UNIVERSIDADE FEDERAL DO ACRE

EDKALLENN SILVA DE LIMA

Aplicação de Mineração de dados e Aprendizagem de Máquina na detecção de conflitos entre políticas

RIO BRANCO - ACRE

2020

EDKALLENN SILVA DE LIMA

Aplicação de Mineração de dados e Aprendizagem de Máquina na detecção de conflitos entre políticas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal do Acre como requisito parcial para a obtenção do Grau de Mestre em Ciência da Computação. Área de concentração: Engenharia de Sistemas de Informação

Aprovada em <MES> de 2020.

BANCA EXAMINADORA

Prof. DRA. LAURA COSTA SARKIS - Orientador, UFAC

Prof. <NOME DO AVALIADOR>, <INSTITUIÇÃO>

Prof. <NOME DO AVALIADOR>, <INSTITUIÇÃO>

Prof. <NOME DO AVALIADOR>, <INSTITUIÇÃO>

RIO BRANCO - ACRE

2020

Dedicatória(s): Este trabalho não seria possível sem a educação que me foi concedida, os ensinamentos e a sabedoria oriundos de você, Lucimar do Rego Albuquerque de Lima, muito mais que uma mãe, uma inspiração. <IN MEMORIAN>

"Assim como casas são feitas de pedras, a ciência é feita de fatos. Mas uma pilha de pedras não é uma casa e uma coleção de fatos não é, necessariamente, ciência".

***Jules Henri Poincaré**, matemático, físico e filósofo da ciência francês*

Agradecimentos

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Duis rutrum maximus fermentum. Duis id quam nibh. Aliquam cursus eget mauris at fermentum. Nam iaculis posuere sapien. Praesent facilisis nisl ipsum, at gravida est scelerisque non. In sed luctus sem. Integer odio sem, feugiat eu eros eget, lacinia lobortis mi. Donec dapibus, tellus non consequat aliquet, dui massa suscipit dolor, nec pretium tortor massa vitae lorem. Quisque non faucibus tellus. Quisque dignissim nibh quis sem imperdiet bibendum. Duis nec dictum turpis. Vestibulum auctor leo nulla, sed venenatis purus placerat et. Nam euismod mauris in justo semper laoreet.

Cras rutrum faucibus eros et feugiat. In sit amet viverra lorem. Sed euismod purus eget sodales ultricies. Praesent et blandit lorem. Pellentesque ut tempus velit. Aenean et vulputate arcu. Suspendisse finibus, tellus eu sollicitudin rutrum, augue nisi maximus lacus, ut blandit ante mi ut nulla.

Aenean metus dui, rhoncus sit amet pretium ut, laoreet quis ipsum. Sed non tempus turpis, ac eleifend lacus. Integer vel dui finibus, imperdiet neque porttitor, venenatis nunc. Vivamus egestas, turpis quis porttitor tincidunt, purus risus suscipit felis, nec consequat justo quam at purus. Pellentesque consequat sem ut nibh malesuada pulvinar. Quisque lobortis pulvinar urna, vitae luctus nulla ultricies vestibulum. Sed accumsan tortor tellus, in aliquam tellus sollicitudin eu. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Duis justo ipsum, vestibulum at magna id, ultrices lobortis turpis. Vivamus porttitor scelerisque odio sit amet posuere. Morbi fermentum ultricies sem, dictum pulvinar turpis.

Resumo

A quantidade de informações disponíveis cresce a cada ano. Aumenta, junto com o volume de dados e informações, o interesse em tratar, analisar e descobrir conhecimento a partir desta avalanche de dados. A mineração de dados juntamente com o aprendizado de máquina são duas ferramentas-chave dentro deste processo de descoberta de conhecimento e utilização de todos esses dados e informações para propósitos úteis. Entretanto, antes que os dados possam ser analisados, eles precisam ser armazenados e os sistemas computacionais sofrem, também de forma crescente, permanentes ameaças à sua segurança. Neste contexto se inserem as políticas para sistemas computacionais que buscam garantir meios para proteção, confidencialidade e confiabilidade dos acessos dos usuários aos objetos dentro dos sistemas de uma organização. Em sistemas com múltiplos sujeitos, muitas ações e diversos objetos, eventualmente, ocorrerão conflitos entre políticas. Um conflito ocorre quando os objetivos de duas ou mais políticas não podem ser atendidos simultaneamente em determinado contexto. Este trabalho tem como objetivo propor que o problema da detecção de conflitos em políticas pode ser convertido em um problema de *data mining* (mineração de dados) resolvido pela tarefa da classificação além de modelar e sintetizar uma forma de detectar estes conflitos mediante o uso de diferentes algoritmos e técnicas da aprendizagem de máquina que consigam acurácias elevadas e forneçam modelos genéricos o suficiente para serem usados em outros contextos.

Palavras-chave: Controle de Acesso. Mineração de dados. Aprendizagem de máquina. Conflitos diretos. Conflitos indiretos. Detecção de conflitos.

Abstract

The amount of information available grows every year. It increases, along with the volume of data and information, the interest in treating, analyzing and discovering knowledge from this avalanche of data. Data mining associated with machine learning are two key tools within this process of discovering knowledge and using all that data and information for useful purposes. However, before data can be analyzed, it needs to be stored and computer systems are also increasingly threatened with permanent security threats. In this context, policies for computer systems are inserted as a way to guarantee protection, confidentiality and reliability of users' access to objects within an organization's systems. In systems with multiple subjects, many actions and different objects, eventually, conflicts between policies will occur. A conflict occurs when the objectives of two or more policies cannot be met simultaneously in a given context. This work aims to propose that the problem of detecting conflicts in policies can be converted into a problem of data mining (data mining) solved by the task of classification, in addition to modeling and synthesizing a way of detecting these conflicts through the use of different machine learning algorithms and techniques that achieve high accuracy and provide generic models to be used in other contexts.

Keywords: Access control. Data mining. Machine learning. Direct conflicts. Indirect conflicts. Conflict detection.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AM	:	*Aprendizado de Máquina*;
ATM	:	*Air-Traffic Management*;
BPNN	:	*Basic Probabilistic Neural Network*;
CPNN	:	*Constructive Probabilistic Neural Network*;
DAC	:	*Discretionary Access Control*;
DARPA	:	*Defense Advanced Research Projects Agency*;
DoS	:	*Denial of Service*;
GPGU	:	*General Purpose Graphic Processor Unit*;
GPU	:	*Graphic Processor Unit*;
KDD	:	*Knowledege Discovery in Data Bases*;
MAC	:	*Mandatory Access Control*;
ML	:	*Machine Learning*;
MLP	:	*MultiLayer Perceptron*;
NFL	:	*No-Free Lucnch*;
PMC	:	*Perceptron MultiCamadas*;
RBAC	:	*Role-Based Access Control*;
ReBAC	:	*Relationship-Based Access Control*;
RNA	:	*Rede Neural Artificial*;
RTLS	:	*Real Time Location Systems*;
SMA	:	*Sistema Multi Agentes*;
SVM	:	*Support Vector Machiner*;

Sumário

Capítulo 1

Introdução

1.1 Contextualização

O volume de dados e informações cresce exponencialmente a cada ano [?], portanto, há uma frequente e ininterrupta demanda por mais infraestrutura de TI nas empresas, nos governos e mesmo nos usuários domésticos [?] e, mais ainda, por um correto tratamento, destino e interpretação à imensidão de dados gerados por pessoas, empresas e governos.

Em uma ampla variedade de campos, os dados estão sendo coletados e acumulados em um ritmo acelerado.[?][?] e há, assim, uma crescente demanda por análise adequada destes. Neste contexto se insere a mineração de dados com suas técnicas para tratamento e extração de conhecimento desse volume crescente de dados.[?][?]

Este trabalho usa diversas tarefas da mineração de dados para modelar uma hipótese que possibilite detectar conflitos em políticas de controle de acesso. Para isso, diversos algoritmos de classificação serão explorados, descritos e utilizados com ênfase nas redes neurais e outras técnicas lineares de classificação.

As políticas de proteção, confidencialidade e confiabilidade da informação, como as de controle de acesso, sendo parte da área de segurança computacional, são uma das formas de garantir, mediante o estabelecimento de regras, padrões e normas a salvaguarda e a disponibilidade das informações dos sistemas.[?][?]. Este tema, cf. [?, p.1] “é um tema de pesquisa importante dentro do contexto de segurança de sistemas, pois é um dos componentes fundamentais em qualquer sistema de computação."

Segundo [?], um aspecto muito relevante e muitas vezes tratado com pouca ênfase na construção de sistemas é a formulação, gerenciamento e manutenção de políticas de segurança da informação, principalmente as de controle de acesso.

Nas palavras de [?, p.1],

> A definição dessas políticas é normalmente orientada por modelos que fornecem um conjunto de regras e mecanismos para o funcionamento seguro de uma representação abstrata de sistemas. Porém, a administração de tais políticas frequentemente *se torna um processo complexo*, pois deve garantir que elas sejam eficientes e que não comprometam o *desempenho* dos sistemas. *[Grifo do autor.]*

Uma política, como as de controle de acesso, descrevem qual ação um sujeito (em um sistema) pode fazer (*permissão*), não pode fazer (*proibição*) ou é obrigado a fazer (*obrigação*) sobre um objeto em um dado contexto [?].

De maneira semelhante são conceituadas as *normas* e os conjuntos de normas usados para lidar com a autonomia e a diversidade de interesses entre os diferentes agentes em um sistema multiagentes como o descrito e estudado em [?]. Essas normas que regulam as ações dos agentes são análogas às definições das políticas citadas na epígrafe deste parágrafo e são, cf. [?] e [?] fatores importantes para garantir a eficácia dos sistemas.

Em contextos reais, porém, muitas vezes as políticas de segurança (e *normas*) apresentam conflitos entre si. Estes surgem quando, por exemplo, duas políticas regulando o mesmo comportamento de determinado objeto em um sistema estão ativas, mas uma delas obriga (ou permite) a realização de determinado comportamento ou ação enquanto a outra proíbe o mesmo. [?][?]. A detecção automatizada destes conflitos, com alta acurácia e custo computacional conveniente, é o problema de pesquisa deste trabalho. A descrição pormenorizada dos conflitos entre políticas encontra-se na seção 2.3.4 deste trabalho.

1.1.1 Problema

O problema investigado neste trabalho consiste na *detecção de conflitos de forma automatizada usando técnicas de mineração de dados e aprendizagem de máquina* que apresentem acurácias superiores a 95% e que, ao analisar simultaneamente várias políticas ao se inserir uma nova instância não leve a um custo computacional exponencial.

1.1.2 Justificativa

Na detecção de conflitos em políticas, geralmente, cf. revisão da literatura, usam-se abordagens semelhantes as de [?] baseadas em análise de ontologias entre os atributos que

compõem uma política e em regras de propagação das mesmas ou procedimentos como os descritos em [?] que utilizam lógica deôntica[1] para encontrar os conflitos.

Entretanto, para que os conflitos sejam detectados nessas abordagens as políticas são analisadas em pares (e sem filtros para agrupamentos) e mesmo quando são verificadas múltiplas *normas* ou políticas, como em [?], todas elas devem ser novamente "consultadas"ou "varridas"a cada nova instância de uma política inserida no sistema (para que o conflito seja ou não detectado). O trabalho de [?] provou que esta forma de analisar políticas é um problema NP-completo, ou seja, ainda não foi provado que esta classe de problemas pode ser resolvida em tempo *polinomial*, sendo assim, são computacionalmente onerosos a cada vez que uma instância nova de política é analisada (em tempo d execução sendo, normalmente exponencial). Com o crescimento orgânico, natural e temporal da quantidade de políticas em um sistema computacional este fato tende a tornar a manutenção e gerenciamento das políticas algo *computacionalmente custoso e dispendioso.*

Conforme se visualiza na seção 2.10, as técnicas e algoritmos de aprendizagem de máquina juntamente com as de mineração de dados foram utilizadas com resultados promissores na detecção de conflitos, principalmente em [?], [?] bem como em [?] e [?] que abordam problemas variados como detecção de colisões em voos, segurança de acesso computacional, incidentes em rodovias e intrusão de sistemas — todos de alguma forma relacionados à conflitos entre normas, regras, políticas ou direção.

Em grandes organizações as *políticas de segurança*, como as de controle de acesso, pela quantidade de objetos, modalidades, sujeitos e ações inerentes a essas instituições tendem a ter grande quantidade de informações e elas crescem conforme o uso diário e constante dos sistemas computacionais. organizações grandes, com múltiplos objetos e ações possuem centenas, às vezes, milhares de políticas (entre elas *políticas de controle de acesso*, por exemplo).[?].

Neste ambiente e considerando que as políticas analisadas em pares, como em [?], tem alto custo computacional por este, como visto em [?], ser um problema NP-completo e com a estratégia de minimizar a complexidade deste problema otimizando-o baseando suas soluções no conhecimento adquirido e já existente nas organizações (os *datasets* de políticas), a mineração de dados com técnicas de aprendizagem de máquina surgem como possibilidades de solução na detecção de conflitos tanto em tempo de design quanto em tempo de execução, pois, minimiza este custo computacional ao se aproveitar da

[1] A lógica *deôntica* é um tipo de lógica usada para analisar de modo formal as normas e as proposições que tratam dessas normas [?]

"história"temporal das políticas da organização, mediante o conhecimento adquirido e "treinado"pelos algoritmos de aprendizagem de máquina.

1.1.3 Hipótese

Diante do contexto apresentado na seção anterior e também por [**?**] tem-se *como hipótese deste trabalho*, que o problema de detectar conflitos entre políticas *pode ser convertido e transformado* em uma tarefa de classificação da mineração de dados e que o uso de algoritmos de aprendizagem de máquina associados a técnicas de *data mining* para detectar estes conflitos configure um método que apresente precisão e acurácia superiores a 95%.

1.2 Objetivos

1.2.1 Objetivo geral

O objetivo deste trabalho é propor que o problema da detecção de conflitos em políticas pode ser convertido em um problema de *data mining* (mineração de dados) resolvido pela tarefa da classificação além de modelar e sumariar uma forma de detectar estes conflitos mediante o uso de diferentes algoritmos e técnicas da aprendizagem de máquina que consigam acurácias elevadas.

1.2.2 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Estabelecer a relação entre machine learning, técnicas de mineração de dados e o problema do conflito entre políticas;
- Determinar e comparar quais algoritmos e técnicas são mais adequados para cada tipo de conflito nas políticas (ou normas) usando as suas acurácias;
- Usar e comparar o desempenho, a precisão e taxa de acertos das principais técnicas usadas no aprendizado de máquina: redes neurais artificiais (RNA), Support Vector Machines (SVM) e redes neurais recorrentes e profundas.
- Transformar o problema da detecção de conflitos em uma tarefa de classificação da mineração de dados mantendo acurácias elevadas;

- Usar *frameworks* de aprendizado de máquina como TensorFlow, Keras, ou Torch, na construção, treinamento e teste de redes neurais, comparando-os, quando adequado, para o problema específico deste trabalho;
- Estabelecer a detecção de conflitos em políticas (ou normas) como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

1.3 Solução Propostas

Diante da hipótese apresentada na seção 1.1.3, a solução para o problema apresentado neste trabalho na seção 1.1.1 concentra-se, prioritariamente, em mostrar que *converter* (ou *transformar*) a detecção de conflitos a um *problema de classificação* da mineração de dados associado a técnicas de aprendizagem de máquina, reestruturando os atributos do *dataset*, se necessário, se configura um método com acurácia superior a 95%.

A primeira solução proposta para conflitos diretos entre políticas é usar as técnicas e algoritmos de classificação (aprendizado supervisionado) para realizar a detecção automatizada de conflitos. Para isso, propõem-se:

- Usar, inicialmente, uma rede neural (um perceptron de uma camada ou com somente uma camada oculta e apenas com *forward*) como técnica algorítmica para a detecção de conflitos e
- Construir a arquitetura de uma rede neural multicamadas (com camadas ocultas), e retropropagação (*backpropagation*), comparando-a com outro classificador linear, como, por exemplo, o SVM, para estabelecer qual técnica de mineração de dados na detecção de conflitos em políticas é mais precisa.

Realizado os múltiplos experimentos, atestar a hipótese mediante os resultados apresentados.

1.4 Método de Pesquisa

O método de pesquisa relatando todos os passos necessários para demonstrar os objetivos descritos na seção 1.2 e de que forma eles foram atingidos serão pormenorizadamente detalhados na seção 3 deste trabalho.

