicados como ferramentas complementares no processo de gestão de segurança em redes de computadores, pois apesar dos esforços empregados para automatizar a tarefa de detecção e respectivas respostas, ainda é indispensável a participação humana para tomar as devidas providências no caso de alertas e relatórios que são gerados pelos *IDSs*.

A utilização das técnicas para detecção de intrusão, pressupõe que o comportamento e as atividades de um usuário ou programa legítimo, são totalmente diferente das atividades de um invasor. O papel principal de um *IDS* é detectar tentativas de intrusão e auxiliar o administrador da rede no monitoramento do ambiente, mantendo-o informado para que possa tomar as devidas providências. Vale ressaltar, que é papel fundamental do *IDS* não somente a detecção de tentativas de intrusão externa, mas também do acesso de usuários internos da rede que não possuem autorização a informações privilegiadas. Faz desejável, então, que um *IDS* possua as seguintes características:

- **Confidencialidade:** é o processo que previne o acesso não-autorizado a recursos e informações sigilosas contidas em determinados pontos da rede, na qual o acesso é restrito a determinados usuários;
- **Integridade:** previne a alteração de recursos e informações por usuários não-autorizados;
- **Disponibilidade:** garante a acessibilidade de recursos e informações no momento desejado por usuários autorizados.

Devido as suas características, os *IDSs* tornaram-se uma ferramenta de grande importância, sendo essencial para auxiliar no processo de administração da segurança de organizações que possuem uma certa quantidade de máquinas em sua rede.

É possível descrever um processo de detecção de intrusão como um conjunto de técnicas e métodos que são utilizados para detectar atividades suspeitas em redes de computadores [Patel, Qassim e Wills 2010]. Tais técnicas fazem uso de informações coletadas do sistema por meio do monitoramento dos computadores, rede ou segmento de rede para

detectar intrusões [Kizza 2009]. Considerado como um sistema projetado para a monitoração de ambientes computacionais, os *IDSs* examinam todo o tráfego da rede buscando encontrar ameaças direcionadas ao sistema, podendo estas, serem em forma de ataques a rede, uma determinada forma de *scan*, ou até mesmo formas de espionagem utilizando técnicas diversas como, por exemplo, força bruta. Tais sistemas buscam informações de diversas fontes e pontos da rede, analisam estas informações no intuito de encontrar alguma anomalia ou sinal de tentativa de intrusão [Zeltser et al. 2002].

As técnicas utilizadas nos sistemas de detecção de intrusos podem ser classificadas em duas categorias principais: técnicas de detecção de intrusão baseadas em assinaturas e técnicas de detecção de intrusão baseadas em anomalias ou comportamento. As próximas seções tratam de explicar ambas.

## 2.1 Detecção de Intrusões Baseados em Assinaturas

Esse processo é definido através de uma base de dados contendo assinaturas de ataques pré-estabelecidos ou conhecidos pelo *IDS*. Quando o ataque ou a tentativa de intrusão ocorre, é efetuada uma comparação da assinatura da intrusão com a contida na base de dados [Neelam e Saurabh 2012]. Caso as mesmas se correspondam, é gerado então um alarme informando o ocorrido ao responsável pela rede.

Pode-se dizer que os *IDSs* baseados em assinaturas são muito precisos em suas detecções, apresentando baixo número de falsos positivos, ou seja, quando um sensor rotula uma instrução benigna de um determinado tipo de pacote como sendo uma tentativa de ataque. A principal exigência para essa técnica é manter a base de dados dos ataques atualizada. Porém, muitas vezes, mesmo com a base atualizada, essa técnica encontrará dificuldades em detectar ataques desconhecidos, mutantes ou camuflados, ou seja, que não possuam uma assinatura arquivada em seu banco de dados pois, constantemente, novas formas de ataques ou mesmo, variações de ataques já conhecidos são criados e esse sistema não é sensível ao ponto de ser capaz de detectar ataques desconhecidos, [Zeltser et al. 2002].

## 2.2 Detecção de Intrusões Baseados em Anomalias

Anomalias aplicam a técnica de detecção de intrusões no comportamento do sistema, atuando em várias áreas tais como nucleo do sistema, *logs* de eventos, informações de pacotes de rede, informações sobre o software em execução, e etc. Tais informações do sistema operacional e qualquer comportamento anormal ou atividade suspeita que possam ocorrer na rede durante o tráfego dos pacotes, são encaminhadas ao administrador acusando uma possível intrusão. Porém, algumas vezes pode ocorrer de o *IDS* acusar atividades anômalas que não sejam intrusivas gerando um falso-positivo [Zheng, Hou e Wang 2011].

É tido como base para esse tipo de sistema de detecção de intrusão que certas atividades de intrusão ou ataque fazem parte de um conjunto composto por atividades anômalas, tendo como uma boa vantagem o fato da capacidade em detectar novos tipos de ataques que sejam diferentes do comportamento normal do tráfego da rede. Entretanto, uma de suas limitações se dá pelas ocorrência de alarmes falsos, dado que nem toda atividade que não seja de utilização cotidiana representa um ataque.

# 3 Técnicas de Classificação de Padrões

Neste capítulo são exploradas as técnicas de classificação de padrões empregada neste trabalho. O capítulo inicia abordando as Redes Neurais Artificiais, uma técnica que apresenta um modelo matemático baseado na estrutura neural de organismos inteligentes. Em seguida, são discutidas as Máquinas de Vetores de Suporte, uma técnica que trata a separabilidade das classes com um modelo estatístico. Logo após, é apresentada a técnica denominada k-vizinhos mais próximos. Finalmente, será destacada a técnica Floresta de Caminhos Ótimos.

## 3.1 Redes Neurais Artificiais

Redes Neurais Artificiais (*Artificial Neural Networks* - ANN) são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência [Haykin 1994]. Uma ANN é caracterizada pelo padrão de conexão (modelo conexionista) entre os neurônios (topologia), pelo método de determinação dos pesos e conexões (treinamento ou aprendizagem) e pela função de ativação responsável pela saída da rede. Esses processos comportam-se de maneira similar aos grupos de neurônios do cérebro humano que recebem e transmitem informações através dos dendritos e axônios, respectivamente [Kovacs 1996].

Quando é apresentado à rede um conjunto de entradas e suas respectivas saídas, as quais estão representando o comportamento de um processo específico, a mesma, através do treinamento, é capaz de se auto-ajustar com o objetivo de mapear o relacionamento funcional entre as entradas e saídas. Por conseguinte, após a execução desse algoritmo de treinamento, a rede deve ser capaz de generalizar o comportamento do processo quando

outras entradas, diferentes daquelas utilizadas no treinamento, são apresentadas. Tal característica é particularmente útil quando o relacionamento entre as entradas e saídas do processo analisado é não-linear, ou então, quando o relacionamento não é claramente definido de forma que a sua modelagem, por técnicas de classificação convencionais, não é intuitiva.

As redes neurais representam o desenvolvimento de sistemas computacionais capazes de reconhecer e classificar padrões, resolver problemas complexos, realizar processos indutivos e dedutivos, entre outros. Em geral, a utilização de redes neurais, comparando-se com outras técnicas, possui as seguintes vantagens [Raghu, Poongodi e Yegnanarayana 1995]:

- os relacionamentos funcionais entre os padrões de entrada e saída podem ser capturados por uma rede neural, sendo que tais relacionamentos não precisam ser conhecidos ou descritos explicitamente;
- nenhuma hipótese precisa ser feita sobre as distribuições estatísticas dos padrões de entrada; e
- as redes neurais são tolerantes à falhas no sentido que, mesmo quando alguns elementos ou conexões do processamento apresentam-se degradados, a performance da rede pode ser pouco afetada.

As arquiteturas de redes neurais artificiais utilizadas para modelar os sistemas neurais biológicos podem ser divididas em três categorias distintas com o intúito de propagação da informação. A primeira categoria de redes neurais, "*feedforward*" transforma conjuntos de sinais de entrada em conjuntos de sinais de saída, sendo que os parâmetros dessa transformação são ajustados de forma supervisionada, de acordo com os resultados desejados. Na segunda categoria de redes neurais, "*feedback*", as informações de entrada definem o estado inicial de atividade do sistema de realimentação. Depois de algumas transições de estado, atinge-se um estado que é considerado o resultado final da computação. Na terceira categoria, as células vizinhas da rede neural interagem mutuamente, a fim de transformarem-se adaptativamente em detectores especializados de diferentes padrões. Nessa última categoria, o aprendizado da rede neural é realizado de forma não supervisionada, através de mapas de auto-organização (*Self-Organizing Maps - SOM*) [Kohonen 1990].

### 3.1.1 Mapas Auto-Organizáveis

No modelo de redes neurais baseado em mapas de auto-organização proposto por Kohonen [Kohonen 1990, Kohonen 1989, Kohonen 1993], a segregação espacial de diferentes respostas e suas organizações em sub-conjuntos topologicamente relacionados resultam em um alto grau de eficiência nas operações típicas de uma rede neural.

O modelo SOM implementa uma projeção não-linear de um espaço multidimensional $X$ em um espaço bidimensional $M$, denominado mapa de auto-organização. Essa projeção, de modo análogo à maioria das projeções encontradas no cérebro, implementam mapeamentos topológicos, ou seja, elementos vizinhos em $X$ são geralmente mapeados em elementos vizinhos em $M$. O mapeamento topológico constitui uma importante característica do modelo de redes neurais proposto por Kohonen, pois permite encontrar características e outras abstrações do espaço multidimensional através de identificação de agrupamentos (*clusters*) no mapa bidimensional.

O mapeamento do espaço multidimensional para o bidimensional é realizado por uma função $f : X \rightarrow M$, onde $X \subset R^n$ e $M \subset R^2$, que associa cada elemento $x \in X$ a um par $(i, j) \in M$. Os elementos $m_{ij}$ do mapa $M$, assim como os dados de entrada, são vetores de tamanho $n$, que mantém os pesos das ligações sinápticas da rede neural. Apresentado um estímulo específico $x \in X$ à rede neural, o "nó vencedor"do mapa $M$ para esse estímulo é determinado verificando-se qual dos elementos do mapa possui a menor distância ao estímulo, ou seja, o nó vencedor tem as coordenadas $(i, j) \in M$, tal que:

$$||m_{ij} - x|| = min\{||m_{kl} - x||, \forall i, j, k, l = 0, \ldots, D\} \tag{3.1}$$

onde $D$ é o tamanho de $M$.

Portanto, um estímulo $x \in X$ apresentado à camada de entrada da rede neural é mapeado no elemento $(i, j)$ da camada de saída da rede neural, ou seja do mapa de auto-organização $M$, para o qual a distância entre esse estímulo e o vetor dos pesos das ligações sinápticas $m_{ij}$ da rede neural seja mínima, comparando-se com os demais vetores de pesos das ligações sinápticas $m_{kl}$. Nesse modelo de rede neural, os neurônios da camada de saída

interagem lateralmente com seus vizinhos. Essas interações se dão na fase de aprendizado, quando os pesos da célula "vencedora" e dos seus vizinhos são ajustados, de tal modo que a célula vencedora tenha pesos idênticos aos valores do estímulo sendo apresentado à rede num determinado instante do treinamento, e as células vizinhas tenham pesos semelhantes aos da célula vencedora, sendo que o grau de semelhança é ponderado de acordo com a distância entre a célula vizinha e a célula vencedora.

A interação da célula vencedora $(i, j)$ com uma célula vizinha $(k, l)$, num determinado instante $t$ do treinamento, é definida por uma função $h_{kl}(t)$ da distância entre elas. Normalmente, a função $h_{kl}(t)$ é uma Gaussiana.

O aprendizado nesse modelo é dito não supervisionado porque o mapeamento do espaço multidimensional para o bidimensional é feito a despeito dos resultados esperados na saída do processamento, ao contrário dos outros modelos, onde os valores esperados e os valores obtidos pela rede, para um dado estímulo de entrada, influenciam no reajuste dos pesos das ligações sinápticas.

No modelo de redes neurais baseadas em mapas de auto-organização, os pesos das ligações sinápticas são ajustadas da seguinte forma:

$$m_{kl}(t+1) = m_{kl}(t) + (x - m_{kl}(t))h_{kl}(t) \tag{3.2}$$

onde $h_{kl}(t) = \alpha(t)exp\left[-\frac{||r_{kl}-r_{ij}||^2}{\delta(t)^2}\right]$, $(i, j)$ são as coordenadas do nó vencedor para o estímulo $x$, $||r_{kl} - r_{ij}||$ é a distância espacial entre o nó vencedor e o nó $(k, l)$, e $\delta(t)$ é uma função do tempo de treinamento $t$, que determina a variância da Gaussiana $h_{kl}$.

O tamanho da vizinhança $N_{ij}(t)$ decresce de acordo com o número de iterações $T$ da fase de treinamento. No início do treinamento, isto é, quando $t = 0$, $N_{ij}(t)$ inclui toda a rede, e no final, quando $t = T$, $N_{ij}(t)$ contém somente o nó vencedor $(i, j)$. A seguinte função linear pode ser adotada para determinar os ajustes nas extensões da vizinhança: $N_{ij}(t) = D\left[1 - \frac{t}{T}\right]$, para $t = 0, \ldots, T$. De forma análoga, as funções $\delta(t)$ e $\alpha(t)$ podem ser reajustadas em cada iteração através das funções lineares $\delta(t) = 2 - \frac{t}{T}$ e $\alpha(t) = 1 - \frac{t}{T}$, para $t = 1, \ldots, T$, respectivamente.

Após a fase de treinamento, os nós da rede são rotulados de acordo com a classe de estímulos para os quais eles apresentam respostas mais intensas. Se essa fase for supervisionada, então deve-se apresentar para a rede conjuntos de estímulos para os quais se conhece a classificação. Para cada classe de estímulos conhecidos verifica-se quais são os nós vencedores no mapa de auto-organização. Esses nós são rotulados com o rótulo associado àquela classe de estímulos. Se essa fase não for supervisionada, é necessário fazer uma análise de agrupamentos no mapa de auto-organização. Um modelo da arquitetura bidimensional típica da rede de Kohonen é apresentado na Figura 3.1.

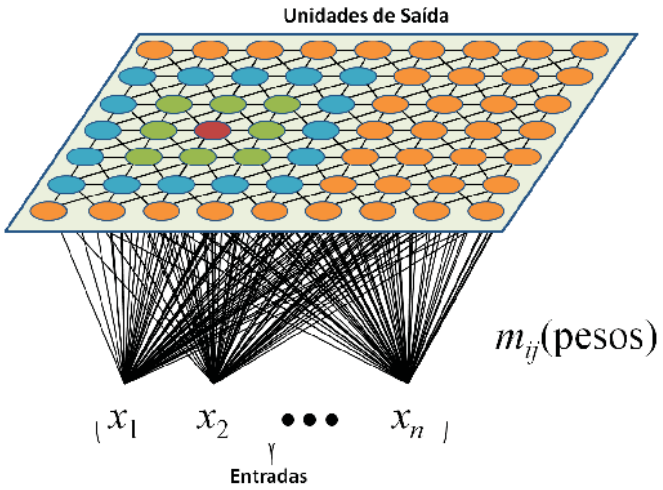



Figura 3.1: Arquitetura típica de uma rede de Kohonen adaptada de [Haykin 1999].

### 3.1.2 Redes Neurais Perceptron Multicamadas

#### 3.1.2.1 Perceptron

Durante as décadas de 50 e 60, uma grande revolução ocorreu no campo de pesquisa relacionado à teoria de reconhecimento de padrões com o aparecimento das chamadas máquinas que aprendem, os perceptrons [Haykin 1994]. Em sua metodologia mais básica, o perceptron aprende uma função de decisão linear que dicotomiza dois conjuntos de treinamento, linearmente separáveis em um número finito de passos iterativos. A resposta desse dispositivo básico baseia-se na soma ponderada de sua entrada, ou seja:

$$d(x) = \phi(\chi); \tag{3.3}$$

$$\chi = \sum_{i=1}^{n} w_i x_i + w_{n+1} \tag{3.4}$$

que é uma função de decisão linear em relação aos elementos do vetor de características. Os coeficientes $w_j$, $j = 1 \ldots n+1$, são os chamados pesos das conexões, sendo análogos às sinapses no sistema neural humano. Tem-se ainda que $\phi$ é a chamada função de ativação e $\chi$ é o somatório da rede [Haykin 1994] possíveis.

Dentre as várias escolhas para $\phi$, tem-se:

- Função limiar (threshold):

$$\phi(\chi) = \begin{cases} 1 & \text{se } \chi \geq 1, \\ 0 & \text{caso contrário} \end{cases} \tag{3.5}$$

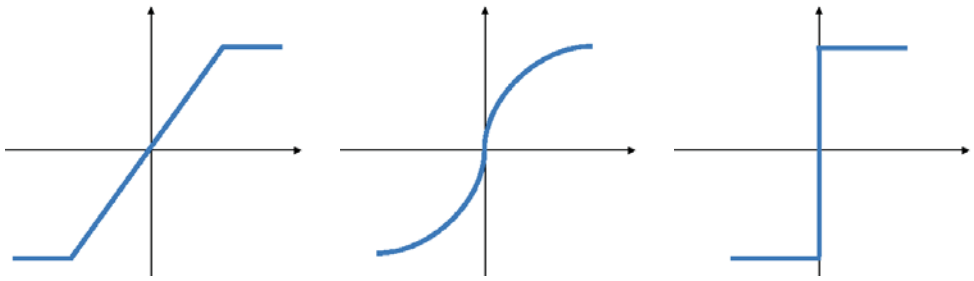
- Função sigmoidal:

$$\phi(\chi) = \frac{1}{1 + \exp(-\chi)} \tag{3.6}$$

- Função identidade:

$$\phi(\chi) = \chi \tag{3.7}$$

A Figura 3.2 ilustra essas funções.



(a) (b) (c)

Figura 3.2: Funções de Ativação Linear (a), Sigmoidal (b) e Identidade (c).

O algoritmo de treinamento do perceptron utiliza uma função de ativação por limia-

rização muito semelhante à Equação (3.5), descrita por

$$\phi(\chi) = \begin{cases} 1 & \text{se } \chi \geq 1, \\ -1 & \text{caso contrário.} \end{cases} \tag{3.8}$$

Geometricamente, a equação

$$\chi = \sum_{i=1}^{n} w_i x_i + w_{n+1} = 0 \tag{3.9}$$

ou

$$\chi = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} = 0, \tag{3.10}$$

define um hiperplano no espaço $n$-dimensional de padrões. Os $n$ primeiros coeficientes estabelecem a orientação do hiperplano, enquanto o último coeficiente, $w_{n+1}$, também chamado de *bias*, é proporcional à distância perpendicular à origem do hiperplano. Caso as duas classes apresentadas ao algoritmo de perceptron sejam linearmente separáveis em $\Re^n$, o mesmo converge em um número finito de passos. Entretanto, caso ambas classes não obedeçam a tal critério, o algoritmo executará indefinidamente, não convergindo para a solução ótima. O modelo simplificado do perceptron é apresentado na Figura 3.3.

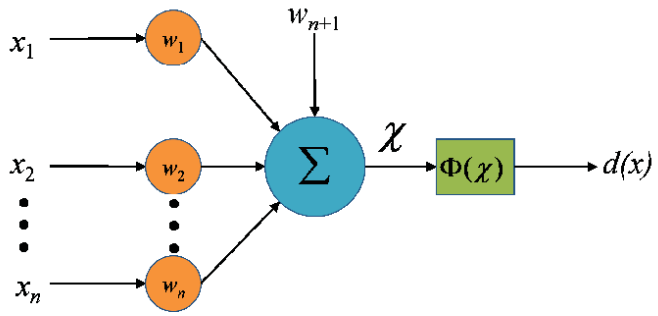



Figura 3.3: Modelo simplificado do perceptron adaptado de [Haykin 1999].

### 3.1.2.2 Perceptron Multicamadas

Para o tratamento de problemas que possuem várias classes, independentemente de as mesmas serem separáveis ou não, o algoritmo de perceptron visto anteriormente é ineficiente, visto que o mesmo funciona adequadamente quando a tarefa de classificação

consiste em duas classes linearmente separáveis. Conectando vários perceptrons, pode-se projetar uma estrutura chamada Perceptron Multicamadas, também conhecida como MLP [Haykin 1994], a qual consiste de várias camadas de elementos computacionais idênticos (neurônios) dispostos de tal maneira que a saída de cada um deles alimente a entrada de cada neurônio da camada seguinte. O número de neurônios da primeira camada, denominada $A$, corresponde a $N_A$; sendo, frequentemente denotado $N_A = n$, ou seja, o número de elementos da primeira camada é igual à dimensionalidade do vetor de características. O número de neurônios da camada de saída, chamada $Q$, corresponde a $N_Q = M$, sendo $M$ igual ao número de classes do problema de classificação. A rede neural reconhece um vetor de padrões $\mathbf{x}$ como pertencente à classe $\omega_m$ caso a $m$-ésima saída da rede possuir o maior valor dentre as $m - 1$ saídas restantes.

No caso da arquitetura citada anteriormente, a entrada de cada elemento em qualquer camada corresponde à soma ponderada das saídas da camada anterior. Denotando $K$ a camada anterior a $J$, tem-se que a entrada de cada neurônio na camada $J$, denotada por $I_j$ é dada por

$$I_j = \sum_{k=1}^{N_K} w_{jk} O_k \tag{3.11}$$

sendo que

$$O_k = \phi(I_k), \tag{3.12}$$

onde $j = 1, 2, \ldots, N_J$, sendo $N_J$ e $N_K$ o número de elementos da camada $J$ e $K$, respectivamente, e $w_{jk}$ são os pesos que modificam as saídas $O_k$ dos elementos da camada $K$.

Algumas convenções geralmente adotadas para tais estruturas de redes neurais são:

- a função de ativação utilizada na camada de entrada é a função identidade, dada pela Equação (3.7);
- camadas não adjacentes não são conectadas diretamente e, por conseguinte,
- todos os neurônios das camadas ocultas possuem a mesma função de ativação $\phi$.

Sendo assim, este modelo de rede neural é chamado de *feedforward*, devido ao fato

de tanto a camada de entrada quanto as intermediárias, ou escondidas, serem submetidas somente à camada mais alta, ou seja, a camada $K$ é subalterna à camada $K-1$, assim por diante. Uma representação de um modelo básico do modelo *feedfoward* pode se visto na Figura 3.4.

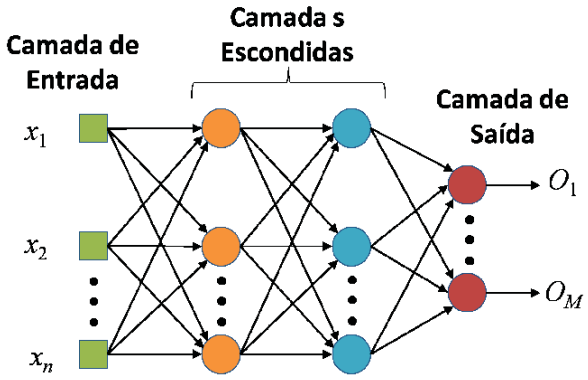



Figura 3.4: Representação básica de uma rede *feedforward* (multicamadas) adaptada de [Haykin 1999].

#### 3.1.2.3 Treinamento por Retropropagação

O principal objetivo deste algoritmo é desenvolver uma regra de treinamento com o intuito de minimizar o erro quadrático total entre as saídas desejadas $r_q$ e as saídas reais $\phi_q$ dos nós em uma camada de saída $Q$, ou seja, minimizar a equação abaixo:

$$E_Q = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - \phi_q)^2, \tag{3.13}$$

onde $N_Q$ é o número de nós da camada de saída $Q$.

Durante o treinamento com o algoritmo de retropropagação, primeiramente, um padrão é apresentado à camada de entrada da rede. A saída obtida pela última camada é comparada com a saída desejada. Caso a mesma não estiver correta, o erro é calculado através da Equação (3.13) e propagado a partir da camada de saída até a camada de entrada, sendo que os pesos das conexões são modificados através da regra delta generalizada conforme o erro é retropropagado.

Desta forma, a regra delta generalizada $\Delta w_{qp}$ permite o ajuste dos pesos em cada uma

das camadas de maneira a tentar minimizar a função de erro mostrada acima, ou seja, calcular o gradiente negativo de $E_Q$, dado pela equação:

$$\Delta w_{qp} = -\alpha \frac{\partial E_Q}{w_{qp}}, \tag{3.14}$$

em que $P$ precede a camada $Q$ e $\alpha$ é um incremento positivo de correção.

Pode-se resumir e generalizar o procedimento de treinamento da seguinte maneira: para quaisquer duas camadas $K$ e $J$, em que $K$ precede imediatamente $J$, a Equação (3.14), através de algumas manipulações algébricas, pode ser re-escrita como:

$$\Delta w_{jk} = \alpha \delta_j \phi_k. \tag{3.15}$$

Caso $J$ seja uma camada de saída, $\delta_j$ é dado por:

$$\delta_j = (r_j - \phi_j)\phi'_j, \tag{3.16}$$

onde $\phi'_j$ denota a derivada de $\phi_j$.

Se $J$ for uma camada interna e $P$ for a próxima camada, então $\delta_j$ é dado por:

$$\delta_j = O'_j \sum_{p=1}^{N_P} \delta_p w_{jp}, \tag{3.17}$$

onde $j = 1, 2, \ldots, N_J$.

Desta forma, todo o processo de treinamento começa com um conjunto arbitrário de pesos da rede. Em seguida, a aplicação da regra delta generalizada em qualquer passo iterativo envolve duas etapas. Na primeira fase, um vetor de treinamento é apresentado à rede e propagado através das camadas da rede para o cálculo de $\phi_j$ para cada nó. As saídas $\phi_q$ dos nós da camada de saída são então comparadas com as respostas desejadas $r_q$ para que os termos de erro $\delta_q$ sejam gerados. A segunda fase envolve uma passagem para trás na rede, retropropagação, durante a qual o sinal de erro apropriado é passado por cada nó e as mudanças correspondentes nos pesos são realizadas. Uma vez que o sistema tenha sido treinado, o mesmo passa a classificar os padrões utilizando os parâmetros estabelecidos durante a fase de treinamento descrita acima.

## 3.2 Máquinas de Vetores de Suporte

As Máquinas de Vetores de Suporte, do inglês "*Support Vector Machines*"(SVM), são normalmente consideradas a primeira aplicação prática da teoria do aprendizado estatístico [Vapnik 1999, Schölkopf e Smola 2002]. Nos últimos anos, seu escopo tem crescido significativamente tanto em termos de novos algoritmos quanto de um entendimento teórico mais aprofundado. Parte destes novos algoritmos se deve aos chamados métodos de nuclearização, uma proposta para solução de problemas de *aprendizado de máquina* cuja arquitetura tem se demonstrado capaz de lidar com questões relativas às bases desta teoria. Além disso, aplicações bem sucedidas de SVM demonstram que esta técnica não só possui uma fundamentação mais sólida do que as Redes Neurais Artificiais como também são capazes de substituí-las com desempenho melhor ou semelhante em determinadas situações [Schölkopf e Smola 2002].

As SVM propõem resolver o problema de classificação de padrões assumindo ser possível separar instâncias de classes diferentes em um espaço de mais alta dimensão. Suponha uma situação na qual os dados não são linearmente separáveis. Tais amostras podem ser separadas em grupos usando curvas ou círculos como superfícies de decisão, porém encontrar tais limiares é uma tarefa custosa. A principal idéia de uma SVM é pré-processar os dados de tal forma que o problema de encontrar uma função discriminante não-linear seja transformado em um problema de encontrar um hiperplano. Assim, mapeia os dados que estão em uma dimensão qualquer para outra maior, tornando os mesmos linearmente separáveis. Isto é feito definindo um mapeamento que transforma o vetor de entrada em outro (usualmente maior) vetor. Espera-se que, escolhendo um mapeamento adequado, o novo conjunto de treinamento seja linearmente separável.