1.5 Resultados Esperados

Ao fim deste trabalho os seguintes resultados são esperados:

- Mostrar que o problema da detecção de conflitos em políticas pode ser convertido em um problema de *data mining* (mineração de dados) resolvido pela tarefa da classificação;
- Demonstrar que o problema da detecção de conflitos é um prolema linearmente separável;
- Mostrar que a política nova (*instância inédita*) é ou não conflitante imediatamente após a criação da mesma usando como base o treinamento da rede neural no *dataset* de políticas existente;
- Modelar e resumir uma forma de detectar estes conflitos mediante o uso de diferentes algoritmos e técnicas da aprendizagem de máquina que consigam acurácias superiores a 95%;
- Estabelecer a relação entre machine learning, técnicas de mineração de dados e o problema do conflito entre políticas;
- Determinar e comparar quais algoritmos e técnicas são mais adequados para cada tipo de conflito nas políticas (ou normas) usando as suas acurácias;
- Usar e comparar o desempenho, a precisão e taxa de acertos das principais técnicas usadas no aprendizado de máquina: redes neurais artificiais (RNA), Support Vector Machines (SVM); redes neurais recorrentes e profundas;
- Estabelecer a detecção de conflitos em políticas (ou normas) como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

1.6 Limitações do Trabalho

Não faz parte do escopo deste trabalho:

- Delinear um modelo de política com objetivos semânticos diferenciados. Para os experimentos deste trabalho será usado o modelo de políticas descrito em [?] e em [?]

- Analisar comparativamente os modelos de extensão de políticas em um determinado contexto;
- Usar redes neurais convolucionais profundas na detecção dos conflitos;
- Abordar a semântica em políticas;

1.7 Organização do trabalho

Este trabalho está organizado da seguinte forma:

- Neste capítulo 1 estão dispostas a contextualização, o problema da pesquisa, a justificativa, a hipótese de pesquisa, os objetivos, as soluções que foram propostas, os resultados esperados e as limitações do trabalho
- No Capítulo 2 apresenta-se todo o referencial teórico compreendendo uma revisão bibliográfica sobre os principais temas desta dissertação, como políticas, detecção de conflitos, mineração de dados e aprendizagem de máquina (e seus algoritmos principais)
- No Capítulo 3 são mostrados o método e os experimentos e resultados obtidos.
- No Capítulo 4 mostra-se o cronograma para a finalização da dissertação
- No Capítulo 5 apresenta-se as conclusões e propostas para trabalhos futuros relacionados a esta pesquisa.

Capítulo 2

Referencial teórico

Nesta seção, alguns conceitos fundamentais da pesquisa bibliográfica realizada serão explanados com o intuito de atingir os objetivos descritos na seção 1.2 e o entendimento da solução proposta neste trabalho. Serão abordados modelos de políticas, como as de controle de acesso e conflitos entre as mesmas. Temas como mineração de dados, aprendizagem de máquina, algoritmos de classificação, com destaque para as *Redes Neurais Artificiais* e **SVM** — *Support Vector Machines*, que são a base da hipótese deste trabalho também serão descritos e suas definições teóricas serão discutidas. Ao final, serão descritos alguns trabalhos relacionados ao tema desta dissertação.

2.1 Segurança da Informação

Vivemos a era da informação onde o volume de dados e informações disponíveis cresce vertiginosamente a cada ano [**?**]. Este fato leva a uma acelerada e incessante procura por infraestrutura de TI nas organizações, nos governos e usuários domésticos [**?**]. Em praticamente todos os campos e áreas da humanidade, os dados estão sendo coletados e acumulados em ritmo crescente. [**?**] [**?**].

Considerando, por exemplo, o ambiente corporativo, segundo [**?**, p.1], a "informação é um recurso essencial para toda organização, independentemente do seu porte e do seu segmento de atuação no mercado".

E prossegue afirmando que:

> É utilizando a informação que processos organizacionais funcionam, as pessoas podem realizar suas atividades profissionais, a geração de conhecimento acontece e o compartilhamento desse conhecimento é realizado. [**?**, p.1]

Neste contexto, tanto a obtenção da informação é um processo importante quanto o são as formas de armazenamento e proteção, além, claro, o fato de que a análise e interpretação destes dados e informações tornam-se atividades essenciais para a manutenção de negócios e o próprio desenvolvimento da sociedade. [?] [?].

A informação, como observa-se, é um recurso crítico para qualquer instituição exigindo, portanto, a adoção de políticas de segurança adequadas que visem a sua proteção, salvaguarda e manutenção para que este ativo, tão significativo, mantenha seu valor, sua abrangência e importância dentro do cenário das organizações, governos e usuários comuns [?]. Isto porque assim como a quantidade de dados e informações cresce exponencialmente [?], aumenta também, a cada ano, as ameaças aos ativos de informação variando desde fraudes informáticas a sabotagens, vandalismo ou espionagem [?].

Nos primórdios da computação, os sistemas de informação e aplicações eram basicamente militares e não ligados em redes, logo, os problemas de segurança se limitavam ao acesso físico aos equipamentos [?]. Os primeiros problemas começaram a surgir com a necessidade do compartilhamento do processamento de informações e de recursos entre várias categorias de usuários, com diferentes níveis de acesso, o que levou ao desenvolvimento dos sistemas operacionais de tempo compartilhado (*time-sharing*).

O foco era, prioritariamente, garantir a segurança já que recursos como terminais de acesso, dispositivos de armazenamento, impressoras, programas e sistemas eram compartilhados e poderiam gerar problemas de segurança [?]. Assim, originou-se o problema clássico da segurança: "como fazer com que usuários autorizados possam ter acesso a determinadas informações, ao mesmo tempo em que os usuários não autorizados não possam acessá-las?"[?]

Destarte, foi criado, em 1967, nos EUA, com a contribuição de órgãos governamentais como o Departamento de Defesa e a CIA um documento chamado "*Security Control for Computer System:Report of Defense Science Board*" que marcou o início do processo oficial de criação de um conjunto de regras para a segurança de computadores.

Outras ações se seguiram ao redor do mundo. Em outra atitude, o Departamento de Defesa norte-americano formulou um plano que daria origem ao *"The Orange Book"*, o documento que contém um conjunto abrangente de regras de segurança. Embora hoje seja considerado ultrapassado, teve grande importância e foi base para diversas iniciativas relativas à segurança da informação. [?]

Era uma época em que o mundo ainda não estava totalmente interconectado por redes

de computadores, em especial, pela Internet que trouxe, além de inúmeros benefícios, diversos outros problemas de segurança da informação e representou um desafio novo à proteção dos dados das organizações, governos e usuários domésticos. [?].

Assim, segundo [?], "a segurança da informação é uma questão de séria preocupação global. A complexidade, acessibilidade e abertura da Internet serviram para aumentar tremendamente o risco à segurança dos sistemas de informação".

[?, p. 2] afirma que:

> Informações que antes estavam escritas em um relatório e poderiam ser protegidas quando guardadas em uma gaveta com chave hoje [...] podem ficar disponíveis pela Internet para milhões de usuários em todo o mundo. Uma pequena falha ou uma ação criminosa pode disponibilizar informações confidenciais ou bloquear o acesso a informações críticas para a realização do objetivos da organização.

Na mesma linha, [?] discorre que:

> as organizações possuem diversos activos de informação que devem ser protegidos de ameaças de variados tipos, através de estratégias e políticas de segurança da informação, implementadas com base em modelos e métricas bem definidas, permitindo às organizações melhorar o seu processo de responsabilização da segurança da informação.

De acordo com [?], as maiores ameaças à segurança da informação são, portanto:

> ataques deliberados a software; falhas técnicas ou erros de software; ato humano de falha ou erro; atos deliberados de espionagem ou transgressão; atos deliberados de sabotagem ou vandalismo; falhas técnicas ou erros de hardware; atos deliberados de furto/roubo; forças da natureza; comprometimento de propriedade intelectual; problemas de qualidade em provedores de serviços; obsolescência tecnológica; e atos deliberados de extorsão de informações.

Diante destas ameaças, sejam elas previsíveis ou extemporâneas, insere-se toda a área da segurança da informação que para cada tipo de ameaça tem diversas opções de soluções para diferentes tipos de organizações. [?]

Segundo [?, p. 2]:

> O que diferencia o uso da informação entre as organizações é a necessidade de disponibilidade, a exigência de integridade e o rigor em relação ao sigilo que cada organização precisa para a sua informação.

E continua, declarando que:

> O grau de disponibilidade, integridade e confidencialidade (sigilo) protegerá a informação para que a organização operacionalize os seus negócios e atenda aos seus objetivos. [?, p. 2]

Portanto, cf. [?], como as informações são um ativo corporativo crítico que se tornou cada vez mais vulnerável a ataques de vírus, hackers, criminosos e erros humanos, as organizações precisam priorizar a segurança de seus sistemas para garantir que seus ativos de informação mantenham sua precisão, confidencialidade e disponibilidade. Na mesma linha, [?, p. 4], afirma que "as organizações precisam implantar um processo de segurança da informação, e este processo deve ser considerado um ativo da organização, como tantos outros".

[?] declara ainda que um mecanismo cada vez mais importante para proteger as informações corporativas e, ao fazer isso, reduzir a ocorrência de violações de segurança, é a formulação e aplicação de uma *política* formal de segurança da informação nas organizações. [grifo do autor]

2.2 Políticas em sistemas de segurança

Para proteger os sistemas de informação dos níveis crescentes de ameaças cibernéticas, as organizações são praticamente obrigadas a instituir programas (e controles) de segurança. Neste contexto, as *políticas de segurança da informação* são uma base imprescindível dos programas de segurança organizacional. [?] [?]

[?, p. 402] consideram que "o mais importante desses controles [e programas de segurança] é a política de segurança da informação". E [?], afirmam que o desenvolvimento de uma política de segurança da informação é o primeiro passo para preparar uma organização contra ataques de fontes internas e externas.

Para [?], a política de segurança da informação trata da integridade, disponibilidade e confidencialidade dos dados eletrônicos mantidos e transmitidos entre os sistemas de informação e é a condição prévia para a implementação de outros obstáculos eficazes.

Uma política é uma regra geral que foi estabelecida em uma organização para limitar a circunspecção dos subordinados a ela. [?]. Pode também ser entendida como regras que governam as escolhas no comportamento de um sistema. [?] [?]

No domínio dos sistemas de informação, a política foi definida em um contexto de planejamento e controle para estabelecer limites de comportamento aceitável, conflitos de decisão e padrões. [?]. As políticas são especialmente relevantes para a segurança dos sistemas de informação, pois fornecem as bases para um programa geral de segurança e criam uma plataforma para implementar práticas seguras em uma organização. [?]. Em todos os trabalhos citados nesta seção seus autores referem-se às políticas de segurança de alto nível, em âmbito regulatório e completo. [?].

2.3 Controle de acesso

Para [?], o objetivo de uma política é fornecer orientação gerencial e suporte à segurança da informação de acordo com os requisitos de negócios e as leis e regulamentos relevantes em determinada organização ou instituição. A política de segurança em um sistema computacional garante, portanto, a proteção de suas informações. Dentre as diversas tecnologias utilizadas para assegurar essas propriedades, temos, por exemplo, o controle de acesso. [?]

O controle de acesso é o mecanismo central para atingir os requisitos de segurança em sistemas de informação. [?]. Dessa forma, trata-se de uma tecnologia indispensável para quem faz uso de qualquer tipo de sistema, podendo basear-se ou coexistir com outros serviços de segurança. [?].

[?, p. 8] afirma que "para entender o que é e como funcionam os mecanismos de controle de acesso de um sistema [...], faz-se necessário classificar os elementos do sistema em três grupos." Que são, segundo este autor, o conjunto de *sujeitos*, de *objetos* e o de *ações*.

De acordo com [?, p. 8], o *conjunto de sujeitos* "englobam qualquer elemento que pode realizar ações sobre objetos, e é composto por usuários, processos e o próprio sistema". Já o *conjunto de objetos* "englobam os elementos sobre os quais podem ser realizadas ações". E, por sua vez, o *conjunto de ações* "compreende a lista de ações que podem ser realizadas sobre cada um dos objetos do sistema". Como, por exemplo, ler, escrever, apagar, abrir conexão, enviar mensagem, encerrar conexão entre muitas outras ações relativas a cada sistema em particular.

Os mecanismos de controle de acesso, portanto, verificam todas as requisições de dados e recursos administrados pelos sistemas definindo, assim, as circunstâncias nas quais as requisições de acesso são permitidas ou negadas. [?]. Desta forma, tanto as informações quanto os ativos do sistema mantém um nível adequado de segurança, conservam o sigilo e estabelecem importantes providências quanto ao acesso a informações ou recursos confidenciais ou protegidos. [?]

Com o controle de acesso o usuário fica limitado apenas a execução de operações e ações no sistema que lhe foram previamente concedidas. Ao requerer acesso ao conteúdo de informações ou permissão de uso de recursos estes só serão outorgados a quem possui o direito de acesso aos mesmos. [?]

Os modelos de controle de acesso fornecem um conjunto de regras e mecanismos para o funcionamento seguro dos sistemas, sendo responsáveis pela definição de políticas específicas de controle de acesso. As políticas são diretrizes de *alto nível* [?] que determinam como os acessos são controlados e decisões de acessos são estabelecidas. [?] [?] [?]

Embora não seja o escopo deste trabalho é importante salientar que existem vários modelos, na literatura, de controle de acesso. Entre os principais, pode-se citar, o controle de acesso discricionário (*Discretionary Acces Control* - DAC); o controle de acesso mandatório (*Mandatory Access Control* - MAC); o controle e acesso baseado em papéis (*Role-Based Access Control* - RBAC) e o controle de acesso baseado em relacionamentos (*Relationship-Based Access Control* - ReBAC). Sobre o DAC e o MAC, sugere-se os trabalhos de [?] e [?]. Sobre RBAC, recomenda-se [?] e [?]. Sobre o ReBAC aconselha-se seguir o trabalho de [?] e [?].

2.3.1 Políticas de Controle de acesso

Uma política de controle de acesso tem como objetivo definir ou limitar o comportamento atual ou futuro de *sujeitos* e *objetos* para garantir que as suas *ações* estejam alinhadas com os objetivos da empresa de acordo com o escopo de acesso de cada sujeito ou grupo de sujeitos, com as permissões, proibições ou obrigações que estes tenham sobre os objetos e quais dados ou recursos lhe são concedidos. [?][?]

As políticas de controle de acesso convencionais, inicialmente foram chamadas de *autorizações* e tinham a seguinte forma: `{sujeito, objeto, ação}`. Estas *autorizações* especificavam quais operações os *sujeitos* podiam executar sobre os *objetos* no contexto de um sistema. [?][?]

Com o desenvolvimento dos sistemas e das políticas, estas últimas passaram a ser direcionadas, principalmente, na especificação e administração de requisitos de controle de acesso expressos na forma de *proibições*, *permissões* e, posteriormente de *obrigações* que são as composições principais na aplicação destas políticas. [?]

2.3.2 Modelos de políticas

Os autores [?] afirmam que um modelo de política deve ter os seguintes atributos fundamentais: `{modalidade, sujeito, objeto e ação}`. A particularidade da política compreende estabelecer uma *autorização*, uma *permissão* ou *proibição*.

O *sujeito* da política é a quem ela é orientada. O *objeto* define o conjunto de objetos no qual a política está focada. A *ação* é estabelecida como procedimentos que podem ser efetuados em *objetos* no sistema [?]. Outros trabalhos da literatura também consideram estes atributos apresentados aqui, com as mesmas conotações, para a definição de uma política. [?]

Outros modelos de políticas são fartamente descritos em [?], também em [?] e [?] além de [?] e [?]

2.3.3 Modelo de Política utilizado

De acordo com [?]:

> Definir uma política de controle de acesso não é uma tarefa simples, principalmente porque algumas vezes é necessário representar formalmente políticas complexas, tais como as que tem origem em práticas de leis e regulamentos organizacionais.

Desta forma, a definição da política deve combinar todos os diferentes regulamentos para ser executada e considerar todas as possíveis ameaças adicionais relativas ao uso de sistemas. [?]

O modelo de política utilizado neste trabalho baseia-se inteiramente no modelo proposto por [?] e [?] baseado nos modelos propostos pelos autores [?], e também o trabalho de [?], bem como o artigo de [?].

Portanto, de acordo com [?, p.36] e [?]:

Uma política é uma tupla da forma:

$$Policy = KP \times Org \times SR \times AA \times OV \times Ac \times Dc \tag{2.1}$$

Onde KP descreve o tipo de política (uma proibição (F), da palavra em inglês Forbidden; uma permissão (P); ou uma obrigação (O)). $Org.$ relata o local (ambiente) onde a política deve ser cumprida, isto é, a organização na qual os sujeitos devem cumprir a política. SR descreve a quem (entidades) se destina a política (pode ser um sujeito $s \in S$ ou um papel $r \in R$, ou seja, $SR = S \cup R$. Um sujeito pode ser um usuário $u \in U$ ou uma organização $org \in Org$ representando o grupo de sujeitos que devem cumprir com a política, isto é, $S = U \cup Org$). AA identifica uma ação $a \in A$ ou uma atividade $act \in Act$ (uma atividade é a união de várias ações relacionadas). OV relata um objeto $o \in O$ ou uma visão $v \in V$ que está sendo manipulada pela ação/atividade (uma visão é a união de vários objetos). Ac é a condição de ativação da política e Dc é a condição de desativação da política. Uma condição constitui a configuração para um evento, em termos de que a política deva seguir.