### 3.2.1 Classificadores por Hiperplano Linear

De acordo com as questões destacadas para o controle da efetividade dos algoritmos de aprendizado, se faz necessário que a capacidade da classe de funções possa ser calculada. Nos primórdios de seu estudo, Vapnik [Vapnik 1999] considerou uma classe de hiperplanos em um espaço $\mathcal{H}$, ou seja, um conjunto de pontos $\mathbf{x}$ que satisfaz

$$\langle \mathbf{w}, \mathbf{x} \rangle - b = 0, \tag{3.18}$$

onde $\langle \cdot \rangle$ e $\mathbf{w} \in \mathcal{H}$ denotam, respectivamente, o produto interno e o vetor normal ao hiperplano, e $b \in \mathbb{R}$. Temos que $\mathbf{w}$ e $b$ correspondem com funções de decisão do tipo

$$f(x) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b). \tag{3.19}$$

Baseado em dois argumentos, foi então proposto o algoritmo de aprendizado denominado Retrato Generalizado (do inglês "*Generalized Portrait*") para problemas separáveis por hiperplanos:

1. Dentre todos os hiperplanos que separam os dados, existe apenas um hiperplano ótimo distinguido pela margem de máxima separação entre qualquer ponto de treinamento e este hiperplano. Esta é a solução de

$$\max_{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}} \quad \min\{||\mathbf{x} - \mathbf{x}_i|| | \mathbf{x} \in \mathcal{H}, \langle \mathbf{w}, \mathbf{x} \rangle - b = 0\} \tag{3.20}$$

   onde $i = 1, ..., m$.

2. A capacidade da classe de hiperplanos de separação decresce com o crescimento da margem. Se os dados de treinamento são linearmente separáveis, é possível selecionar dois hiperplanos de forma que não haja pontos entre eles e, posteriormente, tentar maximizar a distância entre eles.

Para construir tal hiperplano ótimo, é necessário resolver

$$\underset{\mathbf{w} \in \mathcal{H}, b \in \mathbb{R}}{\text{minimize}} \quad \tau(\mathbf{w}) = \frac{2}{||\mathbf{w}||}, \tag{3.21}$$

sujeito à

$$(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq 1 \quad \text{para todo} \quad i = 1, ..., m \tag{3.22}$$

ou

$$(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \leq 1 \quad \text{para todo} \quad i = 1, ..., m. \tag{3.23}$$

Temos que as Equações 3.22 e 3.23 podem ser escritas como:

$$y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) \geq 1 \quad \text{para todo} \quad i = 1, ..., m, \tag{3.24}$$

onde $y_i \in \{+1, -1\}$ denota o rótulo da amostra $\mathbf{x}_i$.

A razão por detrás da minimização de $\mathbf{w}$ (Equação (3.21)) pode ser interpretada da seguinte maneira: se $||\mathbf{w}||$ fosse 1, então o termo da esquerda na Equação (3.22) seria igual à distância de $x_i$ ao hiperplano. Em geral, é preciso dividir $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b)$ por $||\mathbf{w}||$ para transformá-lo nesta distância. Sendo assim, se a Equação (3.22) for satisfeita para todos $i = 1, ..., m$ com um $\mathbf{w}$ de tamanho mínimo, a margem será maximizada como um todo. Um resumo destes argumentos é dado na Figura 3.5.

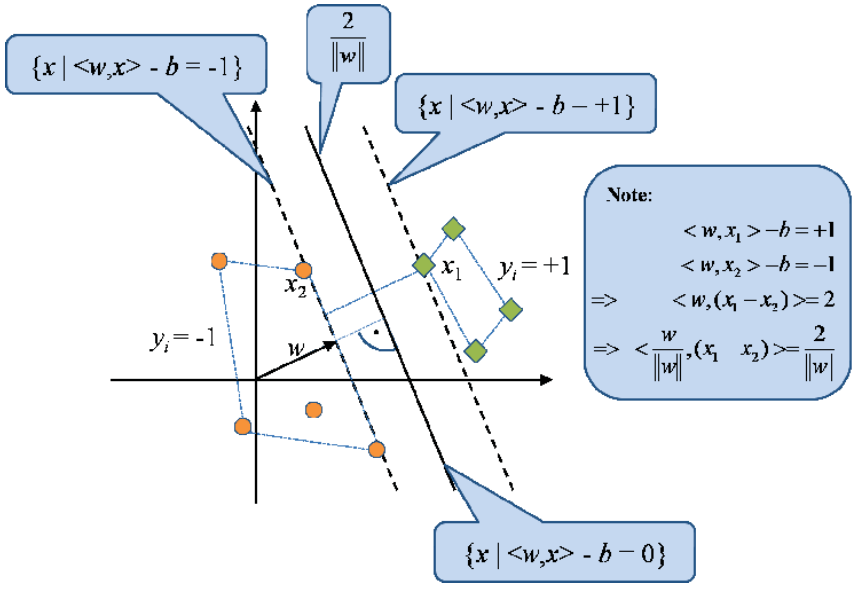



Figura 3.5: Um problema de classificação binária extraído de [Schölkopf e Smola 2002] cujo propósito é separar os círculos dos losangos.

O hiperplano ótimo (Equação (3.21)) é mostrado com uma linha sólida. No problema sendo separado há apenas um vetor ponderado $\mathbf{w}$ e um limiar $b$ tal que $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle - b) > 0$ $(i = 1, ..., m)$. Ao reescalar $\mathbf{w}$ e $b$ com o objetivo de que o(s) ponto(s) mais próximos ao hiperplano satisfaçam $|\langle \mathbf{w}, \mathbf{x}_i \rangle - b| = 1$ se obtém a forma *canônica* $(\mathbf{w}, b)$ do hiperplano.

A função $\tau$ na Equação (3.21) é chamada de *função objetivo*, enquanto a Equação (3.24) representa as *restrições de desigualdade*. Juntas elas dão origem ao que se conhece como *problema de otimização restrita*, que por sua vez é resolvido através da introdução

do *Lagrangiano* abaixo:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}||\mathbf{w}||^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1) \tag{3.25}$$

onde $\alpha_i \geq 0$ são os *multiplicadores de Lagrange* e $L$ tem que ser minimizada com relação às *variáveis primais* $\mathbf{w}$ e $b$ e maximizada com relação às *variáveis duais* $\alpha_i$. Em outras palavras, um ponto de sela deve ser encontrado. Como se pode notar, as restrições do problema original foram incorporadas no segundo termo no Lagrangiano.

Olhando melhor para o que o problema de otimização restrita nos mostra, é possível observar que se a restrição da Equação (3.22) no segundo termo do Lagrangiano for violada com $\sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{x}_i, \mathbf{w} \rangle + b) - 1 < 0$, $L$ pode ser incrementado através do decrescimento do $\alpha_i$ correspondente. Ao mesmo tempo, $\mathbf{w}$ e $b$ terão que mudar (reduzindo a margem) de tal forma que $L$ decresça. Dado que o problema é separável, num dado momento as restrições serão finalmente satisfeitas. De forma análoga, podemos entender que para todas as restrições não atendidas precisamente como igualdades, seu $\alpha_i$ correspondente deve ser 0: este é o valor de $\alpha_i$ que maximiza $L$. Desta última observação extrai-se as chamadas condições de complementaridade de Karush-Kuhn-Tucker (KKT) presente na teoria da otimização.

Considerando que num ponto de sela as derivadas de $L$ com relação às variáveis primais devem se aproximar de zero,

$$\frac{\partial}{\partial b} L(\mathbf{w}, b, \alpha) = 0 \qquad \text{e} \qquad \frac{\partial}{\partial \mathbf{w}} L(\mathbf{w}, b, \alpha) = 0 \tag{3.26}$$

nos leva à

$$\sum_{i=1}^{m} \alpha_i y_i = 0 \tag{3.27}$$

e

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i. \tag{3.28}$$

Consequentemente, o vetor-solução nada mais é que uma expansão (Equação (3.28)) de um subconjunto dos padrões de treinamento, especificamente aqueles com $\alpha_i > 0$, que

são os chamados vetores de suporte (*Support Vectors* - SVs).

Ao substituir as Equações (3.27) e (3.28) no Lagrangiano (Equação (3.25)), elimina-se as variáveis primais $\mathbf{w}$ e $\mathbf{b}$ resultando no chamado *problema de otimização dual*, que é o tipo de problema que as SVM normalmente resolvem na prática:

$$\underset{\alpha \in \mathcal{R}^m}{\text{maximize}} \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j, \rangle \tag{3.29}$$

sujeito à

$$\alpha_i \geq 0 \quad \text{para todo} \quad i = 1, ..., m$$

e

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

Ainda baseado na Equação (3.28), a função-hiperplano de decisão (Equação (3.19)) pode ser escrita como:

$$f(x) = sgn \left( \sum_{i=1}^{m} y_i \alpha_i \langle \mathbf{x}, \mathbf{x}_i \rangle - b. \right) \tag{3.30}$$

Entretanto, ainda nos falta estabelecer como $b$ pode ser calculado. De acordo com as condições KKT, todos os pontos com

$$\alpha_i [y_i (\langle \mathbf{x}_i, \mathbf{w} \rangle - b) - 1] = 0 \quad \text{para todo} \quad i = 1, ..., m, \tag{3.31}$$

são SVs situados na margem e, sendo assim, $\alpha_i > 0$. Em tais casos, é possível demonstrar que

$$\langle \mathbf{x}_i, \mathbf{w} \rangle - b = y_i. \tag{3.32}$$

Dessa forma, o limiar pode ser obtido, por exemplo, por:

$$b = y_i - \langle \mathbf{x}_i, \mathbf{w} \rangle. \tag{3.33}$$

Alternativamente, $b$ também pode ser calculado a partir dos valores das variáveis duais $\alpha_i$ e $\alpha_j$ correspondentes.

### 3.2.2 Nuclearização e Hiperplanos de Margem Suave

Esta seção está relacionada ao mapeamento das entradas para um espaço de maior dimensionalidade e ao acréscimo de artifícios de tolerância para que as SVM possam lidar com problemas não-separáveis.

Para expressar as fórmulas a partir dos padrões de entrada $\mathcal{X}$, emprega-se o produto interno dos vetores $\mathbf{xex}'$ em termos do núcleo $k$ estimado pelos elementos de entrada $xex'$:

$$k(x, x') = \langle \mathbf{x}, \mathbf{x}' \rangle. \tag{3.34}$$

Esta substituição, também conhecida como truque do núcleo (do inglês "*kernel trick*"), é utilizada para estender o conceito de classificação por hiperplanos para máquinas de vetores de suporte não-lineares. O truque do núcleo pode ser aplicado desde que todos os vetores de características ocorram apenas em produtos internos, o que pode ser observado nas Equações (3.29) e (3.30). O vetor ponderado (Equação (3.28)) torna-se então uma expansão no espaço de características e, sendo assim, não mais corresponderá ao seu respectivo vetor no espaço de entradas original. A partir da Equação (3.30), a função de decisão se tornará:

$$\begin{aligned} f(x) &= sgn\left(\sum_{i=1}^{m} y_i \alpha_i \langle \Phi(x), \Phi(x_i) \rangle - b\right) \\ &= sgn\left(\sum_{i=1}^{m} y_i \alpha_i k(x, x_i) - b\right), \end{aligned} \tag{3.35}$$

e o problema de otimização (Equação (3.29)) assumirá a forma

$$\underset{\alpha \in \mathcal{R}^m}{\text{maximize}} \quad W(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j k(x_i, x_j), \tag{3.36}$$

sujeito à $\alpha_i \geq 0$ para todo $i = 1, ..., m$, e $\sum_{i=1}^{m} \alpha_i y_i = 0$. A Figura 3.6 captura a ideia de mapear os dados do espaço de características original para outro de maior dimensão.

Mesmo com as vantagens da nuclearização do problema, na prática, o hiperplano de

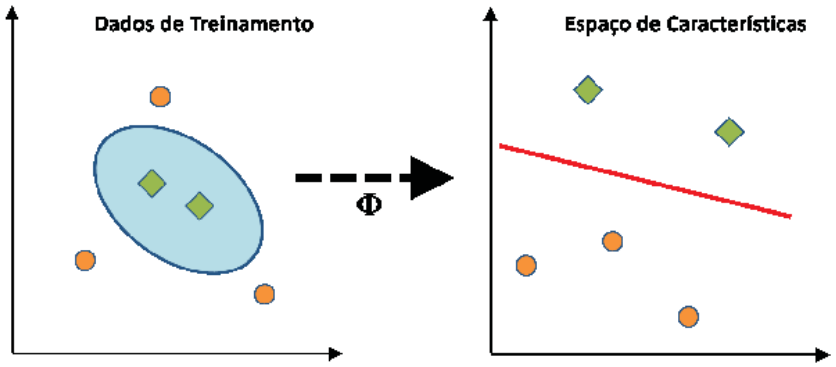



Figura 3.6: A ideia de se aplicar o truque do núcleo nas SVM: mapear os dados de treinamento em um espaço de características de maior dimensão através de $\Phi$ e construir lá um hiperplano de máxima margem de separação. Apesar da fronteira de decisão ser linear no espaço de características, após o mapeamento no espaço de entradas original ela ganha formas não-lineares. Adaptado de [Haykin 1999].

separação ainda assim pode não existir. Isto pode acontecer, por exemplo, em um conjunto de dados com muito ruído onde haja sobreposição das classes. Para permitir que os exemplos que violam a Equação (3.22) possam ser considerados, as variáveis de afrouxamento $\xi_i$ são introduzidas [Schölkopf, Williamson e Bartlett 2000, Schölkopf e Smola 2002], onde:

$$\xi_i \geq 0 \quad \text{para todo} \quad i = 1, ..., m, \tag{3.37}$$

fazendo com que as restrições assumam a forma

$$y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle - b) \geq 1 - \xi_i \quad \text{para todo} \quad i = 1, ..., m. \tag{3.38}$$

Desta maneira, um classificador de boa generalização pode ser obtido pelo controle tanto da capacidade (através de $||\mathbf{w}||$) quando da soma das variáveis de afrouxamento $\sum_i \xi_i$. É possível demonstrar que este somatório provê um limite superior no número de erros de treinamento [Schölkopf e Smola 2002].

Neste contexto, uma possível elaboração de tal classificador de margem suave é obtida pela minimização da função-objetivo

$$\tau(\mathbf{w}, \xi) = \frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m}\xi_i \tag{3.39}$$

sujeita às restrições da Equações (3.37) e (3.38), onde a constante $C > 0$ determina o equilíbrio entre a maximização da margem e a minimização dos erros de treinamento. Quanto maior for $C$, menor será a margem, menor o número de erros de treinamento e menor também a capacidade de generalização da máquina de aprendizado.

Incorporando o núcleo e reescrevendo o problema em termos dos multiplicadores de Lagrange, novamente nos leva à maximização da Equação (3.36), sujeita às restrições $0 \leq \alpha_i \leq C$ para todo $i = 1, ..., m$ e $\sum_{i=1}^{m} \alpha_i y_i = 0$. A única diferença deste caso para o caso separável está no limite superior $C$ dos multiplicadores de Lagrange $\alpha_i$. Sendo assim, a influência dos padrões individuais (que podem ser valores discrepantes) se torna limitada. A solução na Equação (3.35) não muda e o limiar $b$ pode ser calculado pelo fato de que para todos os SVs $x_i$ com $\alpha_i < C$, a variável de afrouxamento $\xi = 0$ e, assim sendo:

$$\sum_{j=1}^{m} \alpha_j y_j k(x_i, x_j) - b = y_i. \tag{3.40}$$

Analisando geometricamente, escolher $b$ significa deslocar o hiperplano, ao passo que a Equação (3.40) determina ser necessário situá-lo de tal forma que todo SVs com variável de afrouxamento iguais à zero estejam nas linhas $\pm 1$ (conforme Figura 3.5).

A função de núcleo apresentada como produto interno na Equação (3.34) é normalmente chamada de função de Núcleo Linear e, dadas algumas restrições [Schölkopf e Smola 2002], pode ser substituída por outras funções. Outra função de mapeamento bastante conhecida é a gaussiana chamada Função de Base Radial (do inglês "*Radial Basis Function*" - RBF),

$$k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right) \tag{3.41}$$

onde $\sigma > 0$ é um parâmetro da função de núcleo. Em função da variável de ajuste $C$, este tipo de SVM é comumente chamado de C-SVC e representa a classificação por SVM em

sua forma original [Cortes e Vapnik 1995].

### 3.2.3 Considerações Adicionais

Como foi possível observar ao longo desta seção, as SVM são fundamentalmente máquinas de classificação binária. No entanto, a maioria dos problemas reais a serem otimizados requerem a classificação em mais de duas classes. Uma questão que emerge desta constatação é sobre como lidar com a propriedade multiclasses inerente a muitos problemas em um classificador binário por definição. Três estratégias são comumente adotadas para emprego de SVM em problemas de $k$ classes:

**Uma Contra o Resto:** Na abordagem uma-contra-o-resto, $k$ SVM são treinadas, onde cada uma delas separa uma determinada classe de todas as demais.

**Classificação aos Pares:** Na abordagem de classificação aos pares, $k(k-1)/2$ máquinas são treinadas. Cada uma destas SVM separa um par de classes. Os classificadores são então arranjadas como nós de uma árvore, onde cada nó representa uma SVM. Tanto a busca de cima para baixo quanto de baixo para cima podem ser adotadas, sendo esta última análoga ao processo de eliminação de equipes em um campeonato esportivo.

**Espaço de diferenças:** Para transformar problemas multiclasses em binários, uma novo espaço de representações chamado espaço de diferenças foi proposto em [Phillips 1999]. Através da modelagem de dissimilaridades, um problema com $k$ classes é transformado em um problema com os conjuntos diferenças intraclasse e diferenças interclasses. Quanto mais parecida do conjunto intraclasse for a diferença entre um exemplo conhecido e um desconhecido, mais provável os dois serem da mesma classe. Por outro lado, quanto mais parecida do conjunto interclasse for a diferença entre um exemplo conhecido e um desconhecido, mais provável deles serem de classes distintas.

Em relação às duas primeiras abordagens, a uma-contra-o-resto é preferível visto que apenas $k$ SVM são treinadas em comparação a $k(k-1)/2$ da abordagem aos pares. A

complexidade da classificação é semelhante a abordagem uma-contra-o-resto *(requer a estimativa de k SVM, enquanto que a classificação aos pares requer $k-1$)*. Considerando o fato de que o número de SVM está atrelado ao número de classes a serem reconhecidas, ambas as abordagens não lidariam com aplicações de classificação em que novas classes são aceitas uma vez que o classificador já esteja treinado e em operação. Este é o cenário típico em um sistema de reconhecimento facial automático onde novos indivíduos são cadastrados a todo momento. Neste caso, a abordagem do espaço de diferenças é preferível. Apesar da necessidade do cálculo das diferenças, apenas uma predição de uma SVM é necessária, independentemente do número de classes no espaço de representação original.

Finalizando esta seção sobre SVM, não se pode deixar de mencionar características relativas ao tamanho do conjunto de treinamento em relação ao tempo demandado para o aprendizado. Esta referência ao tamanho se desdobra em duas informações importantes de serem consideradas. A primeira delas diz respeito ao número de exemplos a serem aprendidos. Ao nuclearizar um conjunto de treinamento com $n$ exemplos, $n \times n$ operações serão necessárias para o mapeamento dos espaços (conforme seção 3.2.2). Ou seja, à medida que o número de exemplos de treinamento cresce, o custo computacional cresce exponencialmente - apesar do aprendizado ser mais robusto. Também relativa ao custo do treinamento, a segunda informação diz respeito ao número de atributos (tamanho) dos exemplos no espaço original. Dependendo deste tamanho e de quão intrinseco é o problema, o uso de funções de núcleo na otimização não se justifica, sendo muitas vezes melhores os resultados obtidos a partir dos dados representados em seu espaço original.

Ainda considerando a questão da nuclearização, algumas funções de núcleo apresentam parâmetros que requerem ajuste. Isso significa que as SVM devem ser treinadas tantas vezes quanto forem necessárias para se encontrar a combinação de parâmetros ótima. Se o otimização já demandar por alto esforço computacional em função do tamanho do conjunto de treinamento, ela será ainda mais dispendiosa ao ter que buscar por parâmetros ótimos.

Com relação ao tempo de classificação, o que o influencia diretamente é o número de SVs resultantes da otimização - que expressam quão intrincado eram os dados de treinamento - e a dimensão destes SVs em seu espaço original. Apesar desta influência, o

desempenho da classificação não costuma representar um empecilho em aplicações práticas.

Uma biblioteca de código aberto bastante conhecida que implementa SVM nas mais variadas formas é a chamada LibSVM [Chang e Lin 2011]. Trata-se de um repositório de código aberto cujo objetivo é ajudar as pessoas a usarem facilmente SVM. Além de prover 4 diferentes tipos de função de núcleo, ela oferece SVM não só para classificação, mas também para ”clusterização“ e regressão. Através de uma interface por linhas de comando, é possível manipular os conjuntos de treinamento e teste, treinar SVM de diferentes maneiras, buscar por parâmetros ótimos das funções de núcleo e executar as classificações.

# 4 Classificador de Padrões Baseado em Floresta de Caminhos Ótimos - OPF

Esta seção tem por objetivo apresentar o classificador baseado em floresta de caminhos ótimos com aprendizado supervisionado. Tal classificador modela o problema de reconhecimento de padrões como um problema de floresta de caminhos ótimos em um grafo definido no espaço de atributos, onde os nós são as amostras, as quais são representadas pelos seus respectivos vetores de atributos, e os arcos são definidos de acordo com uma relação de adjacência pré-estabelecida.

Nesta versão, os arcos são ponderados, e diversas funções de custo podem ser empregadas com o intuito de particionar o grafo em árvores de caminhos ótimos, as quais são enraizadas pelos seus respectivos protótipos (sementes) na fase de treinamento. O rótulo de uma amostra a ser classificada é o mesmo do protótipo mais fortemente conexo a ela.

## 4.1 Classificação supervisionada

O algoritmo OPF com grafo completo foi primeiramente apresentado por Papa et al. [Papa, Falcão e Suzuki 2009, Papa et al. 2012] e tem sido amplamente utilizado em diversas aplicações. A técnica utilizada neste trabalho modela as amostras como sendo os nós de um grafo completo, onde os elementos mais representativos de cada classe do conjunto de treinamento, isto é, os protótipos, são escolhidos como sendo os elementos pertencentes às regiões de fronteira entre as classes.

Os protótipos participam de um processo de competição disputando as outras amostras oferecendo-lhes caminhos de menor custo e seus respectivos rótulos. Ao final deste processo, obtemos um conjunto de treinamento particionado em árvores de caminhos óti-

mos, sendo que a união das mesmas nos remete a uma floresta de caminhos ótimos. Esta abordagem apresenta vários benefícios com relação a outros métodos de classificação de padrões supervisionados: (i) é livre de parâmetros, (ii) possui tratamento nativo de problemas multiclasses e (iii) não faz alusão sobre forma e/ou separabilidade das classes. As próximas seções irão discutir a fundamentação teórica e os algoritmos de treinamento e classificação do algoritmo baseado em OPF utilizando grafo completo.

### 4.1.1 Fundamentação Teórica do Classificador OPF

Seja $Z = Z_1 \cup Z_2 \cup Z_3$ um conjunto de dados, tais que $Z_1$, $Z_2$ e $Z_3$ denotam os conjuntos de treinamento, avaliação e teste, respectivamente. Seja $(Z_1, A)$ um grafo completo cujos nós são amostras em $Z_1$ e qualquer par de amostras define um arco em $A = Z_1 \times Z_1$. Os arcos não precisam ser armazenados, sendo a representação do grafo de maneira implícita. Definimos também o caminho como uma sequência de amostras distintas $\pi_t = \langle s_1, s_2, \ldots, s_{k-1}, s_k \rangle$ com término $t$, onde $(s_i, s_{i+1}) \in A$ para $1 \leq i \leq k-1$.

O caminho é dito *trivial* se $\pi_t = \langle t \rangle$. Nós associamos a cada caminho $\pi_t$ um custo $f(\pi_t)$ dado pela função de conectividade $f$. Um caminho $\pi_t$ é considerado ótimo se $f(\pi_t) \leq f(\tau_t)$ para qualquer outro caminho $\tau_t$. Denotamos, também, $\pi_s \cdot \langle s, t \rangle$ a concatenação de um caminho $\pi_s$ e arco $(s, t)$.

Dado com um conjunto de amostras *protótipos* $S \in Z_1$, a fase de treinamento do OPF consiste em calcular uma floresta de caminhos ótimos sobre o conjunto de treinamento. Essa floresta é, essencialmente, a coleção de árvores de caminhos ótimos com raízes em cada protótipo. A partição (árvore) da grafo completo da floresta de caminhos ótimos é computado nos $\Re^n$ pelo algoritmo da transformada imagem-floresta (IFT) [Falcão, Stolfi e Lotufo 2004].

Considere, agora, $R(s) \in S$ como a raiz de $s \in Z_1$. Dizemos que $s$ é mais fortemente conectada com $R(s)$ do que qualquer outra raiz $u \in S$ em $Z_1$. Isto significa que, uma vez $s$ é conquistada por alguma amostra, a qual pode ser $R(s)$ ou alguma outra amostra $v \in Z_1$ tal que $R(v) = R(s)$, $s$ pertence à mesma árvore de caminhos ótimos de $R(s)$.

Como dito anteriomente, as amostras em $S$ são escolhidas como sendo as mais próxi-

mas com rótulos diferentes em $Z_1$. Para encontrar tais protótipos, Papa et al. [Papa, Falcão e Suzul
propuseram computar uma Árvore de Espalhamento Mínima (*Minimum Spanning Tree* - MST) em $Z_1$ e marcar as amostras conexos com rótulos diferentes como sendo os protótipos. O algoritmo 1 implementa a fase de treinamento do OPF.

**Algoritmo 1** – Treinamento por Floresta de Caminhos Ótimos

| | |
|---|---|
| Entrada: | Um conjunto de treinamento $Z_1$ $\lambda$-rotulado e o par $(v, d)$ composto pelo vetor de características e as distâncias computadas. |
| Saída: | Floresta de Caminhos Ótimos $P_1$, mapa de custo $C_1$, mapa de rótulos $L_1$, e o conjunto ordenado $Z_1'$. |
| Auxiliares: | Fila de prioridade $Q$, conjunto $S$ de protótipos, e variável de custo *cst*. |

1. *$Z_1' \leftarrow \emptyset$ e compute via MST o conjunto de prototipos $S \subset Z_1$.*
2. *Para cada $s \in Z_1 \backslash S$, faça $C_1(s) \leftarrow +\infty$.*
3. *Para cada $s \in S$, faça*
4. *$C_1(s) \leftarrow 0$, $P_1(s) \leftarrow nil$, $L_1(s) \leftarrow \lambda(s)$, insira $s$ em $Q$.*
5. *Enquanto $Q \neq \emptyset$, faça*
6. *Remova de $Q$ uma amostra $s$ tal que $C_1(s)$ e minimo.*
7. *Insira $s$ em $Z_1'$.*
8. *Para cada $t \in Z_1$ tal que $C_1(t) > C_1(s)$, faça*
9. *Compute $cst \leftarrow \max\{C_1(s), d(s,t)\}$.*
10. *Se $cst < C_1(t)$, então*
11. *Se $C_1(t) \neq +\infty$, então remova $t$ de $Q$.*
12. *$P_1(t) \leftarrow s$, $L_1(t) \leftarrow L_1(s)$, $C_1(t) \leftarrow cst$.*
13. *Insira $t$ em $Q$.*
14. *Retorne o classificador $[P_1, C_1, L_1, Z_1']$.*

Após a fase de treinamento, o processo de classificação encontra o caminho ótimo entre uma amostra de teste $t \in Z_3$ (um procedimento similar é aplicado a amostras de $Z_2$) e um nó de treinamento, e $t$ recebe o rótulo da amostra em $Z_1$ que o conquistou, isto é, $L(t) \leftarrow L(P(t))$, onde $P(t)$ denota o predecessor de $t$ no caminho ótimo até $R(t)$. Em outras palavras, $P(t)$ denota a amostra que conquistou $t$. O Algoritmo 2 apresenta este procedimento.