Em [?] definiu-se Ac e Dc como datas, desta forma Ac é a data de ativação da política e Dc é a data de desativação.

A figura 2.1 exemplifica o modelo de políticas utilizadas neste estudo para a mineração de dados e aprendizagem de máquina.

Figura 2.1: Modelo das políticas utilizadas no estudo

```
Policie10-> [Permitted, Administrative_Unit, PROTOCOLIZADOR3, Close, ProcessDispatch, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie11-> [Permitted, Administrative_Unit, PROTOCOLIZADOR3, Create, ProcessDispatch, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie12-> [Permitted, Administrative_Unit, PROTOCOLIZADOR3, Record, ProcessDispatch, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]

Policie25-> [Forbidden, Institution, null, Record, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie26-> [Forbidden, Institution, null, Generate, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie27-> [Forbidden, Institution, PROTOCOLIZADOR3, Move, ProcNURCADesp, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie28-> [Forbidden, Institution, PROTOCOLIZADOR3, Acess, ProcNURCADesp, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie29-> [Forbidden, Institution, PROTOCOLIZADOR3, Record, ProcNURCADesp, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]

Policie43-> [Obliged, Administrative_Unit, null, Open, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie44-> [Obliged, Administrative_Unit, null, Acess, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie45-> [Obliged, Administrative_Unit, PROTOCOLIZADOR, Open, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie46-> [Obliged, Administrative_Unit, PROTOCOLIZADOR, Acess, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie47-> [Obliged, Administrative_Unit, PROTOCOLIZADOR3, Open, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie48-> [Obliged, Administrative_Unit, PROTOCOLIZADOR3, Acess, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
Policie49-> [Obliged, Administrative_Unit, MARY, Open, Process, Sun Aug 18 20:06:14 BRT 15, Tue Dec 31 20:06:14 BRT 15]
```

Fonte: compilação do autor

2.3.4 Detecção de conflitos

Segundo [?], quando um modelo de controle de acesso inclui a possibilidade de especificar permissões, proibições e obrigações podem ocorrer alguns conflitos entre políticas.

Os conflitos podem ocorrer quando diferentes conjuntos de condições resultam em permitir e negar simultaneamente, ao mesmo papel, à mesma solicitação, ou proibir e obrigar o mesmo papel, à mesma solicitação, ou seja, quando os objetivos de duas ou mais políticas não podem ser atendidos simultaneamente. [?]

2.3.4.1 Classificação dos conflitos

De acordo com [?], [?] e [?], os conflitos classificam-se, principalmente em:

- *conflito de modalidade* que decorre de políticas de propriedades contrárias;
- *conflito em potencial* surge da tripla sobreposição de sujeitos, ações e objetos de políticas de predicados opostos;
- *conflito de redundância* que surge da precedência de execução dadas a certas políticas e
- *conflito específico de aplicação* que ocorre quando ações antagônicas são atribuídas para a mesma entidade (sujeito ou papel) mediante controles externos específicos, expressos como metapolíticas que agem como contenções para políticas permitidas.

Neste trabalho, para o modelo de política proposto, não abordam-se os conflitos de redundância já que ordens de precedência não são tratadas. Todas as outras modalidades de conflitos são examinadas e utilizadas.

Assim como em [?, p. 24], neste trabalho, os conflitos em potencial são denominados **conflitos diretos** e os conflitos de aplicação são abarcados pelos chamados **conflitos indiretos**.

2.3.4.2 Conflitos Diretos e Indiretos

[?] usa uma abordagem para os conflitos diretos e simples que será replicada neste trabalho. Diz-se que duas regras estão em conflito quando o cumprimento de uma das regras viola a outra e vice-versa. Ou seja, a verificação se há o conflito é feita entre duas políticas que possuem modalidades contraditórias ou antagônicas, definidas na mesma organização, executadas pelos mesmos sujeitos, efetuando a mesma ação em relação a um objeto específico.

Exemplo de um ***conflito direto***:

{P1= Permitido, na Universidade X, Ana Ester, acessar processos administrativos}

{P2= Proibido, na Universidade X, Ana Ester, acessar processos administrativos}

Os exemplos acima mostram que quando uma política proíbe e a outra permite um sujeito de realizar uma ação estabelecida sobre um objeto específico em uma organização particular ocorre um conflito direto. O conflito, segundo [?], pode ser identificado diretamente utilizando a sobreposição dos atributos das políticas.

Já em um conflito indireto, as políticas conflitantes regulam ações diferentes (mas relacionadas) executadas por distintos sujeitos (porém, relacionados) sobre objetos desiguais (mas, relacionados) em organizações diferentes (mas, relacionadas). [?, p.24]

Além disso, um conflito indireto pode ainda ocorrer, mesmo quando as políticas em conflito não têm modalidades contraditórias ou contrárias.

Ex:

P3 = Obrigado, Empresa E, Funcionário, receber, avaliação, mensal

P4= Permitido, Empresa E, Analista, conceder, avaliação, mensal

Este conflito não seria detectado diretamente, porém há um conflito se considerarmos os relacionamentos.

A capacidade de um sistema reconhecer um estado inconsistente em andamento ou em potencial é denominada **detecção de conflitos.**

Para [?],

> detectar conflitos entre políticas de controle de acesso é o primeiro passo para buscar inibir o surgimento de erros no sistema relativo às políticas aplicadas, tendo em vista que as políticas sem conflitos refletem corretamente o plano de segurança do sistema.

Segundo [?, p.25], "Para um sistema baseado em políticas trabalhar de forma eficaz é importante ter um meio de detectar e resolver os conflitos que possam surgir". Torna-se, assim, indispensável a utilização de abordagens que analisem previamente a existência de conflitos como o proposto nesta dissertação.

2.4 Mineração de Dados

Uma das características de nossa era é produção de dados em grande volume, velocidade e variedade de todas as formas, por dispositivos espalhados em toda parte. Entretanto, dados, mesmo em grande quantidade, são apenas dados. É preciso produzir informação e conhecimento para explorar as vantagens que essa massa pode trazer. O dado necessita ser, de alguma forma, analisado, tratado para que informações e conhecimento possam ser, deles, extraídos. [?] [?]

Conforme [?]:

> [...] Os computadores permitiram que os humanos coletassem mais dados do que podemos digerir, é natural [,portanto,] recorrer a técnicas computacionais para nos ajudar a desenterrar padrões e estruturas significativas a partir dos numerosos volumes de dados. Por isso, [a mineração de dados] é uma tentativa de resolver um problema que a era da informação digital transformou em realidade para todos nós: sobrecarga de dados.

Para [?],

> De forma simplificada, a *mineração de dados* pode ser definida como um processo automático ou semiautomático de explorar analiticamente grandes bases de dados, com a finalidade de descobrir padrões relevantes que ocorrem nos dados e que sejam importantes para embasar a assimilação de informação importante, suportando a geração de conhecimento.

Ainda, segundo [?],

> O termo mineração de dados tem sido usado principalmente por estatísticos, analistas de dados e comunidades de sistemas de informações gerenciais (MIS). Ele também ganhou popularidade no campo do banco de dados. O termo descoberta de conhecimento em bancos de dados [(KDD, da sigla em Inglês)] foi cunhada no primeiro workshop do KDD em 1989 para enfatizar que o conhecimento é o produto final de uma descoberta baseada em dados. Foi popularizado nos campos de IA e aprendizado de máquina.

2.4.1 KDD - Knowledge Discovery in Databases

Assim, a mineração de dados é parte integrante de um processo mais amplo, conhecido como descoberta de conhecimento em bases de dados (*Knowledge Discovery in Databases*, ou *KDD*)[?]. Embora se use *mineração de dados* como sinônimo de KDD, a terminologia é empregada para a etapa de *descoberta* do processo de KDD, que inclui a *seleção* e *integração* das bases de dados, a *limpeza* da base, a *seleção e transformação* dos dados, a *mineração*(propriamente) e a *avaliação* dos dados. [?][?].

Assim, a mineração de dados é definida em termos de esforços para a descoberta de padrões em bases de dados. A partir destes padrões descobertos, há condições de se gerar conhecimento útil para um processo de tomada de decisão (ou a geração de conhecimento para esta tomada).

Mais especificamente, KDD (*Knowledge Discovery in Database*) é um processo de busca de conhecimento em bancos de dados e, de modo geral, consiste de uma sequência iterativa de passos (ou **etapas**)[1]: limpeza de dados; integração dos dados; seleção, transformação e mineração dos dados; avaliação dos padrões e apresentação e assimilação do conhecimento. Este processo é iterativo e, em alguma etapa, pode-se voltar para uma anterior. [?]

A figura 2.2 mostra o funcionamento iterativo do processo de KDD (*Knowledge Discovery in Database*) - Descoberta de Conhecimento em Bases de Dados.

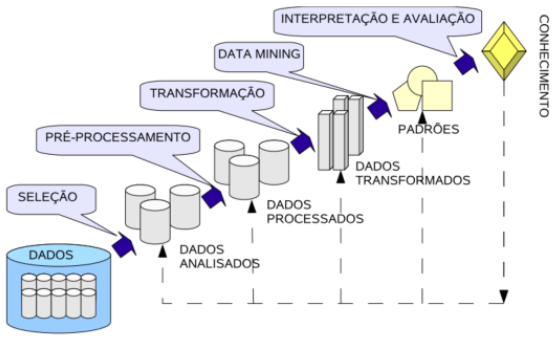


Figura 2.2: Etapas do processo de descoberta do conhecimento em bases de dados - KDD

Fonte: [?, p. 7]

Neste trabalho as tarefas de seleção e transformação dos dados farão parte da etapa chamada de pré-processamento de acordo com, [?] e serão descritas com maior riqueza de

[1] O processo de KDD, segundo [?] é composto por: *Seleção de dados; Pré-processamento; Transformação; Mineração; Análise e assimilação de resultados*

detalhes no capítulo 3.

2.4.2 Modelo de conhecimento

O termo **modelo de conhecimento** (ou hipótese) é utilizado na literatura (e neste trabalho) para fazer referência a um padrão ou conjunto de padrões descobertos (que é, enfim, o *propósito* do processo de KDD). Estes padrões são conhecimentos representados segundo as normas sintáticas de alguma linguagem formal. Estes padrões podem ser classificados em dois tipos: preditivos e descritivos. O intuito dos preditivos é resolver um problema específico de prever os resultados ou valores de um ou mais atributos, em função dos valores de outros atributos. Os descritivos (ou informativos) tem o intuito de apresentar informações interessantes e importantes sobre os dados que um especialista de domínio possa não conhecer. Modelos de conhecimento compostos exclusivamente por padrões preditivos são chamados de modelos preditivos, enquanto que modelos descritivos são modelos de conhecimento compostos por padrões descritivos. [?][?][?]

Para [?],

> Um modelo de mineração é criado aplicando-se um algoritmo a dados, mas é mais que um algoritmo ou um contêiner de metadados: é um conjunto de dados, estatísticas e padrões que podem ser aplicados a novos dados para gerar previsões e fazer inferências sobre relações.

Neste contexto, este trabalho se concentra, portanto, em criar modelos de forma a detectar, mediante o uso de técnicas da mineração de dados (e aprendizagem de máquina) os conflitos entre as políticas de controle de acesso de um sistema. Diversos modelos serão desenvolvidos e confrontados usando-se vários algoritmos e técnicas analisados com métricas específicas.

2.4.2.1 Arquitetura do modelo

Segundo [?], "Um modelo de mineração obtém dados de uma estrutura de mineração e analisa esses dados usando um algoritmo de mineração de dados". Entretanto, é importante diferenciar a estrutura e o modelo de mineração. Ainda de acordo com [?],

> A estrutura de mineração armazena informações que definem a fonte de dados. Um modelo de mineração armazena informações derivadas do processamento estatístico de dados, como os padrões encontrados em decorrência da análise.

Assim, o modelo fica "limpo" até que os dados que foram guarnecidos pela estrutura de mineração sejam processados e avaliados. Depois de produzido o modelo contém *metadados*, resultados e associações e pode, então, ser utilizado para a obtenção de conhecimento.

A arquitetura pode conter também, variáveis, hiperparâmetros, definições do modelo, filtros utilizados e, claro, o algoritmo utilizado na tarefa de análise dos dados. [?].

2.5 Aprendizagem de máquina

Segundo [?, p. 10],

> um dos passos do processo de KDD, o de extração de padrões (ou Mineração de Dados) utiliza métodos de Aprendizado de Máquina (AM) [ou *Machine Learning* - ML] para encontrar regularidades, padrões ou conceitos em conjuntos de dados.

A principal diferença, segundo os autores, [?] entre Aprendizagem de Máquina e KDD reside no fato de "grande parte da literatura em AM se concentra apenas no mecanismo de descoberta de padrões e/ou conceitos, sem se preocupar com o grau de utilidade". Já em KDD, ainda segundo [?], "os padrões extraídos são avaliados para aferir sua utilidade para o usuário em relação à tomada de decisão". Ou seja, o aprendizado na Aprendizagem de Máquina é um atividade-fim enquanto o aprendizado no KDD é uma atividade-meio para a obtenção do conhecimento.

2.5.1 Definição

Aprendizado de máquina ou *machine learning* é um braço da Inteligência Artificial que emprega técnicas e algoritmos na criação de modelos computacionais dos quais a característica principal é a capacidade de descobrir padrões em um grande volume de dados ou de melhorar o desempenho de uma determinada tarefa através da experiência (do *reforço*).[?] [?] [?]

Então, de acordo com [?, p. 278], "os padrões aprendidos, que podem ser bem complexos, são então usados para fazer predições relativas a dados ainda não vistos e novos".

Nas palavras de Arthur Lee Samuel, considerado um dos pioneiros na área de inteligência artificial [?], aprendizado de máquina é "o campo de estudo que dá aos computadores a capacidade de aprender sem serem explicitamente programado". [?, p. 89]. Aprendizado

de máquina tem sido aplicado na automatização de funções que para os humanos são executadas intuitivamente, mas que são difíceis de definir formalmente. [?]

Assim, de forma geral, a aprendizagem de máquina tem por objetivo estudar e desenvolver métodos computacionais para obter sistemas capazes de adquirir conhecimento de forma automatizada. [?].

A capacidade de determinados algoritmos tem de aprender a partir de exemplos é chamado de **aprendizado indutivo**. Estes algoritmos aprendem relacionamentos eventualmente existentes entre os dados, mostrando o resultado nos modelos de conhecimento gerado. [?][?]

Conforme [?, p. 279], os algoritmos de aprendizado de máquina "são fundamentalmente dependentes de uma fase de aprendizado, a qual é usada para produzir um modelo ou uma função que codifica padrões presentes nos dados de entrada".

Então, dependendo de qual é a abordagem de aprendizado usada, os algoritmos de aprendizado de máquina podem ser, basicamente de 3 principais tipos, que são: a aprendizagem supervisionada, a aprendizagem não-supervisionada e a aprendizagem por reforço [2]. [?] [?].

Na **aprendizagem não-supervisionada** o modelo/hipótese busca padrões na entrada, embora não seja fornecido nenhum *feedback* explícito. Portanto, na abordagem não-supervisionada não há, nos dados, uma classe, não há um rótulo prévio, ou seja, não existe a informação da saída desejada. O processo de aprendizado busca identificar regularidades entre os dados e não é necessária a divisão prévia dos dados em dados de treinamento, validação e teste. A tarefa mais comum de aprendizagem não supervisionada é o agrupamento. mas, os algoritmos de aprendizagem não-supervisionada incluem ainda, modelos de redes neurais, análise de componentes independentes e o já citado *clustering* (agrupamento) [?] [?] [?] [?]

Na **aprendizagem supervisionada** o modelo/hipótese observa alguns exemplos de pares de entrada e saída, e aprende uma função (ou modelo) que faz o mapeamento entre a entrada e a saída. Portanto, ela compreende a abstração de um modelo a partir dos dados apresentados na forma de pares ordenados (*entrada, saída, saída desejada*). Há, assim, uma *classe*, ou um atributo especial com o qual se pode comparar e validar o resultado. Esta categoria de aprendizagem de máquina requer uma função de aprendizado dos dados de treinamento fornecidos como entrada. Esses dados, então, são usados para para

[2] Há ainda os tipos de aprendizado *semissupervisionado* e a *transdução* (ou inferência transdutiva) que não serão discutidos neste trabalho

aprender uma função de classificação que pode, assim, ser usada para realizar predições de classes para dados ainda não vistos ou novos. [?] [?] [?]

Na **aprendizagem por reforço**, aprende-se a partir de uma série de reforços — recompensas ou punições. Não está disponível, geralmente, na aprendizagem por reforço, para o algoritmo de aprendizado de máquina, um conjunto de dados para treinamento. O aprendizado se dá, então, pela interação com o ambiente que se deseja atuar por um determinado período com o objetivo de melhorar o desempenho de uma determinada tarefa. [?] [?] [?]