**Algoritmo 2** – Classificação por Floresta de Caminhos Ótimos

| | |
|---|---|
| Entrada: | Classificador $[P_1, C_1, L_1, Z_1']$, conjunto de teste $Z_3$, e o par $(v, d)$ composto pelo vetor de características e as distâncias computadas. |
| Saída: | Rótulo $L_2$ e predecessor $P_2$ mapas definidos por $Z_3$, e valor de acurácia $Acc$. |
| Auxiliares: | Variáveis de custo $tmp$ e $mincost$. |

1. *Para cada* $t \in Z_3$, *faça*
2. $\quad i \leftarrow 1$, $mincost \leftarrow max\{C_1(k_i), d(k_i, t)\}$.
3. $\quad L_2(t) \leftarrow L_1(k_i)$ *e* $P_2(t) \leftarrow k_i$.
4. $\quad$ *Enquanto* $i < |Z_1'|$ *e* $mincost > C_1(k_{i+1})$, *faça*
5. $\quad\quad$ *Compute* $tmp \leftarrow \max\{C_1(k_{i+1}, d(k_{i+1}, t)\}$.
6. $\quad\quad$ *Se* $tmp < mincost$, *então*
7. $\quad\quad\quad$ $mincost \leftarrow tmp$.
8. $\quad\quad\quad$ $L_2(t) \leftarrow L(k_{i+1})$ *e* $P_2(t) \leftarrow k_{i+1}$.
9. $\quad\quad$ $i \leftarrow i + 1$.
10. *Calcule a acuracia Acc de acordo com [Papa, Falcão e Suzuki 2009].*
11. *Retorne* $[L_2, P_2, Acc]$.

O algoritmo baseado em OPF pode ser utilizado com qualquer função de custo suave que pode agrupar amostras com propriedades similares [Falcão, Stolfi e Lotufo 2004]. Na versão OPF com grafo completo a função de custo abordada foi a $f_{max}$ (Equação 4.1). O algoritmo baseado em OPF associa um caminho ótimo $P^*(s)$ de $S$ a toda amostra $s \in Z_1$, formando uma floresta de caminhos ótimos $P$ (uma função sem ciclos, a qual associa a todo $s \in Z_1$ seu predecessor $P(s)$ em $P^*(s)$, ou uma marca $nil$ quando $s \in S$, como mostrado na Figura 4.1d). Seja $R(s) \in S$ a raiz de $P^*(s)$ a qual pode ser alcançada por $P(s)$.

$$\begin{aligned} f_{max}(\langle s \rangle) &= \begin{cases} 0 & \text{se } s \in S, \\ +\infty & \text{caso contrário} \end{cases} \\ f_{max}(\pi \cdot \langle s, t \rangle) &= \max\{f_{max}(\pi), d(s, t)\}, \end{aligned} \tag{4.1}$$

sendo que $f_{max}(\pi)$ computa a distância máxima entre amostras adjacentes em $\pi$, quando $\pi$ não é um caminho trivial. A Figura 4.1 ilustra todos os passos executados pelos algoritmos de treinamento, classificação e testes do classificador de padrões OPF.

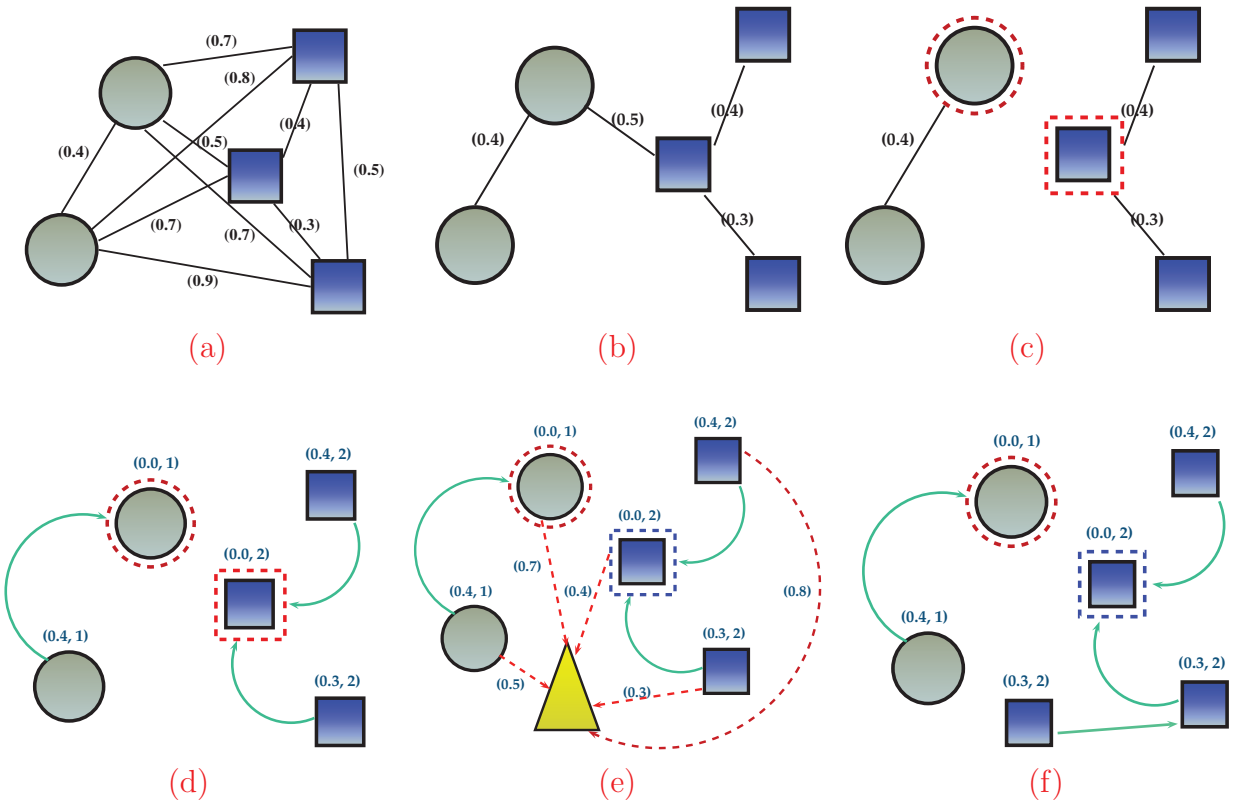



Figura 4.1: (a) Grafo completo ponderado nas arestas para um determinado conjunto de treinamento. (b) MST do grafo completo. (c) Protótipos escolhidos como sendo os elementos adjacentes de classes diferentes na MST (nós circulados). (d) Floresta de caminhos ótimos resultante para a função de valor de caminho $f_{max}$ e dois protótipos. Os identificadores $(x, y)$ acima dos nós são, respectivamente, o custo e o rótulo dos mesmos. A seta indica o nó predecessor no caminho ótimo. (e) Uma amostra de teste (triangulo) é conectada aos demais nós de ambas as classes (linhas pontilhadas) com os nós do conjunto de treinamento. (f) O caminho ótimo do protótipo mais fortemente conexo, seu rótulo 2 e o custo de classificação 0.3 são associados a amostra de teste. Note que, mesmo a mostra de teste estando mais próxima de um nó da classe 1 (Círculo), ela foi classificada como sendo da classe 2 (Quadrado).

### 4.1.2 Algoritmo de Poda do Conjunto de Treinamento

Grandes bases de dados usualmente apresentam redundância. Assim sendo, pelo menos em teoria, seria possível estimar um conjunto reduzido de treinamento ($Z_1$) com os padrões mais relevantes para classificação. Nesse sentido, a utilização de um conjunto

de treinamento e de avaliação ($Z_2$) tem permitido ao OPF aprender as amostras mais relevantes do conjunto de treinamento para a classificação de erros no conjunto de avaliação, trocando amostras de $Z_2$ classificadas erroneamente entre amostras não protótipos de $Z_1$ [Papa, Falcão e Suzuki 2009]. Nesta estratégia de aprendizado, $Z_1$ permanece com o mesmo tamanho e a instância do classificador com maior acurácia em $Z_2$ é selecionada para ser avaliada no conjunto de teste $Z_3$. O Algoritmo 3 implementa esta ideia.

**Algoritmo 3** – Algoritmo de aprendizado por Floresta de Caminhos Ótimos

Entrada: Conjuntos de treinamento e avaliação $Z_1$ e $Z_2$, respectivamente, $\lambda$-rotulados, número $T$ de iterações, e o par $(v, d)$ composto pelo vetor de características e as distâncias computadas.

Saída: Floresta de Caminhos Ótimos $P_1$, mapa de custos $C_1$, mapa de rótulos $L_1$, conjunto ordenado $Z_1'$ e $MaxAcc$.

Auxiliares: Vetores $FP$ e $FN$ de tamanho $c$ para falso positivos e falso negativos, conjunto $S$ de protótipos, e lista $LM$ de amostras classificadas erroneamente.

1. *Determine* $MaxAcc \leftarrow -1$.
2. *Para cada iteração* $I = 1, 2, \ldots, T$, *faça*
3. $\quad$ $LM \leftarrow \emptyset$ *e compute o conjunto* $S \subset Z_1$ *de protótipos.*
4. $\quad$ $[P_1, C_1, L_1, Z_1'] \leftarrow$ *Algoritmo 1*$(Z_1, S, (v, d))$.
5. $\quad$ *Para cada classe* $i = 1, 2, \ldots, c$, *faça*
6. $\quad\quad$ $FP(i) \leftarrow 0$ *e* $FN(i) \leftarrow 0$.
7. $\quad$ $[L_2, P_2, Acc] \leftarrow$ *Algoritmo 2*$([P_1, C_1, L_1, Z_1'], Z_2, (v, d))$
8. $\quad$ *Se* $Acc > MaxAcc$ *então*
9. $\quad\quad$ *Salve a atual instancia* $[P_1, C_1, L_1, Z_1']$
10. $\quad$ *do classificador e conjunto* $MaxAcc \leftarrow Acc$.
11. $\quad$ *Enquanto* $LM \neq \emptyset$
12. $\quad\quad$ $LM \leftarrow LM \backslash t$.
13. $\quad\quad$ *Substitua* $t$ *por uma amostra não-protótipo,*
14. $\quad\quad$ *randomicamente selecionada de* $Z_1$.
15. *Retorne a instância do classificador* $[P_1, C_1, L_1, Z_1']$ *com maior*
16. *acurácia em* $Z_2$, *em seu valor* $MaxAcc$.

Além disso, Papa et al. [Papa et al. 2010] propuseram um algoritmo que tem por objetivo identificar as amostras mais relevantes de $Z_1$ por meio da classificação do conjunto de avaliação $Z_2$. A ideia consiste, basicamente, em marcar cada amostra do conjunto de treinamento até sua raiz na árvore de caminhos ótimos que classificou uma dada amostra de $Z_2$. Este procedimento é repetido até que um critério de convergência tenha sido satisfeito. Ao final do processo, as amostras não marcadas são removidas do conjunto de treinamento.

A abordagem acima visa aprender as amostras mais relevantes do conjunto de treinamento e também reduzir o seu tamanho. Como a complexidade computacional do OPF para classificação é proporcional ao número de amostras de treinamento, podemos acelerar o processo diminuindo o tamanho do conjunto de treinamento. Embora a acurácia no conjunto de teste possa ser afetada, em algumas aplicações não é percebido tal comportamento [Papa et al. 2010] [Pisani et al. 2011] [Pereira et al. 2011].

O critério de parada é dado pela variável $MLoss$, a qual é definida como a diferença absoluta entre a taxa de acerto no conjunto de avaliação utilizando o conjunto de treinamento original, isto é, sem poda, e a taxa de acerto utilizando o conjunto de treinamento reduzido. O Algoritmo 4 implementa esse processo.

**Algoritmo 4** – Algoritmo de aprendizado com poda por Floresta de Caminhos Ótimos

| | |
|---|---|
| Entrada: | Conjunto de treinamento e avaliação, $Z_1$ e $Z_2$, $\lambda$-rotulados, o par $(v, d)$ composto pelo vetor de características e as distâncias computadas, variável $MLoss$ e número $T$ de iterações. |
| Saída: | Classificador $[P_1, C_1, L_1, Z_1']$ com conjunto de treinamento reduzido. |
| Auxiliares: | Conjunto $\mathcal{R}$ de amostras relevantes, e variáveis $Acc$ e $tmp$. |

1. $[P_1, C_1, L_1, Z_1'] \leftarrow$ *Algoritmo 3*$(Z_1, Z_2, T, (v, d))$.
2. $[L_2, P_2, Acc] \leftarrow$ *Algoritmo 2*$([P_1, C_1, L_1, Z_1'], Z_2, (v, d))$.
3. $tmp \leftarrow Acc$ *e* $\mathcal{R} \leftarrow \emptyset$.
4. *Enquanto* $|Acc - tmp| \leq MLoss$ *e* $\mathcal{R} \neq Z_1$ *faça*

5. $\mathcal{R} \leftarrow \emptyset$.
6. *Para cada amostra* $t \in Z_2$, *faça*
7. $\quad s \leftarrow P_2(t) \in Z_1$.
8. $\quad$ *Enquanto* $s \neq nil$, *faça*
9. $\quad\quad \mathcal{R} \leftarrow \mathcal{R} \cup s$.
10. $\quad\quad s \leftarrow P_1(s)$.
11. *Mova as amostras de* $Z_1 \backslash \mathcal{R}$ *para* $Z_2$.
12. $[P_1, C_1, L_1, Z_1'] \leftarrow$ *Algoritmo 3*$(Z_1, Z_2, T, (v, d))$.
13. $[L_2, P_2, tmp] \leftarrow$ *Algoritmo 2*$([P_1, C_1, L_1, Z_1'], Z_2, (v, d))$.
14. *Retorne* $[P_1, C_1, L_1, Z_1']$.

# 5 Metodologia

Nesta seção, descrevemos a metodologia aplicada para avaliar a robustez do classificador OPF com o intuito de detectar anomalias em redes de computadores, bem como os conjuntos de dados utilizados para esta tarefa e os resultados obtidos.

Neste trabalho foram utilizados três conjuntos de dados, conforme descrito abaixo:

- IDS_Bag[1]: este conjunto de dados compreende uma coleção de amostras coletadas no Massachusetts Institute of Technology-Lincoln Lab durante quatro dias de uma semana . Os conjuntos de dados originais foram mapeados para uma coleção de chamadas do sistema, e foram originalmente concebidos para detectar o uso indevido e anomalias em redes de computadores;

- KddCup99[2]: este conjunto de dados é composto por centenas de milhares de amostras, sendo dividido em 23 classes, nas quais 22 são relacionadas com os ataques de rede, e a classe restante representa um acesso normal à rede. Esta base contém amostras com 41 características, sendo que nesse trabalho utilizamos um conjunto de dados reduzido composto por 1% do conjunto original, dado que o custo computacional para o treinamento de algumas das técnicas de classificação comparadas no trabalho seria inviável; e

- NSL-KDD[3]: este conjunto de dados é uma proposta de [Tavallaee et al. 2009] na busca de minimizar registros duplicados nos conjuntos de treinamento e testes da

[1] `http://www.cs.iastate.edu/~dkkang/IDS_Bag`
[2] `http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`
[3] `http://www.iscx.ca/NSL-KDD/`

base de dados KddCup, sendo que para esse experimento utilizamos um conjunto reduzido composto por 35% da base.

A Tabela 5.1 apresenta algumas informações adicionais sobre as bases de dados utilizadas.

| Conjunto de Dados | Número de Amostras | Número de Classes | Número de Características |
|---|---|---|---|
| **KddCup** | 48989 | 23 | 41 |
| **NSL-KDD** | 44090 | 02 | 41 |
| **IDS_Bag Segunda-Feira** | 228 | 02 | 54 |
| **IDS_Bag Terça-Feira** | 445 | 02 | 53 |
| **IDS_Bag Quinta-Feira** | 375 | 02 | 52 |
| **IDS_Bag Sexta-Feira** | 251 | 02 | 53 |

Tabela 5.1: Informações sobre as bases de dados utilizadas.

Os experimentos realizados para avaliar a robustez do classificador OPF na tarefa de detecção de intrusão em redes de computadores foram conduzidos em duas etapas: na primeira (Seção 6.1) foram comparados OPF com SVM-RBF, SOM e um classificador Bayesiano nas bases de dados utilizando os conjuntos de treinamento ($Z_1$) e teste ($Z_3$), e no outro experimento (Seção 6.2) introduzimos o algoritmo de poda do conjunto de treinamento por OPF no contexto dos sistemas de detecção de intrusão. Nesta última etapa, foi utilizado também o conjunto de avaliação ($Z_2$).

Com o intuito de avaliar a robustez das técnicas de reconhecimento de padrões supervisionadas, foi empregado o método tradicional de validação cruzada em ambas etapas acima descritas. Os experimentos foram executados 10 vezes com os conjuntos de treinamento, avaliação e teste gerados aleatoriamente objetivando o cálculo da taxa de reconhecimento média e desvio padrão, bem como o tempo médio de execução das técnicas empregadas. A Figura 5.1 apresenta a divisão dos conjuntos de dados com suas respectivas porcentagens. Cabe destacar que os valores utilizados para o particionamento dos conjuntos foram escolhidos empiricamente, conforme experiência prévia.

Com relação aos classificadores de padrões utilizados, destacamos os seguintes:

- OPF: foi utilizada a implementação da LibOPF [Papa, Suzuki e Falcão 2009], uma biblioteca para o desenvolvimento de classificadores de padrões baseados em floresta

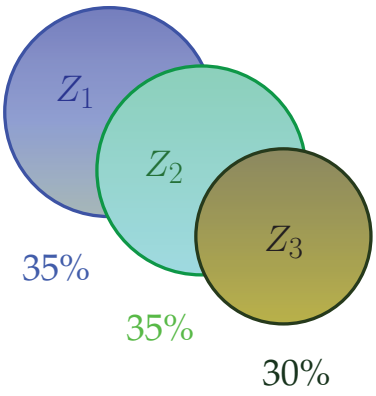



Figura 5.1: Porcentagem utilizada para a divisão das bases de dados utilizadas nos experimentos.

de caminhos ótimos;

- Bayes: corresponde ao classificador Bayesiano. Para os testes, foi utilizada nossa própria implementação;
- SVM-RBF: foi empregada a biblioteca SVMTorch [Collobert e Bengio 2001], a qual utiliza Máquina de Vetores de Suporte com função de núcleo de base radial (*Radial Basis Function* - RBF); e
- SOM: utilizamos nossa própria implementação para este algoritmo de análise de agrupamento. Com relação aos parâmetros, uma grade com $100 \times 100$ neurônios e 10 iterações para aprendizado foram utilizados.

A taxa de reconhecimento foi calculada levando em consideração que as classes podem ter diferentes tamanhos em $Z_2$ e $Z_3$ [Papa, Falcão e Suzuki 2009]. Caso tivéssemos duas classes, por exemplo, com diferentes tamanhos e um classificador sempre associasse o rótulo da classe com mais representantes, sua acurácia diminuiria devido a alta taxa de erro na classe com menor número de elementos.

Seja $NZ_2(i)$, $i = 1, 2, \ldots, c$, o número de elementos em $Z_2$ de cada classe $i$ (Similar definição é aplicado a $Z_3$). Definimos:

$$e_{i,1} = \frac{FP(i)}{|Z_2| - NZ_2(i)} \quad \text{e} \quad e_{i,2} = \frac{FN(i)}{NZ_2(i)}, \; i = 1, \ldots, c \tag{5.1}$$

onde $FP(i)$ e $FN(i)$ são os falsos positivos e falsos negativos, respectivamente. Isto significa que $FP(i)$ corresponde ao número de amostras de outras classes que foram classificadas como sendo da classe $i$ em $Z_2$, e $FN(i)$ é o número de amostras da classe $i$ que foram incorretamente classificadas como sendo de outras classes em $Z_2$. Os erros $e_{i,1}$ e $e_{i,2}$ são usados para definir:

$$E(i) = e_{i,1} + e_{i,2}, \tag{5.2}$$

onde $E(i)$ é a soma parcial dos erros da classe $i$. Finalmente, a acurácia $Acc$ do processo de classificação é dada por:

$$Acc = \frac{2c - \sum_{i=1}^{c} E(i)}{2c} = 1 - \frac{\sum_{i=1}^{c} E(i)}{2c}. \tag{5.3}$$

Cabe destacar que essa medida foi empregada para todos os algoritmos de classificação acima mencionados. As próximas seções tratam de explicar com mais detalhes as duas etapas de experimentos acima mencionadas.

## 5.1 Detecção de Intrusões em Redes de Computadores

Esta etapa dos experimentos objetiva avaliar o desempenho do classificador OPF no contexto de detecção de intrusões em redes de computadores, bem como comparar a sua taxa de reconhecimento com outras técnicas de classificação de padrões supervisionadas da literatura.

Conforme mencionado, particionamos as bases de dados em três conjuntos: treinamento, avaliação e teste. Para essa primeira rodada de experimentos, utilizamos apenas

os conjuntos de treinamento ($Z_1$) e teste ($Z_3$) para avaliar os classificadores. A Figura 5.2 apresenta essa metodologia, para a qual foi utilizada validação cruzada com 10 iterações.

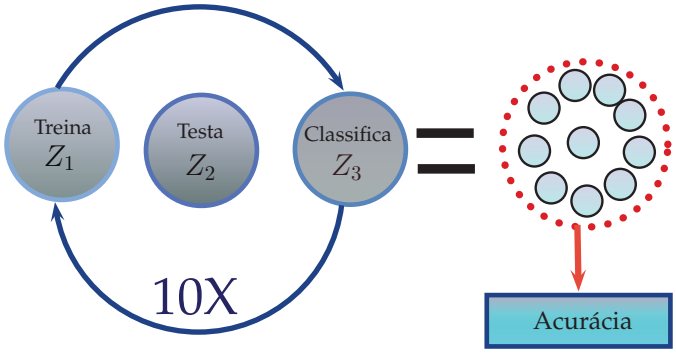



Figura 5.2: Conjuntos e método utilizado para encontrar a acurácia após a fase de treinamento e classificação.

Optou-se por esse número de iterações pelo fato de ser comumente empregado nas atividades e trabalhos científicos do grupo de pesquisa ao qual o presente trabalho está vinculado. Para esse experimento, serão computadas a taxa média de reconhecimento, desvio padrão e tempos médios de treinamento e teste em segundos, bem como o teste estatístico de McNemar [Kuncheva 2004], cujo detalhes são apresentados no Capítulo 6.

## 5.2 Avaliação do Algoritmo de Poda do Conjunto de Treinamento

Existem algumas situações nas quais temos poucos recursos computacionais como, por exemplo, capacidade de armazenamento reduzida e tempo de execução. Desde a etapa de treinamento, a qual muitas vezes requer um esforço computacional elevado, é desejável termos um conjunto de treinamento compacto e representativo. Papa et al. [Papa et al. 2010] propuseram um algoritmo de aprendizagem que é capaz de identificar as amostras de treinamento mais relevantes e descartar as demais. Este procedimento pode levar-nos a representações mais compactas do conjunto de treinamento, sem perda de generalização no conjunto de teste. A Figura 5.3 apresenta a arquitetura de utilização do algoritmo de poda do conjunto de treinamento por Floresta de Caminhos Ótimos.

A ideia consiste em marcar as amostras de treinamento que participaram de algum


Figura 5.3: Metodologia utilizada para a obtenção de um conjunto de treinamento mais compacto e representativo por meio da poda de amostras.

processo de classificação sobre um conjunto de avaliação. Em seguida, as amostras não marcadas são movidas para o conjunto de avaliação e o processo é repetido novamente até que algum critério de parada seja alcançado (Algoritmo 4). Em nosso caso, o critério de parada é dado pela variável $MLoss$, que é definida como a perda máxima de precisão no conjunto de avaliação com relação ao aprendizado no conjunto de treinamento original e no conjunto de treinamento reduzido.

Como exemplo, para $MLoss = 0.3$, isso significa que a diferença absoluta entre a eficácia sobre o conjunto de avaliação utilizando o conjunto de treinamento original e a eficácia utilizando o conjunto de treinamento reduzido é limitada em 30%. Finalmente, o conjunto de treinamento reduzido será utilizado para avaliar a eficácia sobre o conjunto de teste. Assim, o parâmetro $MLoss$ pode ser visto como uma relação custo/benefício entre a taxa de poda do conjunto de treinamento e a eficiência sobre o conjunto de teste. Uma escolha de altos valores para $MLoss$, pode levar o algoritmo a podar muitas amostras, afetando a eficácia no conjunto de teste. Caso contrário, baixos valores de $MLoss$ podem levar-nos a baixas taxas de poda no conjunto de treinamento.

Nesta etapa é então utilizado o conjunto de avaliação ($Z_2$), o qual não foi empregado na etapa anterior. Assim sendo, este conjunto é utilizado pelo algoritmo de poda para o aprendizado das amostras de treinamento irrelevantes, as quais são então descartadas pelo método.

# 6 Resultados Experimentais

Nesta seção descrevemos os resultados dos experimentos realizados para avaliar a robustez do classificador OPF na tarefa de detecção de intrusões em redes de computadores, bem como o desempenho do algoritmo de poda do conjunto de treinamento.

As bases de dados utilizadas neste trabalho utilizam o formato de arquivo especificado na LibOPF [Papa, Suzuki e Falcão 2009], o qual é então mapeado para utilização nos outros classificadores empregados neste trabalho. A Figura 6.1 apresenta um exemplo de uma parte da base IDS_Bag (Segunda-feira) para um melhor entendimento deste formato de arquivo. Os valores contidos na primeira linha representam o número de amostras, número de classes e o número de características por amostra, respectivamente. As demais linhas contém um identificador da amostra (número inteiro de 0 a $n-1$, onde $n$ corresponde ao tamanho da base), o número da classe (número natural de 1 a $c$, onde $c$ denota o número de classes) e o valor das características.

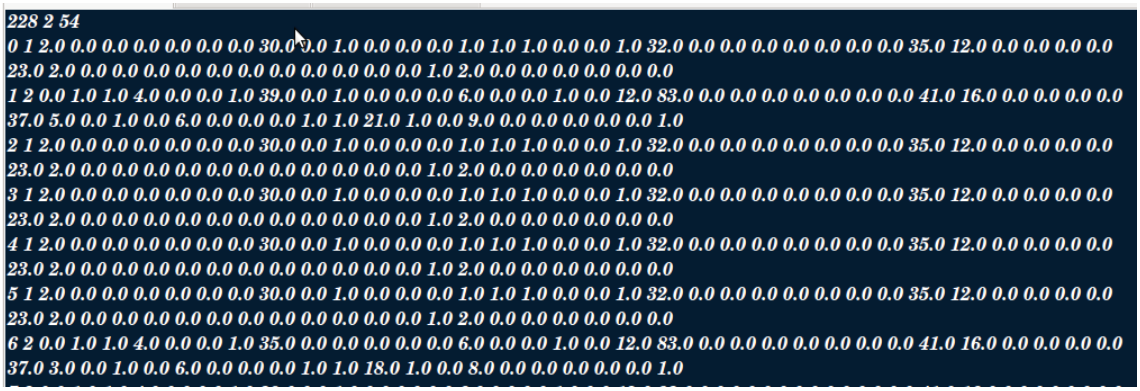



Figura 6.1: Formato de arquivo requerido pela LibOPF [Papa, Suzuki e Falcão 2009] (IDS_Bag - Segunda-feira).

Os experimentos realizados com as bases de dados citadas anteriormente foram conduzidos em duas etapas, as quais são descritas nas próximas seções. Cabe destacar que, para a primeira etapa dos experimentos, foi realizada uma análise estatística por meio do teste de McNemar [Kuncheva 2004], o qual objetiva comparar os resultados obtidos pelos classificadores aos pares. A Equação 6.1 apresenta a formulação desse teste, tal que o valor de $\chi^2$ é computado como segue:

$$\chi^2 = \frac{((N_{01} - N_{10}) - 1)^2}{N_{01} + N_{10}}, \tag{6.1}$$

onde $N_{01}$ representa o número de vezes que a primeira técnica errou e o número que a segunda acertou, e $N_{10}$ o número de vezes que a primeira acertou e a segunda errou. Tomou-se como hipótese nula que as duas técnicas obtiveram resultados similares. Assim, se o valor de $\chi^2$ é maior que 3.84 ($\rho = 0.05$), a hipótese nula pode ser rejeitada, concluindo que as técnicas comparadas apresentam resultados distintos. Esse valor de significância $\rho$ foi adotado empiricamente. Pode ser notado que os classificadores mais eficazes (em negrito) são OPF e Bayesiano, sendo que os mesmos podem ser considerados similares caso utilizemos o desvio padrão.