2.6 Algoritmos de classificação

A classificação é considerada uma das tarefas principais do processo de aprendizagem de máquina, sendo, inclusive, a tarefa mais comum. [?] [?].

Segundo, [?, p. 86], "Na classificação, os dados devem possuir uma *classe* a qual queremos prever". Conforme [?], "o termo *classe* deve ser usado quando existe informação sobre quantas e quais são as partições presentes em um conjunto de dados, bem como qual exemplar pertence a qual partição"

Comumente denomina-se *classificação* o processo pelo qual se determina uma função de mapeamento capaz de indicar a qual classe pertence algum exemplar de um domínio sob análise, baseando-se em um conjunto já classificado. [?].

Assim, de acordo com [?], classificação é uma técnica de mineração de dados (aprendizado de máquina) usada para prever a associação ao grupo para instâncias de dados. É, segundo [?] e [?], como citado anteriormente, a tarefa mais utilizada em mineração de dados. Além de ser a mais complexa e a que possui a maior quantidade de algoritmos disponíveis, conforme descrito em [?].

A classificação é uma das tarefas *preditivas* de Mineração de Dados e aprendizado de máquina. Tarefas de predição consistem na análise de um *dataset* (conjunto de dados), descritos por atributos e rótulos associados com o objetivo de descobrir um **modelo** capaz de mapear corretamente cada um dos dados a seus rótulos apropriadamente. Esse objetivo é alcançado por meio de técnicas, normmalmente, chamadas de supervisionadas. Este tipo de análise preditiva pode ser dividida em *categórica*, também chamada de *classificação* ou em *numérica*, também chamada de regressão. Cf. [?] e [?], também exposto em [?] e [?]

Formalmente, a tarefa de classificação pode ser descrita como a busca por uma função

de mapeamento para um conjunto X de vetores de entrada (ou, exemplares — os dados) $\vec{x_i} \in E^d$ para um conjunto finito de rótulos C de cardinalidade c. A função F é, então, definida como $F : E^d \times W \rightarrow C$, em que d é a dimensão do espaço E, ou seja, a quantidade de coordenadas do vetor $\vec{x_i}$, e W é um espaço de parâmetros ajustáveis por meio do algoritmo de indução supervisionada. [?]

Pode ser dividida em, ao menos, duas categorias: *classificação binária* e *classificação multiclasse*. Na binária, a cardinalidade c é 2. Para o caso em que $c > 2$, o problema é considerado de múltiplas classes. [?] [?]

Os textos de [?], [?], além dos de [?], [?] e [?] trazem reflexões, técnicas, comparações e explicações detalhadas de muitos algoritmos de classificação, entre eles, árvores de decisão, k-vizinhos mais próximos, Naive Bayes e Redes Bayesianas, Redes Neurais Artificiais, Máquinas de Vetores de Suporte (SVM) entre outros.

Sobre *teoria da aprendizagem* e *algoritmos de classificação* há uma discussão em [?] sobre qual seria, em relação às hipóteses de modelos de aprendizagem, aquela (ou aquelas) que melhor se ajuste aos dados futuros. Os autores citam a ***suposição de estacionaridade***, ou seja, que há uma distribuição de probabilidade sobre os dados que permanece estacionária ao longo do tempo. Supõe-se, portanto que cada exemplo de ponto de dados (antes de conhecê-lo) é uma variável aleatória E_j cujo valor observado $e_j = (x_j, y_j)$ é amostrado da distribuição e é independente dos exemplos anteriores.

Assim:

$$P(E_j|E_{j-1}, E_{j-2}, ...) = P(E_j), \tag{2.2}$$

e cada exemplo tem uma distribuição de probabilidade anterior idêntica:

$$P(E_j) = P(E_{j-1}) = P(E_{j-2}) = \ldots \tag{2.3}$$

Estes exemplos são chamados de *independentes e identicamente distribuídos* ou **i.i.d**. Esta suposição é, segundo os autores, necessária para *tentar a previsão sobre o futuro dos dados*. Há claro, ainda em [?], um alerta sobre o fato de ser possível a aprendizagem ocorrer caso haja pequenas alterações (lentas) na distribuição.

Outro fato importante para a definição e avaliação da escolha da melhor hipótese (modelo) de um algoritmo de classificação é definir o "melhor ajuste". Em seu conhecido livro de Inteligência Artificial, os autores [?] definem a **taxa de erro** de uma hipótese como uma métrica importante para definir o "melhor ajuste" de um modelo/hipótese.

2.6.0.1 Taxa de erro

A taxa de erro é, assim, a proporção de erros que o algoritmo classificador comete — a proporção de vezes que $h(x) \neq y$ para o exemplo (x, y) — sendo $h(x)$ a função que mapeia uma *hipótese/modelo* h com a *previsão/valor* conhecido y. Nem sempre, como alerta, [**?**], uma hipótese/modelo h que tenha uma taxa de erro baixa no conjunto de treinamento generaliza bem e se comporta eficientemente para dados não conhecidos. A forma de testar o algoritmo é importante. Para isso há, na literatura, algumas técnicas que são utilizadas como estratégia de treinamento, validação e teste.

2.6.0.2 Estratégias de validação

Como citado por [**?**], [**?**] entre outros autores e, de acordo com [**?**, p. 125],

> independentemente da medida de avaliação a ser usada para atestar a qualidade de um modelo, não é adequado avaliá-lo [apenas] por seu desempenho em relação aos exemplares apresentados no processo de treinamento (indução). É sempre necessário saber como o modelo se comporta quando aplicado a exemplares que ainda não conhece, ou seja, não usados no processo de sintonização de seus parâmetros.

Essa observação, claro, é porque modelos preditivos, dependendo de como são criados, podem levar ao ***overfitting*** (sobreajuste). O fenômeno do sobreajuste ocorre, cf. [**?**] "quando o modelo preditivo é gerado de forma a representar os exemplares usados para sua geração com uma fidelidade mais alta que o necessário".

Assim, modelos sobreajustados (ou superajustados) não são capazes de realizar predições adequadas para dados novos. Busca-se, na construção de modelos preditivos, uma maior *generalização* e no caso do sobreajuste o fenômeno contrário ocorre. A maior causa de sobreajuste é quando os dados de treinamento não representam fielmente os dados novos (ou de produção) ou por serem diferentes (caso de dados antigos, por exemplo) ou por não serem significativos (poucos dados)[3]. A figura 2.3 mostra um modelo preditivo em que o lado direito representa um modelo sobreajustado e o lado esquerdo um modelo regularizado para o mesmo *dataset*.

Nas palavras de [**?**, p. 95], "o processo de construção de um modelo de aprendizado de máquina busca, obviamente, maximizar a precisão e minimizar a taxa de erros". Para

[3] é possível também o sobreajuste devido a *ruídos* nos dados, pelo uso de um modelo de forma inapropriada ou de uma *classe rara* [**?**, p.35]

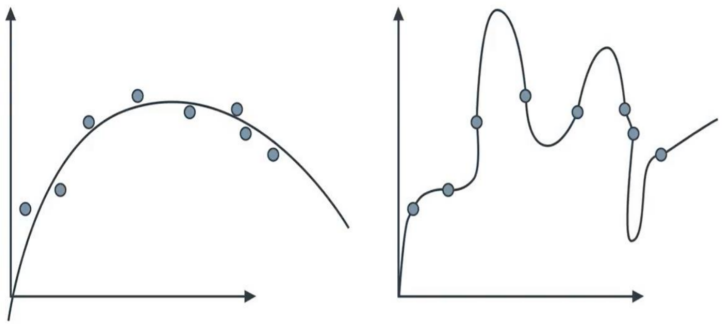


Figura 2.3: O lado direito representa um modelo sobreajustado e o lado esquerdo um regularizado para o mesmo *dataset*.

Fonte: baseado no encontrado em [?]

tanto, a literatura cita diversas técnicas e estratégias para a avaliação de modelos preditivos. Autores diversos, como [?], [?], além de [?] e [?] citam, geralmente, como estratégia de treinamento, validação e teste as seguintes técnicas:

- Resubstituição;
- Holdout;
- Validação cruzada;
- Bootstrap;

Na **resubstituição**, segundo [?], as medidas de avaliação dos classificadores são aplicadas no próprio conjunto de dados usados para indução do modelo. Essa técnica, embora tenha alguns vantagens discutidas em [?] e [?], pode levar ao, já citado, sobreajuste (*overfitting*) e é fartamente discutido em [?], também em [?] e [?]. Como já afirmado anteriormente, o sobreajuste é quando se produz um modelo de bom desempenho com os dados de treinamento, mas que não lida bem com novos dados.

Na técnica de **Holdout**, pressupõem-se uma divisão, ou criação de dois subconjuntos de dados distintos, a partir do conjunto de dados disponível pra uso na indução do modelo/hipótese. Um desses subconjuntos será usado para treinamento (indução) do modelo de previsão e o segundo, para teste após o término do treinamento e, consequentemente, na aplicação das medidas de avaliação do modelo/hipótese. [?]

A imagem 2.4 mostra o funcionamento da técnica de holdout de forma mais detalhada

Na estratégia de **validação cruzada**, todos os dados farão parte, em algum momento, do conjunto de dados usado no teste do modelo/hipótese. A ideia é que cada exemplo

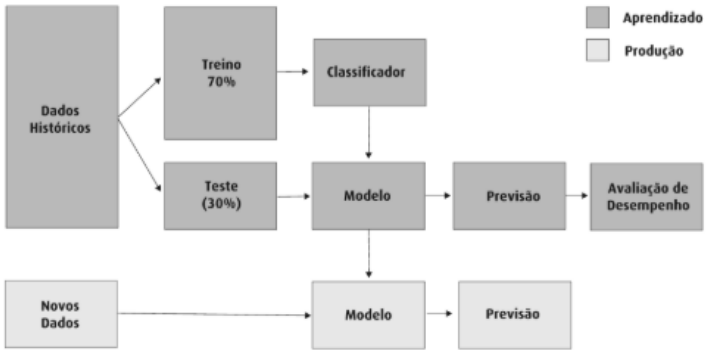


Figura 2.4: Funcionamento da técnica holdout.

Fonte:[?]

sirva duplamente — como dados de treinamento e dados de teste. Primeiro divide-se o conjunto em k subconjuntos iguais. Em seguida realiza-se k rodadas de aprendizagem; em cada iteração $\frac{1}{k}$ dos dados é retido como conjunto de teste e os exemplos restantes são usados como treinamento.

Valores populares de k são 5 e 10 — o suficiente para uma estimativa estatisticamente provável que seja precisa a um custo 5-10 vezes maior no tempo de computação. Há também o extremo do $k = n$, também conhecido como **validação cruzada com omissão de um**. O método de validação cruzada permite que o modelo/hipótese seja avaliado uma série de vezes, cada série sendo conhecida como partição (ou *fold*). Ao final, a avaliação pode ser realizada aplicando medidas estatísticas como média, desvio-padrão e intervalo de confiança ao conjunto de k avaliações obtidas ou somando-se os desempenhos obtidos pelos k modelos gerados e dividindo essa soma pelo número de exemplares original. [?][?][?] [?]

A imagem 2.5 mostra um exemplo didático de como funciona a validação cruzada. Os dados que não fazem parte do conjunto de teste em cada rodada são utilizados no treinamento.

Figura 2.5: Funcionamento da técnica *cross validation*

Fonte:[?]

Já a técnica de ***Bootstrap*** funciona de forma parecida à estratégia *holdout*. Ela também usa dois conjuntos, um de treinamento e outro para teste, porém durante o processo de formação dos subconjuntos, exemplares que já foram sorteados podem novamente serem contemplados, com probabilidade igual. É uma estratégia que permite, portanto, a reposição.

Neste trabalho, todos os algoritmos de classificação usados foram testados usando as técnicas de resubstituição, *holdout* (com taxas de 70-30, 75-25 e 60-40), além de *cross-validation* com 3, 5 e 10 folds.

Como explicado em [?] e [?] não existe um algoritmo de aprendizado superior a todos os demais quando considerados todos os problemas de classificação possíveis (teorema **NFL**, ou *No Free Lunch*), portanto, variações foram executadas nos experimentos em todas técnicas avaliadas, alterando-se os padrões para chegar a métricas e medidas de avaliação mais eficientes.

2.6.0.3 Medidas de avaliação

Para [?],

> Tradicionalmente, a métrica usada na avaliação e seleção de modelos de classificação é a acurácia (ou taxa de erro) estimada em relação a um dado conjunto de teste. Essa metodologia é justificada pela formulação padrão do problema do aprendizado supervisionado que visa a minimização da probabilidade do erro global.

Há, porém, conforme [?], [?], [?] e [?] diversas medidas usadas na avaliação de classificadores.

A que será usada neste trabalho (com o objetivo de minimizar a probabilidade do erro globla) é a, já citada, acurácia ou taxa de classificações corretas.

A métrica acurácia é dada, portanto, por:

$$\text{Acurácia} = |y - f(\aleph) = 0|, \tag{2.4}$$

em que $|\cdot|$ representa a contagem de vezes em que $\cdot$ é verdadeiro, f é o modelo preditivo, $\aleph$ é o subconjunto de dados sob o qual o modelo está sendo avaliado, $f(\cdot)$ é a classificação fornecida pelo modelo preditivo para cada um dos exemplares (dos dados),

e y é a classe esperada como resposta. [?, p. 129]

A acurácia de um classificador também pode ser descrita em termos do **erro de generalização** ξ_g, e uma função de perda binária e, portanto, ser interpretada como a probabilidade de ocorrer uma classificação correta. Dessa forma:

$$\text{Acurácia}_g = 1 - \xi_g \tag{2.5}$$

Ou seja, a acurácia é, basicamente o número de acertos (positivos) divido pelo número total de exemplos. Será a métrica mais usada para avaliar os classificadores neste trabalho.

Há, entretanto, em modelos preditivos que trabalham com dados numéricos (como as redes neurais e o SVM), pode-se utilizar uma função de perda contínua, capaz de medir o erro entre a resposta obtida pelo modelo preditivo e a resposta aguardada. [?] [?].

Para [?]

> A função de custo reduz todos os aspectos bons e ruins de um sistema complexo a um único número, um valor escalar, o que permite ranquear e comparar as soluções candidatas.

Algumas funções de perda (*loss functions*), comuns, neste contexto são: [?]

$$\textit{Erro absoluto} = \sum_{<\vec{x}_i, y_i> \in \aleph} |y_i - f(\vec{x}_i)| \tag{2.6}$$

$$\textit{Erro quadrático} = \sum_{<\vec{x}_i, y_i> \in \aleph} (y_i - f(\vec{x}_i))^2 \tag{2.7}$$

$$\textit{Erro absoluto médio} = \sum_{<\vec{x}_i, y_i> \in \aleph} |y_i - f(\vec{x}_i)|/m \tag{2.8}$$

$$\textit{Erro médio quadrático} = \sum_{<\vec{x}_i, y_i> \in \aleph} (y_i - f(\vec{x}_i))^2/m \tag{2.9}$$

Onde m é a quantidade de instâncias existentes em $\aleph$. [?], [?] além de [?] e [?] afirmam que é importante analisar o tipo de erro que o modelo está cometendo e, principalmente, no caso de classificadores binários, é possível observar que, mesmo com uma acurácia alta o classificador pode não estar respondendo de forma adequada. Para realizar esse tipo de análise, os autores citados no princípio deste parágrafo, citam que uma ferramenta

adequada é a ***matriz de confusão***.

Geralmente, uma matriz de confusão tem dimensões $C \times C$, em que C é o número de classes presentes no problema de classificação que está sendo avaliado. As linhas dessa matriz são indexadas seguindo as ´´classes esperadas" (y), e as colunas seguindo as "classes preditas" ($f(x) ou \hat{y}$). Cada célula é um contador que é incrementado a depender do resultado da comparação de $f(x)$ e y. As respostas corretas do modelo geram os valores que entram na diagonal principal da matriz. [**?**, p. 130]

A figura 2.6 demonstra o procedimento de preenchimento de uma matriz de confusão conforme a resolução de um problema hipotético de classificação.

Problema sob resolução	
f(x)	y
1	1
1	1
1	1
2	1
2	2
2	2
2	2
3	1
3	2
3	3

Classe Esperada (y)	Classe Predita (f(x))		
	Classe 1	Classe 2	Classe 3
Classe 1	3	1	1
Classe 2		3	1
Classe 3			1

Figura 2.6: Preenchimento de uma matriz de confusão

Fonte: [**?**, p. 130]

Matrizes de confusão são, cf. [**?**]e [**?**], particularmente úteis para avaliação de classificadores binários. Neste caso, os valores são atribuídos conforme ilustrado na figura (neste caso, as duas classes estão definidas como ´´classe positiva" e "classe negativa").