## 6.1 Eficácia dos Classificadores para Detecção de Intrusões

Nesta seção são apresentados e discutidos os resultados da primeira etapa dos experimentos. As próximas seções apresentam os resultados de cada base separadamente.

### 6.1.1 Base de Dados KddCup

A Tabela 6.1 apresenta a taxa média de reconhecimento e seu desvio padrão, bem como os tempos médios de treinamento e teste em segundos para a base de dados KddCup.

Como mencionado na seção anterior, executamos também o teste estatístico de McNemar com o intuito de adicionar uma robustez aos experimentos. A Tabela 6.2 apresenta os resultados obtidos para a base de dados KddCup, sendo que os valores de $\chi^2$ são

| Classificador | Acurácia | Treinamento [s] | Teste [s] |
|---|---|---|---|
| **OPF** | **93.19±2.49** | **13.109** | **20.134** |
| **Bayes** | **91.38±2.49** | **4.9268** | **263.439** |
| SVM-RBF | 89.56± 3.91 | 5018.6606 | 1.9853 |
| SOM | 89.95±1.97 | 3121.4089 | 8.05812 |

Tabela 6.1: Resultados obtidos na base KddCup. Os classificadores mais eficazes estão em negrito.

computados para todos os pares de métodos.

| $\chi^2$ | OPF | Bayes | SVM-RBF | SOM |
|---|---|---|---|---|
| **OPF** | - | 0.0865 | 7105.0001 | 7134.0001 |
| **Bayes** | 0.0865 | - | 2.4622 | 1.3452 |
| **SVM-RBF** | 7105.0001 | 2.4622 | - | 0.1488 |
| **SOM** | 7134.0001 | 1.3452 | 0.1488 | - |

Tabela 6.2: Valores de $\chi^2$ computado para todos os pares de métodos com relação a base de dados KddCup.

Como dito anteriormente, para um valor de $\rho = 0.05$ temos que a hipótese nula (os classificadores são similares) é rejeitada quando $\chi^2 > 3.84$. Assim sendo, temos que os pares de classificadores (OPF,Bayes), (Bayes,SVM-RBF), (Bayes,SOM) e (SVM-RBF,SOM) são considerados similares.

Desta forma, podemos considerar OPF e Bayes como sendo as técnicas mais eficazes, com OPF sendo 13.08 vezes mais rápido que Bayes para a fase de teste e este 2.66 vezes mais rápido para a etapa de treinamento. Adicionalmente, se considerarmos o tempo total de execução, isto é, treinamento e teste, o classificador OPF é 8.07 vezes mais eficiente que o classificador Bayesiano.

### 6.1.2 Base de Dados NSL-KDD

A Tabela 6.3 apresenta os resultados obtidos para a base de dados NSL-KDD.

Desta forma, podemos considerar OPF e Bayes como sendo as técnicas mais eficazes, com OPF sendo 1.94 vezes mais rápido que Bayes para a fase de teste e este 3.70 vezes mais rápido para a etapa de treinamento. Considerando o tempo total de execução, isto

| Classificador | Conjunto de Dados | Acurácia | Treinamento [s] | Teste [s] |
|---|---|---|---|---|
| OPF | NSL-KDD | 99.50±0.017 | 125.8236 | 425.9925 |
| Bayes | NSL-KDD | 99.50±0.021 | 33.9259 | 219.1123 |
| SVM-RBF | NSL-KDD | 89.56± 3.91 | 5018.6606 | 1.9853 |
| SOM | NSL-KDD | 99.38±0.025 | 9650.4257 | 23.2324 |

Tabela 6.3: Resultados obtidos na base NSL-KDD. Os classificadores mais eficazes estão em negrito.

é, treinamento e teste, o classificador OPF é 2.18 vezes mais eficiente que o classificador Bayesiano. A Tabela 6.4 apresenta os resultados obtidos para a base de dados NSL-KDD, sendo que os valores de $\chi^2$ são computados para todos os pares de métodos.

| $\chi^2$ | OPF | Bayes | SVM-RBF | SOM |
|---|---|---|---|---|
| **OPF** | - | 0.0160 | 0.7901 | 0.6545 |
| **Bayes** | 0.0160 | - | 0.0208 | 0.0208 |
| **SVM-RBF** | 0.7901 | 0.0208 | - | 1 |
| **SOM** | 0.6545 | 0.0208 | 1 | - |

Tabela 6.4: Valores de $\chi^2$ para os pares de métodos na base de dados NSL-KDD.

Conforme já apresentado, para um valor de $\rho = 0.05$ temos que a hipótese nula (os classificadores são similares) é rejeitada quando $\chi^2 > 3.84$. Assim sendo, temos que todos os pares de classificadores são considerados similares.

### 6.1.3 Base de Dados IDS_Bag

A Tabela 6.5 apresenta os resultados obtidos para o conjunto de dados IDS_Bag, o qual é composto por 4 subgrupos representando diferentes dias da semana.

Como pode ser notado, o classificador OPF esteve entre as melhores técnicas (eficácia) em 3 dos 4 subgrupos da base IDS_Bag, sendo que para o conjunto "Segunda-Feira" o seu resultado foi bastante próximo dos demais. Foi também observado que OPF foi o mais eficiente para a classificação dos dados em todos os conjuntos, sendo o classificador Bayesiano o mais rápido na etapa de treinamento dos dados.

As Tabelas 6.6, 6.7, 6.8 e 6.9 apresentam os valores de $\chi^2$ para as bases de dados "Segunda-Feira", "Terça-Feira", "Quinta-Feira" e "Sexta-Feira", respectivamente. Pode ser

| Classificador | Conjunto de Dados | Acurácia | Treinamento[s] | Teste[s] |
|---|---|---|---|---|
| OPF | Segunda-Feira | 99.01±0.00 | 0.000365 | 0.000905 |
| Bayes | Segunda-Feira | 100.00±0.00 | 0.000265 | 0.002026 |
| SVM-RBF | Segunda-Feira | 100.00±0.00 | 0.835137 | 0.008105 |
| SOM | Segunda-Feira | 99.76±0.46 | 34.81 | 0.109492 |
| OPF | Terça-Feira | 97.46±5.36 | 0.002504 | 0.002184 |
| Bayes | Terça-Feira | 98.27±0.00 | 0.001034 | 0.008875 |
| SVM-RBF | Terça-Feira | 99.56±0.43 | 1.101572 | 0.009766 |
| SOM | Terça-Feira | 99.18±0.48 | 63.09 | 0.182244 |
| OPF | Quinta-Feira | 99.47±0.00 | 0.001651 | 0.001589 |
| Bayes | Quinta-Feira | 99.47±0.00 | 0.000712 | 0.005399 |
| SVM-RBF | Quinta-Feira | 99.79±0.25 | 0.729479 | 0.005907 |
| SOM | Quinta-Feira | 99.79±0.25 | 55.37 | 0.165348 |
| OPF | Sexta-Feira | 95.83±0.00 | 0.000378 | 0.000354 |
| Bayes | Sexta-Feira | 95.83±0.00 | 0.000152 | 0.002575 |
| SVM-RBF | Sexta-Feira | 90.83±3.11 | 0.856063 | 0.006249 |
| SOM | Sexta-Feira | 91.50±4.11 | 40.15 | 0.117429 |

Tabela 6.5: Resultados obtidos na base IDS_Bag. Os classificadores mais eficazes estão em negrito.

observado nas Tabelas 6.6, 6.7 e 6.8 que houve similaridade entre os pares de classificadores (Bayes,SVM-RBF), (Bayes,SOM) e (SVM,SOM) para as bases de dados "Segunda-Feira", "Terça-Feira"e "Quinta-Feira". Já para a Tabela 6.9, ou seja, base de dados "Sexta-Feira", os pares de classificadores similares foram (OPF,SVM-RBF), (Bayes,SVM-RBF) e (Bayes,SOM).

| $\chi^2$ | OPF | Bayes | SVM-RBF | SOM |
|---|---|---|---|---|
| **OPF** | - | 67.0145 | 70.0139 | 73.0133 |
| **Bayes** | 67.0145 | - | 0.0519 | 0.3378 |
| **SVM-RBF** | 70.0139 | 0.0519 | - | 0.0635 |
| **SOM** | 73.0133 | 0.3378 | 0.0635 | - |

Tabela 6.6: Valores de $\chi^2$ para os pares de métodos na base de dados Segunda-Feira.

| $\chi^2$ | OPF | Bayes | SVM-RBF | SOM |
|---|---|---|---|---|
| **OPF** | - | 46.0208 | 44.0217 | 38.0250 |
| **Bayes** | 46.0208 | - | 0.0200 | 1.1667 |
| **SVM-RBF** | 44.0217 | 0.0200 | - | 0.6579 |
| **SOM** | 38.0250 | 1.1667 | 0.6579 | - |

Tabela 6.7: Valores de $\chi^2$ para os pares de métodos na base de dados Terça-Feira.

| $\chi^2$ | OPF | Bayes | SVM-RBF | SOM |
|---|---|---|---|---|
| **OPF** | - | 48.0200 | 66.0147 | 34.0278 |
| **Bayes** | 48.0200 | - | 0.3404 | 4.2250 |
| **SVM-RBF** | 66.0147 | 0.3404 | - | 0.7206 |
| **SOM** | 34.0278 | 4.2250 | 0.7206 | - |

Tabela 6.8: Valores de $\chi^2$ para os pares de métodos na base de dados Quinta-Feira.

| $\chi^2$ | OPF | Bayes | SVM-RBF | SOM |
|---|---|---|---|---|
| **OPF** | - | 14.4500 | 1 | 19.0476 |
| **Bayes** | 14.4500 | - | 0 | 0.2667 |
| **SVM-RBF** | 1 | 0 | - | 19.0476 |
| **SOM** | 19.0476 | 0.2667 | 19.0476 | - |

Tabela 6.9: Valores de $\chi^2$ para os pares de métodos na base de dados Sexta-Feira.

## 6.2 Redução do Conjunto de Treinamento por Poda de Amostras

Esta seção tem por objetivo apresentar os resultados dos experimentos com a poda das amostras do conjunto de treinamento. Dado que a complexidade da etapa de classificação do OPF é dada em função dos conjuntos de treinamento e teste, o objetivo da poda é obter um conjunto de treinamento mais compacto e representativo. Assim, podemos tornar a etapa de classificação mais eficiente.

Em certas aplicações que possuem pouca redundância nos dados, a poda do conjunto de treinamento pode não ser uma boa solução. Entretanto, em problemas associados à detecção de intrusões em redes de computadores existe uma certa redundância, dado que podem existir muitos acessos simultâneos e similares à uma mesma rede. Cabe destacar que grande parte das amostras da base correspondem à acessos normais. A Tabela 6.10 apresenta as taxas médias de reconhecimento e desvio padrão, bem como os tempos médios de treinamento e classificação em segundos, o valor médio da variável $MLoss$ e também da taxa de poda.

| Classificador | Conjunto de Dados | Acurácia | Treinamento [s] | Testes [s] | $MLoss$ | Taxa de poda |
|---|---|---|---|---|---|---|
| OPF | KddCup | 93.19±2.49 | 13.109 | 20.134 | - | |
| OPF Pruning | KddCup | 91.19±2.49 | 2.866 | 3.35 | 0.6 | 99.88% |
| OPF | NSL-KDD | 99.50±0.017 | 125.8236 | 425.9925 | - | |
| OPF Pruning | NSL-KDD | 95.19±0.49 | 19.79 | 67.011 | 0.6 | 72.67% |
| OPF | IDS_Bag (Segunda) | 99.019±0.00 | 0.000365 | 0.000905 | - | - |
| OPF Pruning | IDS_Bag (Segunda) | 99.019±0.00 | 0.000530 | 0.001477 | 0.43 | 89.09% |
| OPF | IDS_Bag (Terça) | 97.466±5.36 | 0.002504 | 0.002184 | - | - |
| OPF Pruning | IDS_Bag (Terça) | 97.466±5.36 | 0.001687 | 0.001665 | 0.35 | 91.72% |
| OPF | IDS_Bag (Quinta) | 99.479±0.00 | 0.001651 | 0.001589 | - | - |
| OPF Pruning | IDS_Bag (Quinta) | 99.479±0.00 | 0.000705 | 0.001199 | 0.65 | 88.38% |
| OPF | IDS_Bag (Sexta) | 95.833±0.00 | 0.000378 | 0.000354 | - | - |
| OPF Pruning | IDS_Bag (Sexta) | 95.833±0.00 | 0.000579 | 0.000518 | 0.51 | 87.09% |

Tabela 6.10: Experimentos realizados por meio da poda de amostras do conjunto de treinamento.

Como pode ser notado, existe uma grande redundância nas bases de dados empregadas

no presente trabalho, chegando a 99.88% de poda no caso da base KddCup. Este fato reflete diretamente nos tempos de treinamento e classificação, os quais tornam-se bastante reduzidos. Em algumas bases, a taxa de reconhecimento não sofre alterações, como é o caso da IDS_Bag. Já para as bases de dados KddCup e NSL-KDD, existe um decréscimo na eficácia, mais precisamente de 2.19% e 4.52%, respectivamente.

Cabe destacar que o valor de $MLoss$ foi obtido conforme proposto por Nakamura et al. [Nakamura et al. 2011], o qual utiliza o algoritmo de otimização evolucionista Busca Harmônica (*Harmony Search* - HS) [Geem 2009] para encontrar o valor de $MLoss$ que minimiza o tamanho do conjunto de treinamento e, ao mesmo tempo, tenta maximizar a taxa de acerto no conjunto de avaliação.