Classe Esperada (y)		*Classe Predita (f(x))*	
		positivo	*negativo*
	positivo	*Verdadeiros positivos (VP)*	*Falsos negativos (FN)*
	negativo	*Falsos Positivos (FP)*	*Verdadeiros negativos (VN)*

Figura 2.7: Matriz de confusão para o caso de um classificador binário

Fonte: [**?**, p. 131]

Cada célula, neste caso, possui o seguinte significado:

- Verdadeiro Positivo (VP): classificação correta na classe positiva. A instância pertence a classe positiva e o modelo classificou na classe positiva.
- Falso Positivo (FP): classificação incorreta na classe negativa. A instância pertence à classe negativa, mas o classificador a classificou como pertencente à classe positiva.
- Verdadeiro Negativos (VN): classificação correta na classe negativa. A instância pertence a classe negativa e o modelo classificou na classe negativa.
- Falso Negativo (FN): classificação incorreta na classe negativa. classificação incorreta na classe negativa. A instância pertence à classe positiva, mas o classificador a classificou como pertencente à classe negativa.

2.7 Redes Neurais Artificiais - RNA

As redes neurais instituem um campo da ciência da computação, parte da área da inteligência artificial, que busca efetivar modelos matemáticos que se assemelhem às redes neurais biológicas. Elas apresentam capacidade de adaptar seus parâmetros como resultado da interação com o meio externo. [?][?]

2.7.1 Definição

Para [?, p. 41]

> Uma rede neural artificial consiste de um sistema composto por neurônios dispostos em camadas, interligados através de pesos sinápticos construindo um sistema que simula o cérebro humano, inclusive seu comportamento. E uma técnica computacional que apresenta um modelo inspirado na estrutura neural de organismos inteligentes e que adquire conhecimento através da experiência.

De acordo com [?, p. 47], “redes neurais podem ser caracterizadas como modelos computacionais com capacidades de adaptar, aprender, generalizar, agrupar ou organizar dados".

Inicialmente, portanto, se desenvolveram como uma estratégia de simular os processos mentais humanos, como reconhecimento de imagens e sons, e após, como instrumento tecnológico e eficiente para muitas tarefas. [?]

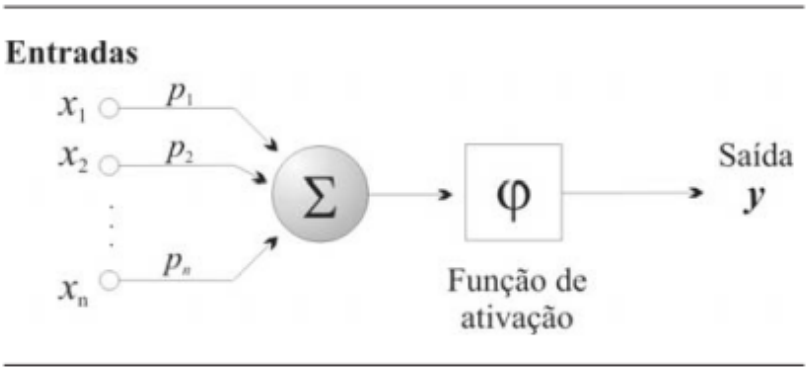
Para [?], as redes neurais artificiais podem ser usadas efetivamente para prover soluções para um amplo espectro de aplicações, incluindo mapeamento de padrões e classificação, análise e codificação de imagens, processamento de sinais, otimização, manipulação de grafos, reconhecimento de caracteres, reconhecimento automático de alvo, fusão de dados, processamento de conhecimento, controle de qualidade, mercado de ações, processamento de hipotecas, triagem de créditos para empréstimos entre muitos outros problemas.

2.7.2 Modelo de neurônio artificial

Desde a década de 1940 com o trabalho de [?] que se busca um modelo computacional que simule o cérebro humano e suas conexões. O interesse pela pesquisa nesta área cresceu e se desenvolveu durante os anos 50 e 60. É dessa época que [?] sugeriu um método de aprendizagem para as redes neurais artificiais chamado *perceptron*.

Até o final da década de 1960 muitos trabalhos foram feitos usando o percepton como modelo, mas ao final desta década, [?] apresentaram significativas limitações do perceptron.

A pesquisa diminui consideravelmente nos anos seguintes (o chamado inverno da IA), porém durante os anos 80, a excitação ressurge mediante os avanços metodológicos importantes e, também, ao aumento dos recursos computacionais disponíveis. O modelo de neurônio artificial da figura 2.8 é uma simplificação do apresentado por [?, p. 36]



Fonte: [?, p. 36]

Figura 2.8: Modelo matemático de um neurônio

Este modelo acima (da figura 2.8) é composto por três elementos:

- um conjunto de n conexões de entrada $(x_1, x_2, \ldots, x_n)$, caracterizadas por pesos $(p_1, p_2, \ldots, p_n)$;

- um somador ($\sum$) para acumular os sinais de entrada;
- uma função de ativação (φ) que, no caso específico do neurônio apresentado por McCullock-Pitts [?] é uma função de limiar. [?] [?]

O comportamento das conexões entre os neurônios é simulado através de seus pesos $(p_1, p_2, ..., p_n)$. Os valores podem ser positivos ou negativos (dependendo se a conexão é inibitiva ou excitativa.

O efeito de um sinal proveniente de um neurônio é determinado pela multiplicação do valor do sinal recebido pelo peso da conexão correspondente $(x_i \times p_i)$.

Então é efetuada a soma dos valores $x_i \times p_i$ de todas as conexões e o valor resultante é enviado para a função de ativação que define a saída (y) do neurônio. cf. [?] [?], além de [?], [?] e [?]

Matematicamente, a saída y_k do neurônio mostrado na figura 2.8 é dada pela expressão abaixo:

$$Y_k = \varphi(u_k) \qquad \text{onde} \qquad u_k = \sum_{i=1}^{n} P_{ki} X_i \tag{2.10}$$

A função de ativação proposta inicialmente por McCullock e Pitts em seu trabalho seminal, [?] é uma função de limiar:

$$y_k = \varphi(u_k) = \begin{cases} 1, & \text{se } u_k > 0 \\ 0, & \text{se } u_k < 0 \end{cases} \tag{2.11}$$

Uma alteração importante neste modelo foi a introdução de um parâmetro polarizador (*bias* ou *offset*) b_k, conforme a figura 2.9 cujo objetivo é deslocar o valor da informação referente à entrada líquida u_k, de forma a verter a função de ativação no eixo correspondente ao valor de u_k. Assim a saída $y_k = \varphi(u_k)$.

Dessa forma, o polarizador pode ser tratado como mais um peso da rede. Basta considerar um nova entrada do tipo $x_0 = 1$, com um peso associado $w_0 = b_k$, assim a representação fica sendo, considerando os pesos como w, de acordo com a figura 2.9:

$$u_k = w_0 + w_1 x_1 + w_2 x_2; \qquad y_k = \varphi(u_k) \tag{2.12}$$

Assim, esse neurônio pode ser empregado, segundo, [?, p. 58], para "separar classes distintas de padrões de entradas para aplicações de classificações de padrões". E prossegue:

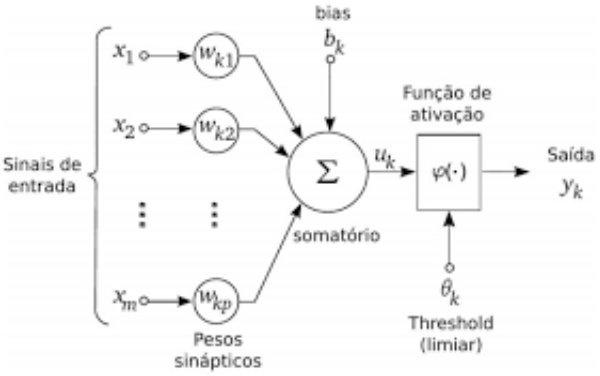


Figura 2.9: Adição de um *offset* (*bias*) no modelo do neurônio
Fonte: [?, p. 58]

"se a entrada líquida for maior que o limiar, o padrão dessa entrada pertence à classe 1, caso contrário, pertence à classe 0". Analisando matematicamente o modelo *perceptron*, se percebe que ele pode ser considerado um típico caso de discriminador linear.

A fronteira de decisão para um *perceptron* como o da figura 2.9, dada duas entradas e a função de ativação mostrada na equação 2.11 será, então, uma reta cuja equação é definida por:

$$w_1 x_1 + w_2 x_2 - b = 0 \tag{2.13}$$

Portanto, cf. [?], "pode-se concluir que o Perceptron se comporta como um classificador de padrões cuja função é dividir classes que sejam linearmente separáveis".

A figura 2.10 mostra uma reta posicionada na fronteira de separação entre as classes. Para este tipo de problema o perceptron é um classificador adequado.

As redes neurais artificiais (**RNA**) se formam quando diversos neurônios se combinam. De forma resumida, "uma rede neural artificial (RNA) pode ser vista como um grafo onde os nós são os neurônios e as ligações fazem a função das sinapses". Isto está demonstrado na figura 2.11

As redes neurais artificiais se diferem pelas suas arquiteturas e pela forma como os pesos associados às conexões são ajustados durante o processo de aprendizado. A arquitetura de uma rede neural restringe o tipo de problema no qual a rede poderá ser utilizada, e é definida pelo número de camadas (camada única ou múltiplas camadas), pelo número de nós em cada camada, pelo tipo de conexão entre os nós (*feedforward* ou *feedback*) e por sua topologia. [?, p. 46-49]

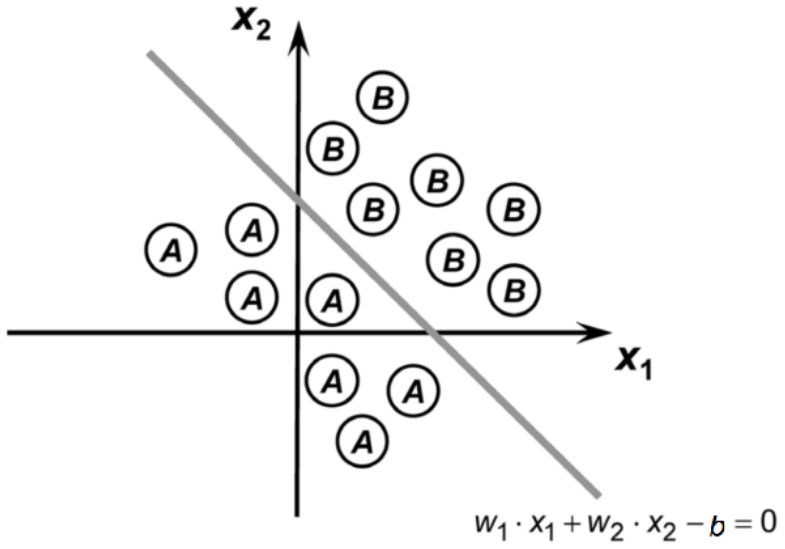


Figura 2.10: Fronteira de separação (perceptron com duas entradas)
Fonte: [?, p. 62]

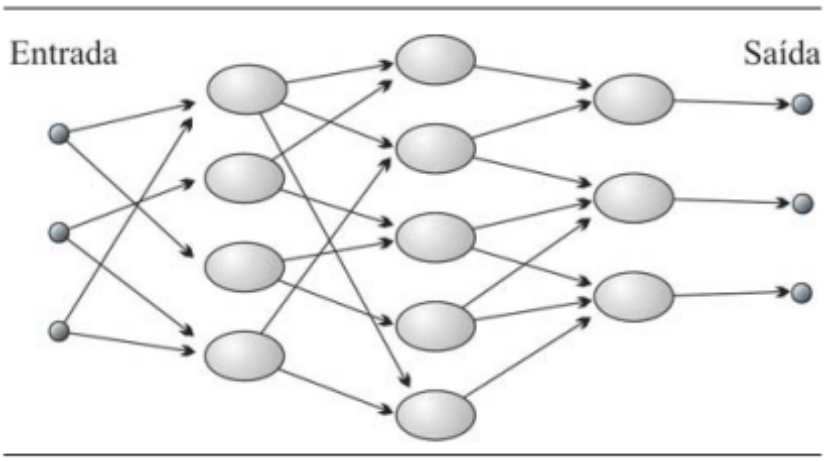


Figura 2.11: Representação simplificada de uma RNA
Fonte: [?, p.26]

O desenvolvimento de uma rede neural artificial consiste em determinar sua arquitetura, ou seja, os números de camadas e de neurônios em cada camada, bem como o ajuste dos pesos na fase conhecida como treinamento.[?] [?]

Uma das características mais importantes de uma rede neural artificial é a habilidade de aprender através de exemplos e fazer inferências sobre o que aprendeu, melhorando, assim, o seu desempenho. As RNA's utilizam um algoritmo de aprendizagem que serve, basicamente, para ajustar os pesos de suas conexões. [?] [?] [?] [?]. Aqui também há, cf. explicitado na seção 2.5, duas formas básicas de aprendizado, o supervisionado e o não-supervisionado.

2.7.3 Redes do tipo Perceptron de múltiplas camadas

As arquiteturas do tipo perceptron de múltiplas camadas (MLP) são os modelos de redes neurais mais utilizados e conhecidos. Elas, basicamente, consistem de uma camada de entrada e uma ou mais camadas intermediárias (ou ocultas) além, claro da camada de saída (uma ou mais unidades sensoriais - neurônios). [?].

Os sinais de entrada são propagados camada a camada pela rede em uma direção, ou seja, da entrada para a saída (*feedfoward*). Esta arquitetura retrata uma generalização do perceptron. Portanto, segundo [?, p. 26],

> As redes Perceptron de múltiplas camadas (PMC) são caracterizadas pela presença de pelo menos uma camada intermediária (escondida) de neurônios, situada entre a camada de entrada e a respectiva camada neural de saída.

Desta forma, elas possibilitam elevadas possibilidades de aplicações em muitas áreas do conhecimento, entre as principais: aproximação universal de funções, reconhecimento de padrões, identificação e controle de processos, previsões de séries temporais, otimização de sistemas entre muitos outros. [?].

A figura 2.12 ilustra uma rede do tipo perceptron de multicamadas. O treinamento deste tipo de rede é do tipo supervisionado e, geralmente, se utiliza um algoritmo muito popular chamado *retropropagação* do erro (*error backpropagation*). Este algoritmo é baseado numa regra de aprendizagem que "corrige" o erro durante o treinamento. [?]

Substancialmente, o método de *retropropagação* é constituído de duas etapas: uma fase de propagação do sinal no sentido tradicional (*feedforward*) e uma de retropropagação do erro (*backpropagation*) de todas as camadas e seus respectivos pesos. Na fase de ida, os vetores de dados e pesos são aplicados às unidades de entrada, e seu efeito se propaga pela rede, camada por camada. [?] [?]

Após isso, um conjunto de saídas é produzido como resposta da rede. Na retropropagação os pesos são ajustados de acordo com uma regra de correção de erro (normalmente uma função matemática com 2.6, 2.8, 2.7 e 2.9, sendo esta última uma das mais utilizadas em muitas arquiteturas, na prática). [?] [?] [?]

A resposta da rede em um instante é subtraída da saída desejada (*target*) para produzir um valor de erro. Este valor de erro é propagado da saída para a entrada, camada a camada, de onde vem o nome "*retropropagação* do erro". Os pesos são, então, redefinidos de forma que a distância entre a resposta da rede e a resposta desejada seja reduzida (o

erro seja minimizado). O processo é repetido diversas vezes até que uma tolerância global de erro seja assumida. Cada iteração é denominada época (*epoch*). [?] [?] [?].

Uma arquitetura de um MLP (*Multi-Layer Perceptron* - Perceptron Multicamadas), possui, portanto três propriedades distintas: a função de ativação, o número de camadas ocultas e forma das conexões (totalmente conectada ou não).

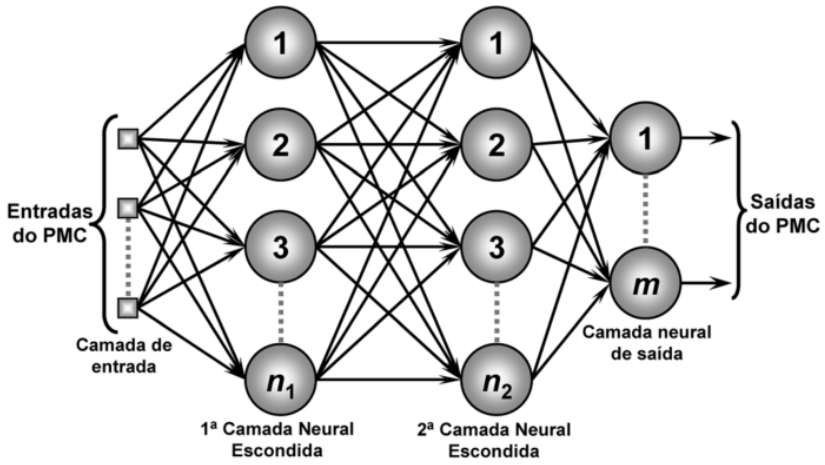


Figura 2.12: Rede Perceptron de multicamadas

Fonte: [?, p. 92]

2.7.4 Funções de ativação

Como visto na seção 2.7.2, o modelo perceptron, lida bem com problemas linearmente separáveis, mas tem problemas ao lidar com problemas não-lineares. [?]. Para dar capacidade representativa às redes neurais artificiais, são essenciais as diferentes funções de ativação, pois só assim elas conseguirão lidar um componente de não-linearidade, como o são a maioria dos problemas práticos. [?].

Ao se introduzir ativações não-lineares, a superfície de custo da rede neural deixa de ser convexa fazendo com que a otimização se torne mais difícil. [?] [?].

As principais funções de ativação, utilizadas na literatura ena prática, são, portanto:

2.7.4.1 Função de ativação limiar

Proposta na primeira definição de rede com neurônios artificiais [?]. O modelo de neurônio usado por RosenBlatt foi o mesmo sugerido por [?]. Neste modelo as saídas são binárias, ou seja, assumem, normalmente o valor 0 ou 1. A saída é 1 se o valor da entrada líquida

for superior a um determinado valor chamado *thereshold*. na maioria das vezes, esse valor é zero.

Matematicamente:

$$y_k = y = f_i(a_i(t)) = \begin{cases} 1, & \text{se } a_i(t) >= 0 \\ 0, & \text{se } a_i(t) < 0 \end{cases} \tag{2.14}$$

A imagem 2.13 mostra o 'funcionamento' da função de ativação limiar. Eventualmente podem ser utilizados os valores -1 e 1. Uma rede de camada simples que utiliza este tipo de ativação é o perceptron simples.

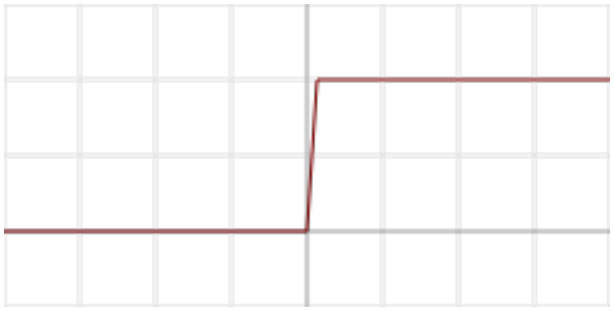


Figura 2.13: Função de ativação Limiar

Fonte: Adaptado de [?]

2.7.4.2 Função de ativação linear

A saída do neurônio, neste caso, é representada por uma função linear da forma descrita na figura 2.14. Redes que usam este tipo de função de ativação apresentam apenas uma camada de entrada e uma de saída. Possuem, portanto, uma série de representações quanto ao que são capazes de representar. [?]

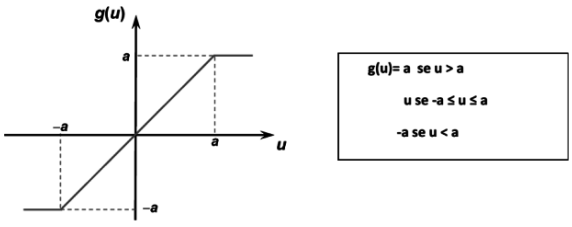


Figura 2.14: Função de ativação Linear

Fonte: Adaptado de [?]

2.7.4.3 Funções de ativação semilineares

As funções de ativação mais usadas deste tipo são a função logística e a tangente hiperbólica. Elas são populares por conta de suas derivadas (que são necessárias nas etapas de treinamento, particularmente no *Gradiente Descent* — descida do gradiente) poderem ser expressas a partir das próprias funções.

Estas funções, tanto a logística (ou *sigmoide*) e a tangente hiperbólica, respectivamente, podem ser expressas pelas equações 2.15 abaixo, onde a representa a entrada líquida da unidade. [?] [?]

$$f(a) = \frac{1}{1 + e^{(-2\beta a)}} \qquad g(a) = tanh(\beta a) \tag{2.15}$$

E as respectivas derivadas das funções 2.15 acima são dadas por:

$$f'(a) = 2\beta f(1 - f) \qquad g'(a) = \beta(1 - g^2) \tag{2.16}$$

A figura 2.15 mostra o comportamento da função sigmoide e a figura 2.16 mostra o da função tangente hiperbólica.

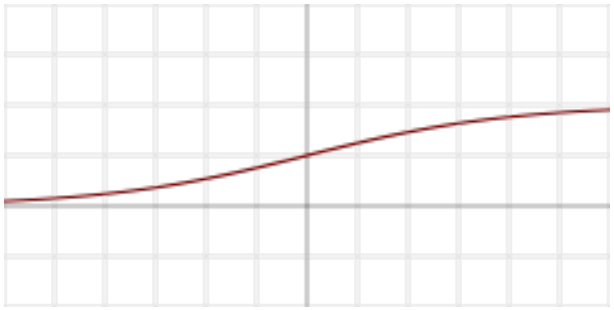


Figura 2.15: Função de ativação Logística (sigmoide)
Fonte: Adaptado de [?]

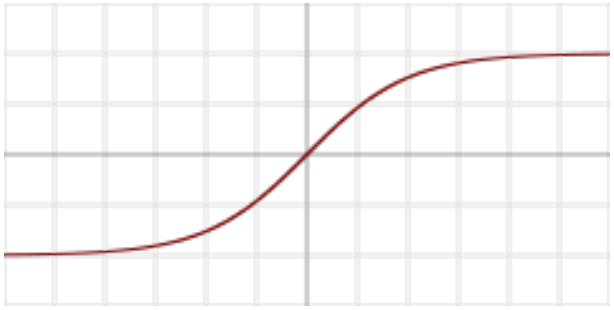


Figura 2.16: Função de ativação Tangente Hiperbólica
Fonte: Adaptado de [?]

2.7.4.4 Função de ativação ReLu

A função ReLU é a unidade linear retificada. É definida como: [?]

$$ReLU(x) = max(0, x) \tag{2.17}$$

Sua derivada é dada por:

$$ReLU'(x) = \begin{cases} 1, & \text{se } x > 0 \\ 0, & \text{caso contrário} \end{cases} \tag{2.18}$$

Segundo, [?],

> ReLU é a função de ativação mais amplamente utilizada ao projetar redes neurais atualmente. Primeiramente, a função ReLU é não linear, o que significa que podemos facilmente copiar os erros para trás e ter várias camadas de neurônios ativados pela função ReLU.

Ainda, conforme [?]

> A principal vantagem de usar a função ReLU sobre outras funções de ativação é que ela não ativa todos os neurônios ao mesmo tempo. [...] Se [...] a entrada for negativa, ela será convertida em zero e o neurônio não será ativado. Isso significa que, ao mesmo tempo, apenas alguns neurônios são ativados, tornando a rede esparsa e eficiente e fácil para a computação.

A figura2.17 demonstra o comportamento da função de ativação ReLu em função da entrada

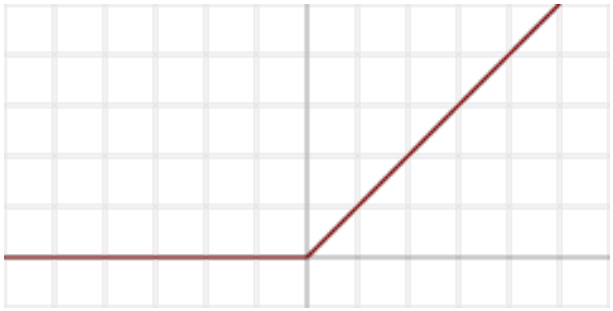


Figura 2.17: Função de ativação ReLu (sigmoide)
Fonte: Adaptado de [?]

Redes com a função ReLU são mais fáceis de otimizar porque a função é parecida

com a função identidade. A única diferença é que a ReLU produz zero em metade de seu domínio. [?].

2.8 Otimização

Redes Neurais tem uma etapa iterativa em que os pesos são ajustados e se pode realizar otimizações para melhorar o erro e incrementar o modelo. [?].

Normalmente, o fluxo de treinamento se resume nos seguintes passos iterativos:

1. Opera a entrada na rede
2. Cálculo da função de perda (*loss function*) ou outra função de cálculo de erro
3. Cálculo do gradiente
4. Atualização dos pesos
5. Volta para o passo 1

Este processo é repetido até que um hiperparâmetro de tolerância seja alcançado para o cálculo dos erros mediante a *loss function* ou outra função de cálculo do erro.

O cálculo do gradiente descrito anteriormente diz respeito, cf. [?] a um dos mais usados algoritmos para otimizar a tarefa de aprendizagem dos pesos de uma rede neural.

De acordo com [?]:

> A Descida do Gradiente é uma ferramenta padrão para otimizar funções complexas iterativamente dentro de um programa de computador. Seu objetivo é: dada alguma função arbitrária, encontrar um mínimo. Para alguns pequenos subconjuntos de funções – aqueles que são convexos – há apenas um único *minimum* que também acontece de ser global. Para as funções mais realistas, pode haver muitos mínimos, então a maioria dos mínimos são locais.[... É preciso] que a otimização encontre o "melhor" *minimum* e não fique preso em mínimos sub-otimistas (um problema comum durante o treinamento do algoritmo).

O algoritmo consiste basicamente de subtrair o valor do gradiente ∇f dos pesos w da rede, assim:

$$w_i = w_i - \alpha \times \nabla f_i \tag{2.19}$$

Sendo α o multiplicador que nos permite controlar o tamanho do passo de otimização. ∇f é a derivada da função de ativação no ponto específico. α é um hiperparâmetro conhecido como taxa de aprendizado e repreenta a *velocidade* em que a rede neural "aprende" os melhores pesos para o problema específico. [?].

Uma iteração consiste em um passo de otimização como o descrito em 2.8 e corresponde a uma ligação sináptica de *forward* na rede e uma de *backpropagation*. Isto constitui uma **época**. Normalmente, muitas épocas são necessárias, visto que o aprendizado em uma rede neural é, geralmente, lento para que a otimização evite os mínimos locais.

2.9 SVM - Support Vector Machines

Segundo [?], o algoritmo SVM (*Support Vector Machines*) é um dos mais efetivos para a tarefa de classificação.

Cf. [?],

> No algoritmo SVM, o conjunto de dados de entrada é utilizado para construir uma *função de decisão* $f(x)$, tal que:
>
> $$\begin{aligned} &Se \quad f(x_i) \geq 0, \quad \text{então} \quad y_i = 1 \\ &Se \quad f(x_i) < 0, \quad \text{então} \quad y_i = -1 \end{aligned} \tag{2.20}$$
>
> O algoritmo SVM constrói os denominados classificadores lineares, que separam o conjunto de dados por meio de um *hiperplano* que é a generalização do conceito de *plano* para dimensões maiores que três.

Assim, SVM, cf. [?, p. 45] "são um algoritmo de classificação que maximizam as margens entre instâncias mais próximas, dessa forma, é criado um vetor otimizado que é então utilizado para classificar novas instâncias".

Conforme se vê na figura 2.18, os dois vetores *não pontilhados* são as margens otimizadas. As instâncias por onde as margens otimizadas passam são os vetores de suporte. O vetor pontilhado é a referência para classificar novas instâncias. Assim, a nova instância, na figura 2.18 é classificada como triângulo.

Seguindo o estudo de [?] há duas razões principais que levaram os autores do artigo citado de usarem SVMs para detecção de intrusão:o primeiro é a velocidade já que a performance é prioritariamente uma das características mais importantes para sistemas de detecção de intrusos.

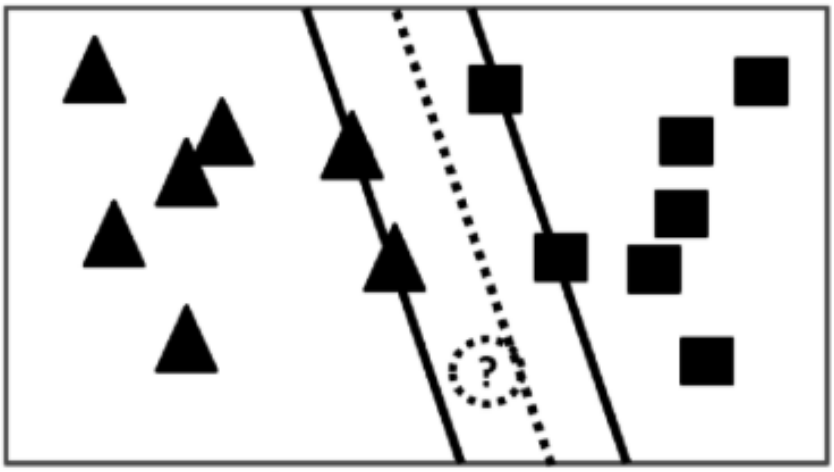


Figura 2.18: Vetores de Suporte
Fonte: [?, p. 45]

A segunda razão é a escalabilidade, pois, cf. os autores, SVMs são relativamente indiferentes ao número de *data points* e a complexidade da classificação não depende da dimensionalidade do espaço de características. Dependendo da aplicação, ainda conforme os autores, uma vez que os dados estão classificados em duas classes, um algoritmo de otimização adequado pode ser usado, se necessário, para identificação de mais características.

Neste trabalho, também foi usado, com boa eficácia (cf. se vê na seção 3) o algoritmo SVM.

2.10 Trabalhos Relacionados

Esta seção expõe alguns dos principais trabalhos encontrados na literatura que estão, de alguma forma, relacionados a esta dissertação. Foram estudadas e analisadas obras acerca das políticas de segurança da informação, abordando a detecção de conflitos em diversos contextos como identificação e resolução de conflitos aéreos, rodoviários e de *normas* além de trabalhos sobre mineração de dados e técnicas de aprendizagem de máquina tanto em conjunturas distintas como regressão e classificação quanto para localização e solução de conflitos, no caso específico das políticas de segurança, principalmente sendo usadas em registros de patentes americanas fechadas — o que, de certa forma, evidencia o quanto este tema está na vanguarda.

Por exemplo, [?] aborda um método para construir modelos de *aprendizagem de máquina* com o objetivo de detectar ataques cibernéticos em RTLSs (*Real Time Location*

Systems - sistemas de localização em tempo real) em um ambiente de cibersegurança para sistemas robóticos usando técnicas de *Machine Learning*. O artigo mostra que os ciberataques nos sistemas de localização em tempo real para sistemas robóticos podem ser detectados por um sistema criado usando o aprendizado supervisionado. Além disso, mostra que alguns tipos de ciberataques em sistemas de localização em tempo real, especificamente negação de serviço e falsificação(*DoS* e *Spoofing*), podem ser detectados por um sistema construído usando técnicas de aprendizado de máquina. Oito classificadores e algoritmos preditores conhecidos foram avaliados neste artigo e a análise de validação cruzada mostrou que os classificadores MLP (*Multi Layer Perceptron*) funcionam melhor que os outros obtendo maior acurácia e menor erro, sendo também o modelo com menor *overfitting* e maior *sensibilidade*[4]. Este artigo inspirou a hipótese desta dissertação.

Já [?] versa acerca de mineração de dados em políticas de controle de acesso, especificamente do modelo ReBAC (*Relationship-Based Access Control* - controle de acesso baseado em *relacionamento*), mas não foca na detecção de conflitos entre as políticas, e sim, na propagação de seus relacionamentos na criação de novas políticas. No artigo, os algoritmos de mineração de política do ReBAC propostos puderam reduzir significativamente o custo da migração dos sistemas de controle de acesso legados para o ReBAC, automatizando parcialmente o desenvolvimento de uma nova política do ReBAC.

O trabalho de [?] discorre sobre conflitos no gerenciamento de sistemas distribuídos com base em políticas de controle de acesso. O artigo analisa os conflitos de políticas, concentrando-se nos problemas de detecção e resolução dos mesmos. Discute-se, no artigo, os vários relacionamentos de precedência que podem ser estabelecidos entre as políticas e é apresentado uma ferramenta de análise de conflitos que faz parte de uma estrutura de gerenciamento baseada em funções, porém, sem usar mineração de dados ou aprendizagem de máquina.

O artigo de [?], estudou, usando como base grafos, a detecção e resolução de conflitos nas especificações das políticas de controle de acesso. Os autores usam propriedades formais de transformações de grafos para detectar sistematicamente inconsistências entre restrições, entre regras e entre uma regra e uma restrição em políticas de controle de acesso e estabelecer as bases para suas resoluções, porém, também não usando mineração de dados ou aprendizagem de máquina.

Já [?], abordaram a detecção de conflitos em políticas de segurança usando as tecno-

[4]Sensibilidade é a proporção de verdadeiros positivos: a capacidade do sistema em predizer corretamente a condição para casos que realmente a têm - É também conhecida como ***Recall***.

logias da Web Semântica. É mostrado, no artigo, como as ferramentas fornecidas pelas tecnologias de gerenciamento de Web Semântica e Ontologia oferecem uma base adequada para a realização de técnicas capazes de suportar a análise de conflitos nas políticas de segurança. Com base no uso dessas técnicas, é proposto no artigo, uma solução para duas variantes diferentes de análise de conflitos: (a) incompatibilidade de políticas e (b) satisfação de separação de tarefas.