# 7 Conclusão

Sistemas de detecção de intrusões em redes de computadores tem sido fundamentais nos últimos anos [Kompella e Varghese 2004]. Como o número de acessos maliciosos e não-autorizados à informações privadas tem aumentado, uma atenção especial tem sido dedicada a sistemas de detecção de intrusões que fazem uso de inteligência artificial para melhorar a sua eficácia. Tais sistemas são capazes de aprender e detectar novos ataques, mas nem sempre com eficiência razoável. Com o intuito de aliar eficiência no treinamento dos dados e eficácia em sua classificação, propusemos neste trabalho um sistema de detecção de intrusão baseado no classificador Floresta de Caminhos Ótimos, ressaltando que esta técnica está sendo aplicada pela primeira vez a esse contexto.

Os experimentos foram realizados em duas fases considerando três bases de dados: KDDCup, NSL-KDD e IDS_Bag. Na primeira etapa, foram comparados os classificadores OPF, SVM-RBF, SOM e um classificador Bayesiano, sendo que no segundo experimento mostramos como tornar a etapa de classificação do OPF mais rápida para a detecção de ataques em redes de computadores. A ideia consiste em utilizar o algoritmo de poda por OPF com o intuito de encontrar um conjunto de treinamento mais compacto e representativo.

Na primeira rodada de experimentos, mostramos que OPF, SVM-RBF e o classificador Bayesiano obtiveram resultados semelhantes, sendo OPF a abordagem mais rápida se considerarmos o tempo de execução total, ou seja, treinamento e teste. Em seguida, na segunda rodada de experimentos, mostramos que a etapa de classificação da técnica OPF pode ser otimizada utilizando um algoritmo de poda do conjunto de treinamento, sem afetar muito a precisão no conjunto de teste. O presente trabalho serviu como ponto

principal para a utilização do classificador OPF no contexto de detecção de intrusões em redes de computadores, sendo que o mesmo mostrou-se bastante eficiente para esta tarefa.

# 8 Trabalhos Aceitos para Publicação

- Artigos em conferências
  - Intrusion Detection in Computer Networks Using Optimum-Path Forest Clustering. *IEEE Conference on Local Computer Networks*, Flórida - USA/2012 (Qualis A2 - CC);
  - Intrusion Detection System Using Optimum-Path Forest. *IEEE Conference on Local Computer Networks*, Bonn - Alemanha/2011 (Qualis A2 - CC);
  - Optimum-Path Forest Pruning Parameter Estimation Through Harmony Search. *XXIV SIBGRAPI - Conference on Graphics, Patterns and Images*, Maceió - Brasil/2011 (Qualis B1 - CC);
  - Metallic Precipitates Segmentation from Scanning Electron Microscope Images Through Machine Learning Techniques. *International Workshop on Combinatorial Image Analysis*, Madri/2011 (Qualis B4 - CC);
- Artigos em periódicos
  - An Optimum-Path Forest Framework for Intrusion Detection in Computer Networks. *Engineering Applications of Artificial Intelligence* (Qualis A2 - CC).

# Referências

[Cannady 1998]CANNADY, J. Artificial neural networks for misuse detection. In: *National Information Systems Security Conference*. [S.l.: s.n.], 1998. p. 443–456.

[Chang e Lin 2011]CHANG, C.-C.; LIN, C.-J. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, ACM, New York, NY, USA, v. 2, p. 27:1–27:27, 2011. ISSN 2157-6904. Disponível em: <http://doi.acm.org/10.1145/1961189.1961199>.

[Chen, Hsu e Shen 2005]CHEN, W. H.; HSU, S. H.; SHEN, H. P. Application of SVM and ANN for intrusion detection. *Computers and Operations Research*, Elsevier Science Ltd., Oxford, UK, UK, v. 32, n. 10, p. 2617–2634, 2005. ISSN 0305-0548.

[Collobert e Bengio 2001]COLLOBERT, R.; BENGIO, S. SVMTorch: support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, JMLR.org, v. 1, p. 143–160, September 2001.

[Cortes e Vapnik 1995]CORTES, C.; VAPNIK, V. Support vector networks. *Machine Learning*, v. 20, p. 273–297, 1995.

[Cortes e Vapnik 1995]CORTES, C.; VAPNIK, V. Support-vector networks. *Machine Learning*, Kluwer Academic Publishers, p. 273–297, 1995.

[Falcão, Stolfi e Lotufo 2004]FALCÃO, A.; STOLFI, J.; LOTUFO, R. The image foresting transform: Theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 1, p. 19–29, 2004.

[Falcão, Stolfi e Lotufo 2004]FALCÃO, A. X.; STOLFI, J.; LOTUFO, R. A. The image foresting transform theory, algorithms, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 1, p. 19–29, 2004.

[Geem 2009]GEEM, Z. W. *Music-Inspired Harmony Search Algorithm: Theory and Applications*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2009. ISBN 364200184X, 9783642001840.

[Ghosh, Wanken e Charron 1998]GHOSH, A. K.; WANKEN, J.; CHARRON, F. Detecting anomalous and unknown intrusions against programs. In: *Proceedings of the Annual Computer Security Application Conference*. [S.l.: s.n.], 1998. p. 259–267.

[Haijun et al. 2007]HAIJUN, X. et al. Ad hoc-based feature selection and support vector machine classifier for intrusion detection. In: *Proceedings of IEEE International Conference on Grey Systems and Intelligent Services*. [S.l.: s.n.], 2007. p. 18–20.

[Haykin 1994]HAYKIN, S. *Neural networks: a comprehensive foundation*. [S.l.]: Prentice Hall, 1994.

[Haykin 1998]HAYKIN, S. *Neural Networks: A comprehensive foundation*. [S.l.]: Prentice-Hall, 1998.

[Haykin 1999]HAYKIN, S. *Neural networks: a comprehensive foundation*. [S.l.]: Pearson, Prentice Hall, 1999.

[Jucá, Boukerche e Sobral 2002]JUCá, K. R. L.; BOUKERCHE, A.; SOBRAL, J. B. M. Intrusion detection based on the immune human system. In: *Proceedings of the 16th International Parallel and Distributed Processing Symposium*. Washington, DC, USA: IEEE Computer Society, 2002. p. 187. ISBN 0-7695-1573-8.

[Kayacik, Heywood e Heywood 2007]KAYACIK, H. G.; HEYWOOD, A. N. Z.; HEYWOOD, I. M. A hierarchical SOM-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 20, n. 4, p. 439–451, 2007. ISSN 0952-1976.

[Kizza 2009]KIZZA, J. M. *Guide to Computer Network Security*. [S.l.]: Springer, 2009.

[Kohonen 1989]KOHONEN, T. *Self-Organizing and Associative Memory*. [S.l.]: Springer-Verlag, 1989.

[Kohonen 1990]KOHONEN, T. Self-organizing map. *Proceedings of the IEEE*, v. 78, p. 1464–1480, 1990.

[Kohonen 1993]KOHONEN, T. Things you haven't heard about self-organiznig map. *IEEE International Conference on Neural Networks*, III, p. 1147–1156, 1993.

[Kompella e Varghese 2004]KOMPELLA, S. S. R. R.; VARGHESE, G. On scalable attack detection in the network. In: *Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*. New York, NY, USA: ACM, 2004. (IMC '04), p. 187–200.

[Kovacs 1996]KOVACS, Z. L. *Redes Neurais Artificiais: Fundamentos e Aplicações*. [S.l.]: Escola Politécnica da Universidade de São Paulo, 1996.

[Kuncheva 2004]KUNCHEVA, L. I. *Combining Pattern Classifiers: Methods and Algorithms*. [S.l.]: Wiley-Interscience, 2004.

[Lei e Ghorbani 2004]LEI, J. Z.; GHORBANI, A. A. Network intrusion detection using an improved competitive learning neural network. In: *Proceedings of the 2nd Annual Conference on Communication Networks and Services Research*. [S.l.]: IEEE Computer Society, 2004. p. 190–197.

[Levchenko 2004]LEVCHENKO, A. A. Institutional quality and international trade. In: . [S.l.: s.n.], 2004.

[Li e Gu 2009]LI, H.; GU, D. A novel intrusion detection scheme using support vector machine fuzzy network for mobile ad hoc networks. *Web Mining and Web-based Application, Pacific-Asia Conference on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 47–50, 2009.

[Nakamura et al. 2011]NAKAMURA, R. Y. M. et al. Optimum-path forest pruning parameter estimation through harmony search. In: *Proceedings of the 24th SIBGRAPI Conference on Graphics, Patterns and Images*. Washington, DC, USA: IEEE Computer Society, 2011. p. 181–188.

[Neelam e Saurabh 2012]NEELAM, S.; SAURABH, M. Layered approach for intrusion detection using naïve bayes classifier. In: *Proceedings of the International Conference on Advances in Computing, Communications and Informatics*. New York, NY, USA: ACM, 2012. p. 639–644.

[Oke, Loukas e Gelenbe 2007]OKE, G.; LOUKAS, G.; GELENBE, E. Detecting denial of service attacks with bayesian classifiers and the random neural network. In: *FUZZ-IEEE*. [S.l.]: IEEE, 2007. p. 1–6.

[Papa, Suzuki e Falcão 2009]PAPA, J.; SUZUKI, C.; FALCÃO, A. *LibOPF: A library for the design of optimum-path forest classifiers*. [S.l.], 2009. Software version 2.0 available at `http://www.ic.unicamp.br/~afalcao/LibOPF`.

[Papa e Falcão 2008]PAPA, J. P.; FALCÃO, A. X. A new variant of the optimum-path forest classifier. In: *Proceeding of the 4th International Symposium on Advances in Visual Computing*. Berlin, Heidelberg: Springer-Verlag, 2008. p. 935–944. ISBN 978-3-540-89638-8.

[Papa et al. 2012]PAPA, J. P. et al. Efficient supervised optimum-path forest classification for large datasets. *Pattern Recognition*, Elsevier Science Inc., New York, NY, USA, v. 45, n. 1, p. 512–520, 2012.

[Papa et al. 2010]PAPA, J. P. et al. Robust pruning of training patterns for optimum-path forest classification applied to satellite-based rainfall occurrence estimation. *IEEE Geoscience and Remote Sensing Letters*, v. 7, n. 2, p. 396–400, 2010.

[Papa, Falcão e Suzuki 2009]PAPA, J. P.; FALCÃO, A. X.; SUZUKI, C. T. N. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, Wiley-Interscience, v. 19, n. 2, p. 120–131, 2009.

[Papa, Falcão e Suzuki 2009]PAPA, J. P.; FALCÃO, A. X.; SUZUKI, C. T. N. Supervised pattern classification based on optimum-path forest. *International Journal of Imaging Systems and Technology*, John Wiley & Sons, Inc., New York, NY, USA, v. 19, n. 2, p. 120–131, 2009.

[Patel, Qassim e Wills 2010]PATEL, A.; QASSIM, Q.; WILLS, C. A survey of intrusion detection and prevention systems. *Information Management, Computer Security*, v. 18, p. 277–290, 2010.

[Pereira et al. 2011]PEREIRA, C. R. et al. Intrusion detection system using optimum-path forest. In: *LCN'11*. [S.l.: s.n.], 2011. p. 183–186.

[Phillips 1999]PHILLIPS, P. J. Support vector machines applied to face recognition. *Advances in Neural Information Processing Systems*, MIT Press, v. 11, p. 803–809, 1999.

[Pisani et al. 2011]PISANI, R. et al. Land use image classification through optimum-path forest clustering. In: *IGARSS'11*. [S.l.]: IEEE, 2011. p. 826–829.

[Raghu, Poongodi e Yegnanarayana 1995]RAGHU, P. P.; POONGODI, R.; YEGNANARAYANA, B. A combined neural network approach for texture classification. *Neural Networks*, v. 8, p. 975–987, 1995.

[Schölkopf e Smola 2002]SCHÖLKOPF, B.; SMOLA, A. J. *Learning with Kernels*. [S.l.]: MIT Press, 2002.

[Schölkopf, Williamson e Bartlett 2000]SCHÖLKOPF, B.; WILLIAMSON, R. C.; BARTLETT, P. L. New support vector algorithms. *Neural Computation*, MIT Press, p. 1207–1245, 2000.

[Shyu et al. 2003]SHYU, M. L. et al. A novel anomaly detection scheme based on principal component classifier. In: *Proceedings of the IEEE Foundations and New Directions of Data Mining Workshop*. [S.l.: s.n.], 2003. p. 172–179.

[Tang e Mazzoni 2006]TANG, B.; MAZZONI, D. Multiclass reduced-set support vector machines. In: *Proceedings of the 23th International Conference on Machine Learning*. New York, USA: ACM Press, 2006. p. 921–928. ISBN 1-59593-383-2.

[Tavallaee et al. 2009]TAVALLAEE, M. et al. A detailed analysis of the kdd cup 99 data set. In: *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. [S.l.: s.n.], 2009. p. 1 –6.

[Vapnik 1999]VAPNIK, V. N. An overview of statistical learning theory. *IEEE Transactions on Neural Networks*, v. 10, p. 988–999, 1999.

[Whitman e Mattord 2004]WHITMAN, M. E.; MATTORD, H. J. *Principles of Information Security*. Boston, MA, United States: Course Technology Press, 2004. ISBN 0619216255.

[Zanero e Savaresi 2004]ZANERO, S.; SAVARESI, S. M. Unsupervised learning techniques for an intrusion detection system. In: *Proceedings of the 2004 ACM symposium on Applied computing*. New York, NY, USA: ACM, 2004. p. 412–419. ISBN 1-58113-812-1.

[Zeltser et al. 2002]ZELTSER, L. et al. Desvendando seguranca em redes o guia definitivo para fortificacao de perimetros de rede usando firewall, vpns, roteadores e sistemas de deteccao de intrusao. Rio De Janeiro New Riders Pub. Campus, p. 2–10, 2002.

[Zheng, Hou e Wang 2011]ZHENG, H.; HOU, M.; WANG, Y. An efficient hybrid clustering-pso algorithm for anomaly intrusion detection. *JSW*, p. 2350–2360, 2011.

Autorizo a reprodução xerográfica para fins de pesquisa.

São José do Rio Preto, 23 de Julho de 2012

Assinatura