[?] aborda um sistema de rede neural multicamadas para segurança de acesso a computadores com o objetivo de identificar usuários do mesmo. Os vetores de entrada foram compostos pelos intervalos de tempo entre pressionamentos de teclas sucessivos criados pelos usuários ao digitar uma sequência conhecida de caracteres. Usando aprendizado supervisionado, cada vetor de entrada foi classificado em uma das várias classes, identificando assim o usuário que digitou a sequência de caracteres. Neste artigo, uma precisão máxima de classificação de 97,5% foi alcançada usando um classificador de padrões baseado em rede neural multicamadas *feedforward* treinada usando *backpropagation*, o algoritmo de retropropagação. Essa abordagem visa melhorar a segurança do acesso ao computador. Este artigo trouxe importantes contribuições para esta dissertação na maneira de construir a arquitetura da rede neural e também no ajuste dos hiperparâmetros.

O trabalho de [?] aborda a detecção de conflitos sobre a ótica da prevenção de colisões no voo livre de aeronaves comerciais usando redes neurais com exemplos preparados por meio de programação não linear.

Na mesma linha, o artigo de [?] estuda a detecção de invasões usando redes neurais e máquinas de vetores de suporte e a ideia central é descobrir padrões úteis ou características que descrevam o comportamento intrusivo de um usuário em um sistema e os autores usam este conjunto de características para construir classificadores que puderam reconhecer anomalias e intrusões conhecidas em tempo real. É usado um conjunto de dados de referência de uma competição de KDD (*Knowledge Discovery in Databases* — Descoberta de Conhecimento em Bases de Dados) projetada pela DARPA (Defense Advanced Research Projects Agency), e é demonstrado que classificadores eficientes e precisos podem ser construídos para detectar invasões. Ao final é comparado o desempenho de redes neurais e máquinas de vetores de suporte para a detecção de intrusões. Nesta dissertação, baseando-se neste artigo de [?] também serão avaliados os desempenhos de redes neurais e máquinas de vetores de suporte, mas para detecção de conflitos em políticas.

[?] apresenta uma discussão sobre a criação de perfis de autores multilíngues em mensagens SMS usando a abordagem de aprendizado de máquina com seleção estatística

de recursos que mostra, de forma didática, formas de tratar os hiperparâmetros de uma rede neural usando uma abordagem estatística. Este artigo forneceu suporte teórico para o sucessivo e repetitivo trabalho de ajuste dos hiperparâmetros tanto das redes neurais construídas nesta dissertação quanto dos outros classificadores.

Já o trabalho de [?] aborda o desenvolvimento e uso de uma rede neural probabilística construtiva (**CPNN** - *Constructive Probabilistic Neural Network*) na detecção de incidentes em rodovias, incluindo a construção e adaptação dos modelos. Esta CPNN foi estruturada com base no modelo Gaussiano de mistura e treinada por um algoritmo de ajuste dinâmico de decaimento (para correção dos erros). Os incidentes em rodovias, como colisões de veículos, podem ser extrapolados para um modelo de conflito entre agentes de um sistema. O modelo foi treinado e avaliado sobre um banco de dados de incidentes simulados em Singapura e adaptado para a rodovia I-880 na Califórnia sendo então investigada em ambientes on-line e off-line. Este artigo citado compara o desempenho do modelo CPNN e um modelo de rede neural probabilística básica (BPNN - Basic Probabilistic Neural Network).

O artigo de [?] estuda um componente de rede neural para um sistema de detecção de intrusão. O modelo *aprende* os hábitos que um usuário tem enquanto trabalha com o computador e emite avisos quando o comportamento atual não é consistente com os padrões *aprendidos* anteriormente. O modelo de de rede neural é usado para modelar o comportamento do usuário como uma característica componente para o sistema de detecção de intrusão.

O artigo de [?] trabalha com a identificação e a resolução de conflitos de voo com base em redes neurais. No artigo, o autor considera os problemas de detecção e resolução de conflitos no gerenciamento de tráfego aéreo (ATM - *Air Traffic Management*) sob a perspectiva da geometria computacional e fornece algoritmos para resolver esses problemas além de propor um método que pode rotear várias aeronaves, sem conflitos, através do espaço aéreo, usando um esquema de roteamento priorizado no espaço-tempo através do uso de conjunto de soluções ótimas por meio de uma rede neural.

Importante citar a patente assinada pelos inventores [?], "Sistema e método para mineração de dados e gerenciamento de políticas de segurança", tendo a McAfee como cessionária, que propõe um método para mineração de dados automatizada e criação de políticas de segurança em redes e sistemas.

Todos estes trabalhos foram base para o estudo, amadurecimento bibliográfico e aprofundamento teórico sobre o problema e as soluções propostas neste trabalho, principal-

mente aqueles relacionados a intrusões, detecção de conflitos aéreos e as soluções baseadas em aprendizado de máquina, com ênfase nas redes neurais e nas máquinas de vetores de suporte (SVM) pois não só serviram de inspiração como ofereceram estímulo para que as técnicas pudessem ser extrapoladas para o uso na detecção de conflitos em políticas, tema desta dissertação.

[?], [?] e [?] são os trabalhos-base no estudo da detecção de conflitos, da determinação do modelo das políticas utilizadas e analisadas além de serem a fonte de algumas das principais definições empregadas neste trabalho.

[?] estuda a verificação de conflitos entre múltiplas normas em sistemas multiagentes (SMA). As normas, nesta tese, são semelhantes às políticas, inclusive em suas definições, mas restritas a um contexto de agentes autônomos. Para a resolução de conflitos, o autor usa uma estratégia de aplicação de filtros para suavizar o custo computacional e utiliza transformação deôntica para análise de diversas normas ao mesmo tempo.

As hipóteses e problemas estudados por esta dissertação derivam do estudo seminal de [?] e são, em parte, uma complementação do mesmo usando algoritmos de mineração de dados e aprendizagem de máquina.

Capítulo 3

Experimentos/Resultados

A forma geral de como a estrutura dos experimentos foram realizados para o problema proposto em 1.1.1 é a seguinte:

1. Definição do problema;
2. Coleta de dados;
3. Pré-processamento dos dados;
4. Engenharia e seleção de atributos;
5. Modelagem: definição, configuração e arquitetura da rede (ou modelagem dos hiperparâmetros do SVM);
6. Treinamento (Aprendizagem da rede neural e do SVM);
7. Testes e validação do modelo;
8. Avaliação e ajuste do modelo;
9. Apresentação dos resultados;

Este modelo de método foi livremente adaptado de [?], [?] e [?].

3.0.1 Experimentos iniciais - arquivo com 68 políticas

Para os experimentos iniciais, um arquivo de políticas foi gerado randomicamente a partir do proposto em [?] e, de acordo com o exposto na seção 2.3.3. O arquivo gerado possui,

inicialmente, cerca de 68 políticas nomeadas (constituindo a *fase de seleção*[1] da Mineração de Dados) e da fase coleta de dado do método proposto anteriormente.

Este arquivo foi usado nos testes preliminares da hipótese descrita na seção 1.1.3 deste trabalho. Para este problema da detecção de conflitos diretos serão usadas técnicas de aprendizagem supervisionada.

Para tanto, ao arquivo com as políticas, no pré-processamento foi acrescentada uma coluna rotulando os conflitos da seguinte forma: **1**: *conflito direto* e **0**: *sem conflito*. Esta estratégia é a mesma utilizada no trabalho de [**?**] onde ele usa um modelo de matriz de controle de acesso usando operações lógicas `AND` e `OR` para identificação de conflitos.

A figura 3.1 demonstra o aspecto do arquivo das políticas geradas paraos experimentos deste trabalho. Na imagem, pode-se notar a classe (coluna) criada para guiar o aprendizado supervisionado dos algoritmos utilizados no estudo.

```
1  Politica,Acesso,Organizacao,Sujeito,Acao,Objeto,DataAtivacao,DataDesativacao,Conflito
2  Policy01, Permitted,UFAC,Secretario_de_Curso_Academico,Abertura, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
3  Policy02, Forbidden,UFAC, Sandra_Maria_Soares_da_Rocha,Abertura, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
4  Policy03, Permitted,UFAC, null, Solicitar, Produtos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
5  Policy04, Forbidden,UFAC, Secretario_de_Centros_Academicos, Solicitar, Produtos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
6  Policy05, Permitted,UFAC, null,Acessar,Almoxarifado, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
7  Policy06, Forbidden,UFAC, Jaider_Moreira_de_Almeida,Acessar,Almoxarifado, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
8  Policy07, Permitted,UFAC, Secretarias, Solicitar, Materiais, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
9  Policy08, Forbidden,UFAC, Secretario_de_Centros_Academicos, Solicitar, Materiais, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
10 Policy09, Permitted,UFAC, Grupo_IPTU, Gerar, Planilhas_de_Calculo, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
11 Policy10, Permitted,UFAC, Grupo_IPTU, Calcular, IPTU, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
12 Policy11, Forbidden,UFAC, Grupo_IPTU, Calcular, IPTU, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
13 Policy12, Permitted,UFAC, Socorro_Pontes,Acessar, Portal_do_Aluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
14 Policy13, Permitted,UFAC, Socorro_Pontes, Matricular,Aluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
15 Policy14, Permitted,UFAC, Jose_Rodrigues_Bardalles, Solicitacao, Central_de_Copias, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
16 Policy15, Permitted,UFAC, Jose_Rodrigues_Bardalles, Analise, Central_de_Copias_Analise, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
17 Policy16, Permitted,CCET, Jose_Rodrigues_Bardalles, Solicitacao, Central_de_Copias, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
18 Policy17, Permitted,CCSJA, Jose_Rodrigues_Bardalles, Analise, Central_de_Copias, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
19 Policy18, Permitted,UFAC, Grupo_Almoxarifado, Requisitar, Material, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
20 Policy19, Forbidden,UFAC, Grupo_Almoxarifado, Criar, Guia_de_Requisicao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
21 Policy20, Forbidden,UFAC, Grupo_Almoxarifado, Inserir, Guia_de_Requisicao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
22 Policy21, Permitted,UFAC, Usuario_Qualquer, Cadastrar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
23 Policy22, Forbidden,UFAC, Usuario_Qualquer, Nova, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
24 Policy23, Forbidden,UFAC, Usuario_Qualquer, Alterar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
25 Policy24, Forbidden,UFAC, Usuario_Qualquer, Cadastrar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
26 Policy25, Permitted,UFAC, Usuario_Qualquer, Nova, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
27 Policy26, Permitted,UFAC, Usuario_Qualquer, Alterar, Convencao, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
28 Policy27, Permitted,UFAC, Socorro_Pontes, Solicitar, MatriculaAluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
29 Policy28, Permitted,UFAC, Socorro_Pontes, Efetivar, MatriculaAluno, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
30 Policy29, Permitted,UFAC, Joao_Josino, LancarMedia, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
31 Policy30, Permitted,UFAC, Joao_Josino, VoltarLancamento, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
32 Policy31, Forbidden,UFAC, Joao_Josino, LancarMedia, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,1
33 Policy32, Permitted,UFAC, Joao_Josino, VoltarLancamento, Notas, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
34 Policy33, Permitted,UFAC, Secretario_de_Unidade_Administrativa, SolicitacaoAbertura, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
35 Policy34, Forbidden,UFAC, Secretario_de_Unidade_Administrativa, Cancelamento, Documentos, Tue Mar 24 16:23:59 GMT-05:00 2015, Thu Sep 24 16:23:59 GMT-05:00 2020,0
```

Figura 3.1: Aspecto do arquivo das políticas geradas para os experimentos

Fonte: compilação do autor

3.0.1.1 Recursos computacionais

Dois **ambientes computacionais** foram utilizados para as tarefas de mineração: um **notebook** Intel Core i5 vPro-8350U (8ª Geração de 64 bits com 1.70GHz e 8 GB de RAM, com SSD de 256 GB rodando Windows 10 Pro.

O outro ambiente foi um **Desktop** Intel Core i7 vPro-6700 de 8ª geração de 64 bits com 3.40 Ghz e 20 GB de RAM, com HD de 1 TB rodando o Windows 10 Pro.

[1] cf. seção 2.4 deste trabalho.

3.0.1.2 Pré-processamento

Ainda na fase de *pré-processamento*, a coluna 9 (Conflito) foi transformada do tipo de dado *Numérico para Nominal*. Para isso foi usado o softwate WEKA (descrito em [**?**]) aplicado o filtro *NumericToNominal* do software. Além disso, tanto o primeiro atributo quanto a data foram removidos, pois, dentro do escopo estudado neste trabalho, eles não influenciariam nos resultados finais.

3.0.1.3 Resultados - Arquivo com 68 políticas

Logo após, mais de 30 experimentos foram realizados de forma preliminar no *dataset* envolvendo os diversos algoritmos e muitos parâmetros alterados (a maioria com pequena ou nenhuma variação) para se chegar às técnicas finais que foram utilizadas nos posteriores experimentos e que serão explicitadas a seguir.

Utilizando-se a ferramenta WEKA ([**?**]) para as últimas fases do KDD (Mineração de Dados), foram utilizados alguns algoritmos de classificação que segundo [**?**] são alguns dos mais utilizados na Mineração de Dados. Para avaliar o desempenho definiu-se o método *cross-validatio*n com 10 folds. Em seguida suas acurácias foram comparadas.

A tabela 3.1 mostra o resultado destes experimentos:

Tabela 3.1: Acurácia dos classificadores

Classificador/Algoritmo	Acurácia
Multi Layer Perceptron	0.9705
SVM kernel linear	0.9705
Random Forest	0.9542
J48	0.9411
K* (K-star)	0.9411
Trees LMT	0.9117
IBk (KNN, com k =1)	0.8970
JRip	0.8970
Nayve Bayes	0.8674
Random Tree	0.7794

Fonte: Elaborada pelo autor mediante experimentos

As figuras 3.2 e 3.3 mostram os resultados das classificações do arquivo de políticas usando, respectivamente, os classificadores/algoritmos: *SVM* e o *MultiLayer Perceptron* (que foram os principais citados nos trabalhos relacionados, cf. descrito na seção 2.10). Cf. mostrado na tabela 3.1, os classificadores SVM e MultiLayerPerceptron ficaram empatados

em relação à acurácia. Nos experimentos posteriores, apenas os dois foram considerados.

```
=== Classifier model (full training set) ===

LibSVM wrapper, original code by Yasser EL-Manzalawy (= WLSVM)

Time taken to build model: 0 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          66               97.0588 %
Incorrectly Classified Instances         2                2.9412 %
Kappa statistic                          0.9344
Mean absolute error                      0.0294
Root mean squared error                  0.1715
Relative absolute error                  6.6784 %
Root relative squared error             36.5941 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,957    0,000    1,000      0,957    0,978      0,936    0,978     0,986     0
               1,000    0,043    0,917      1,000    0,957      0,936    0,978     0,917     1
Weighted Avg.  0,971    0,014    0,973      0,971    0,971      0,936    0,978     0,964

=== Confusion Matrix ===

  a  b   <-- classified as
 44  2 |  a = 0
  0 22 |  b = 1
```

Figura 3.2: Saída do software WEKA. Classificador: SVM

Fonte: compilação do autor

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          66               97.0588 %
Incorrectly Classified Instances         2                2.9412 %
Kappa statistic                          0.9344
Mean absolute error                      0.0574
Root mean squared error                  0.1664
Relative absolute error                 13.029  %
Root relative squared error             35.5106 %
Total Number of Instances               68

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0,957    0,000    1,000      0,957    0,978      0,936    0,996     0,998     0
               1,000    0,043    0,917      1,000    0,957      0,936    0,996     0,992     1
Weighted Avg.  0,971    0,014    0,973      0,971    0,971      0,936    0,996     0,996

=== Confusion Matrix ===

  a  b   <-- classified as
 44  2 |  a = 0
  0 22 |  b = 1
```

Figura 3.3: Saída do software WEKA. Classificador: *MultiLayer Perceptron*

Fonte: compilação do autor

Assim, com uma acurácia de 97,05% na classificação dos conflitos diretos, tanto o algoritmo Multilayer Perceptron (que implementa uma rede neural sigmoide multicamadas) quanto o SVM tiveram a maior acurácia, com 95,7% de *TP rate*(taxa de *True Positives* ou verdadeiros positivos) para a classe 0 (não há conflito) e, somente, 4,3% de *FP rate*(taxa de Falsos Positivos) para a classe 1 (quando há conflito direto). Nos experimentos realizados (assim como se esperava inicialmente na hipótese deste trabalho — baseado

em evidências da literatura), estes modelos algorítmicos foram os mais eficientes para a detecção de conflitos diretos.

A interface visual do WEKA é excelente para observar o comportamento inicial dos algoritmos, mas para as arquiteturas de redes neurais e seus diversos parâmetros, configurações e quantidade de camadas ocultas, de neurônios nas camadas ocultas, entre outras configurações levaram a outros experimentos. Para isso e com o objetivo de evitar o overfitting (como explicado na seção 2.6.0.3 novos procedimentos e ferramentas foram adotados).

3.0.2 Outros experimentos - arquivos com 139 e 281 políticas

Foi gerado, então, dois novos arquivos de políticas foram gerados randomicamente também a partir do proposto em [**?**] e, ainda conforme o exposto na seção 2.3.3. Os arquivos gerados agora, possui, 139 e 281 políticas nomeadas.

Os mesmos filtros anteriores foram aplicados e os resultados para os algoritmos (ainda usando a ferramenta Weka) foram os seguintes:

Para o MultiLayer Perceptron, a tabela 3.2 mostra como as acurácias ficaram com os arquivos com quantidades diferentes de políticas.

Tabela 3.2: Acurácia do MLP

Qtd. de Políticas	Acurácia
68	97.05
139	98.73
281	99.28

Fonte: Elaborada pelo autor mediante experimentos

Para o classificador SVM, a tabela 3.3 mostra como as acurácias ficaram com os arquivos com quantidades diferentes de políticas.

Mesmo com uma leve queda na acurácia do SVM no arquivo de 139 políticas, percebe-se que, quanto maior o número de políticas, maior é a acurácia do classificador e melhor é a classificação.

Tabela 3.3: Acurácia do SVM

Qtd. de Políticas	Acurácia
68	97.05
139	96.40
281	99.28

Fonte: Elaborada pelo autor mediante experimentos

3.0.3 Experimentos com Pandas, NumPy e sklearn

Entretanto, outros experimentos foram realizados utilizando as bibliotecas Pandas, NumPy e sklearn e o Notebook Jupyter. O notebook criado demonstra o treinamento de uma rede neural Rede Neural Multicamadas usando o classificador `MLPClassifier` da biblioteca `sklearn`

No arquivo, é demonstrado os experimentos na construção de uma arquitetura de uma rede neural com a estratégia de testes *holdout* sendo a divisão da base (split) em atributos previsores e classe com cerca de 75% da base sendo usada para treinamento da rede e 25% para teste.

Há também um pré-processamento importante focado em um tratamento dos dados categóricos do *dataset* e sua conversão para dados numéricos e divisão da base original em dois conjuntos de dados (previsores e classe).

Todos os atributos categóricos do dataset de políticas (o maior, com 281 instâncias) foram transformados para numéricos sendo um dicionário de dados construído.

Os passos foram os seguintes:

Primeiramente a base foi importada da seguinte forma, como demonstrado na figura 3.4 mostrando o aspecto inicial do dataset.

A partir daí, foram selecionados da base todas as linhas dos atributos do *dataset* que estavam com os tipos *object*. Foram procurados valores nulos e não foram encontrados. Foi usada uma técnica que transforma um atributo categórigo com k valores em uma representação numérica com valores inteiros para cada k valor. Há vantagens e desvantagens nessa abordagem. Elas estão discutidas em detalhes em [**?**].

A figura 3.5 demonstra como este procedimento foi realizado para o atributo que representa o acesso (Permitido, Proibido e Obrigatório).

```
In [4]: base.shape
Out[4]: (281, 6)

In [5]: base.head()
```

Out[5]:

	Acesso	Organizacao	Sujeito	Acao	Objeto	Conflito
0	Permitted	UFAC	Secretario_de_Curso_Academico	Abertura	Documentos	0
1	Forbidden	UFAC	Sandra_Maria_Soares_da_Rocha	Abertura	Documentos	0
2	Permitted	UFAC	Outro_Usuario	Solicitar	Produtos	0
3	Forbidden	UFAC	Secretario_de_Centros_Academicos	Solicitar	Produtos	0
4	Permitted	UFAC	Outro_Usuario	Acessar	Almoxarifado	0

Figura 3.4: Aspecto do dataset importado

Fonte: compilação do autor

```
In [15]: #Usando dicionarios para trocar os valores das colunas
         acesso = {"Acesso": {"Permitted":1, "Forbidden": 2, "Obliged": 3}}
         obj_df.replace(acesso, inplace=True)
         obj_df.head()
```

Out[15]:

	Acesso	Organizacao	Sujeito	Acao	Objeto
0	1	UFAC	Secretario_de_Curso_Academico	Abertura	Documentos
1	2	UFAC	Sandra_Maria_Soares_da_Rocha	Abertura	Documentos
2	1	UFAC	Outro_Usuario	Solicitar	Produtos
3	2	UFAC	Secretario_de_Centros_Academicos	Solicitar	Produtos
4	1	UFAC	Outro_Usuario	Acessar	Almoxarifado

Figura 3.5: Engenharia de atributos - dados categóricos textuais

Fonte: compilação do autor

Em seguida a base foi dividida em atributos previsores e a classe. Os previsores são as colunas que representam a política em si e a classe é a representação binária do conflito.

O aspecto dos atributos previsores ficou como o mostrado na imagem 3.6.

Já a classe ficou com o aspecto da figura 3.7

Em seguida os atributos foram transformados usando padronização para manter as variáveis na mesma ordem de grandeza. Na padronização, a média se iguala a 0 e o desvio-padrão se mantém em 1. A fórmula da padronização é a seguinte:

$$z = \frac{x - \mu}{\sigma} \tag{3.1}$$

onde μ é a média aritmética e σ é o desvio-padrão dos dados. Como todos os dados estão agora em formato numérico, é um passo importante.

O código para a padronização foi o abaixo:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
previsores_transformados = scaler.fit_transform(previsores)
```

```
In [39]: previsores
```

Out[39]:

	Acesso	Organizacao	Sujeito	Acao	Objeto
0	1	5	25	13	5
1	2	5	38	13	5
2	1	5	2	4	11
3	2	5	33	4	11
4	1	5	2	8	12
...	...	...	...	...	...
276	1	2	6	1	2
277	1	8	6	1	2
278	2	2	16	8	16
279	1	2	5	1	20
280	2	2	29	9	1

281 rows × 5 columns

Figura 3.6: Aspecto dos atributos previsores

Fonte: compilação do autor

```
In [40]: classe
Out[40]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,
                1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
                1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
                0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
                1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0,
                0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1], dtype=int64)
```

Figura 3.7: Aspecto do atributo classe

Fonte: compilação do autor

Assim os dados foram padronizados e ficaram na mesma escala. Foi gerado um novo conjunto de dados de treinamento com 75% do *dataset* (sendo escolhidos randomicamente). Deixando, assim, 25% da base para validação.

Logo em seguida, um modelo de Multi-Layer Perceptron foi criado usando o classificador `MLPClassifier` da biblioteca `sklearn` com os seguintes hiperparâmetros:

- $max\_iter = 10000$,
- $tol = 0.0000010$,
- $solver =' adam'$,

- $hidden\_layer\_sizes = (100)$,
- $shuffle = False$,
- $activation =' relu'$

Onde, respectivamente estão configuradas, o número máximo de épocas de treinamento (10000), a tolerância (0.0000010), a função de otimização de peso ('adam' refere-se a um otimizador estocástico baseado em gradiente descendente), a quantidade de neurônios na única camada oculta, se as amostras devem ser embaralhadas em cada iteração (marcado como falso) e a função de ativação da camada oculta (função de ativação ReLU).

A figura 3.8 mostra o código no notebook e as iterações finais onde o classificador indica que a função de perda no treinamento não melhorou mais do que a tolerância, $tol = 0,000001$ por 10 épocas consecutivas e, portanto, ele encerrou as iterações.

```
import random
random.seed(42)
from sklearn.neural_network import MLPClassifier
classificador = MLPClassifier(verbose = True,
                              max_iter=10000,
                              tol = 0.0000010,
                              solver = 'adam',
                              hidden_layer_sizes=(100),
                              shuffle=False,
                              activation='relu')
classificador.fit(previsores_treinamento, classe_treinamento)
```

```
Iteration 4143, loss = 0.01138099
Iteration 4144, loss = 0.01160489
Iteration 4145, loss = 0.01174808
Iteration 4146, loss = 0.01174473
Iteration 4147, loss = 0.01157006
Iteration 4148, loss = 0.01140509
Iteration 4149, loss = 0.01138531
Iteration 4150, loss = 0.01144552
Iteration 4151, loss = 0.01150888
Iteration 4152, loss = 0.01146306
Iteration 4153, loss = 0.01137697
Training loss did not improve more than tol=0.000001 for 10 consecutive epochs. Stopping.
```

Figura 3.8: Código do **MLPClassifier**

Fonte: compilação do autor

A figura 3.9 mostra algumas métricas de validação após o modelo criado realizar as predições na base de teste (25% do dataset ou 71 políticas). Nela, pode-se perceber a acurácia do classificador em 95.77%, classificando, conforme a matriz de confusão mostrada na mesma figura, somente 3 previsões incorretas. Levando-se em conta que o modelo se aprimora conforme a quantidade de instâncias aumenta, de acordo com o pressuposto no *teorema da aproximação universal* [**?**], pode se considerar este como um modelo satisfatório.

A figura 3.10 mostra a dimensionalidade dos dados após transformados pelo processo de padronização e dá uma visão geral de 3 atributos, ***sujeito, ação e objeto*** e o im-

```
previsoes = classificador.predict(previsores_teste)
```

```
from sklearn.metrics import confusion_matrix, accuracy_score
precisao = accuracy_score(classe_teste, previsoes)
matriz = confusion_matrix(classe_teste, previsoes)
print(matriz)
print('Precisão: {}%'.format(precisao * 100))
```

```
[[43  2]
 [ 1 25]]
Precisão: 95.77464788732394%
```

Figura 3.9: Validações para o modelo **MLPClassifier**

Fonte: compilação do autor

pacto na difusão dos conflitos no espaço após os atributos categóricos terem sido, todos, transformados para uma representação numérica.

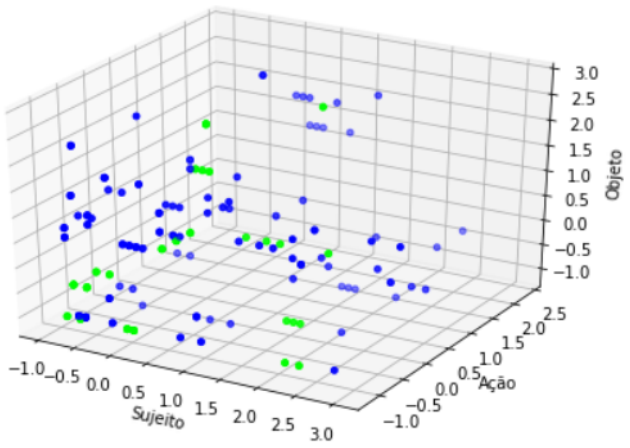


Figura 3.10: Dimensionalidade dos dados: atributos sujeito, ação e objeto

Fonte: compilação do autor

3.0.4 Experimentos com TensorFlow e Pytorch

De acordo com [?, p. 233], o *TensorFlow* é uma biblioteca de software para cálculo numérico de código aberto especialmente adequada e ajustada para o Aprendizado de Máquina em larga escala. Foi desenvolvido pela *Google Brain Team* para uso intensivo de redes neurais profundas. O código fonte foi disponibilizado em 2015 e está à disposição no link: `https://github.com/tensorflow/tensorflow`.

É possível usar computação paralela em várias CPU's ou GPU's além de suportar computação distribuída para que seja possível o treinamento e o uso de redes neurais em grandes conjuntos de treinamento dividindo os cálculos por centenas de servidores em um período de tempo razoável. [?].

Entre suas características importantes destacam-se: rodar em diversos sistemas operacionais (e inclusive na nuvem); diversas APIs públicas para criação, treinamento e avaliação de arquiteturas de diferentes tipos de redes neurais; diversas outras API's de alto nível foram construídas com base no TensorFlow como o *Keras* e o *Pretty Tensor*; implementações em C++ altamente eficientes para operações de Aprendizado de Máquina; nós de otimização avançados para procura por parâmetros que minimizem uma função de custo; usa extensões CUDA, uma API destinada a computação paralela, GPGPU, e computação heterogênea, criada pela Nvidia que dá acesso ao conjunto de instruções virtuais da GPU e a elementos de computação paralela. [**?**].

Os cálculos no *TensorFlow* são expressos como grafos de fluxo de dados , seu nome deriva das operações que as redes neurais realizam em arranjos de dados multidimensionais, chamados de "tensores"(generalização matemática de escalares, vetores e matrizes). [**?**]

Foi realizado, assim, no escopo deste trabalho experimentos com a base de dados de com 281 políticas já citada anteriormente. O notebook completo está disponível online no endereço: `https://bit.ly/33BmCzt`. Foi todo construído no ambiente Google Collab que é um serviço de nuvem gratuito hospedado pelo Google para incentivar a pesquisa de Aprendizado de Máquina e Inteligência Artificial, similar ao Jupyter Notebook, é uma lista de células que podem conter textos explicativos ou códigos executáveis e suas saídas. [**?**].

Neste modelo os hiperparâmetros principais foram ajustados como: tamanho do batch em 20 (analisa, computa e reajusta os pesos de 20 instâncias de cada vez, por época), o número de GPU's trabalhando em conjunto para 4, a taxa de aprendizado em 0.00001, o decaimento dos pesos em 0.000005 e o número de épocas padrão em 30 (apenas para testes inciais).

Para o processo de validação, usou-se o *holdout* com a separação em 70% das instâncias para o treinamento da rede neural e 25% para teste, predição e validação. Na figura 3.11 pode-se visualizar como o processo foi realizado,inclusive com a quantidade de instâncias em cada conjunto de dados (na linha 2 é feita a randomização do *dataset* original para evitar o overfitting e balancear a probabilidade da distribuição).

O pacote `torch.util.data` do PyTorch possui a classe abstrata `Dataset`. Ela permite que seja implementado o próprio dataset reescrevendo os métodos:

- `__init__(self)`: Define a lista de amostras do dataset

```
[78]  1 torch.manual_seed(1)
      2 indices = torch.randperm(len(dados_com_rotulos)).tolist()   # a
      3
      4 #separando os dados de teste e treino
      5 train_size = int(0.7*len(dados_com_rotulos))
      6 df_train   = dados_com_rotulos.iloc[indices[:train_size]]
      7 df_test    = dados_com_rotulos.iloc[indices[train_size:]]
      8
      9 print('Treino : ', len(df_train), ' - Teste: ', len(df_test))
     10 display(df_test.head())
     11
     12 df_train.to_csv('politicas_train.csv', index=False)
     13 df_test.to_csv('politicas_test.csv', index=False)
     14
     15 !ls

Treino :  196  - Teste:  85
```

Figura 3.11: Separação dos dados de teste e treino

Fonte: compilação do autor

- `__getitem__(self, idx)`: Carrega uma amostra, aplica as devidas transformações e retorna uma tupla (dado, rótulo)
- `__len__(self)`: Retorna a quantidade de amostras do dataset

Dessa forma, a figura 3.12 mostra a elaboração de uma classe chamada `Politicas` que implementa uma classe-filha que herda da superclasse, *Dataset* descrita acima.

```
1  class Politicas(Dataset):
2    def __init__(self, csv_path):
3      self.dados = pd.read_csv(csv_path).to_numpy()
4
5    def __getitem__(self, idx):
6
7      sample = self.dados[idx][0:5]    # [2:14] são as colunas do dataset
8      label  = self.dados[idx][-1:]     # [-1:]
9
10     # converte pra tensor
11     sample = torch.from_numpy(sample.astype(np.float32))
12     label  = torch.from_numpy(label.astype(np.float32))
13
14     return sample, label
15
16   def __len__(self):
17     return len(self.dados)
```

Figura 3.12: Implementação da classe Politicas

Fonte: compilação do autor

Em seguida, dois objetos DataLoader são criados, um para a base de treinamento e um para a base de teste. Em seguida, a Rede Neural Multicamadas é instanciada mediante a criação de uma classe chamada MLP que herda da classe `nn.Module` que representa um módulo genérico de uma rede neural.

A arquitetura da rede é configurada dentro da classe que a cria sendo: uma camada linear de entrada, duas camadas lineares ocultas com 32 neurônios em cada camada usando

a função de ativação ReLU e uma camada linear de saída com dois neurônios, representando os dois rótulos do atributo que é a classe, neste caso específico, o atributo binário **conflito** que será predito. É criada, no mesmo código da classe MLP, uma função que faz o avanço (fed-forward) das computações na rede e, ao final, é instanciada uma variável chamada `net` com as variáveis descritas: 5 atributos/neurônios na camada linear de entrada, 2 camadas ocultas com 32 neurônios cada e 2 camadas de saída representando as variáveis preditas. Na mesma linha que cria o objeto `net` é feito o *cast* da rede na GPU para que ela possa, ao ser treinada, fazer uso dos poderes computacionais em paralelo da API, CUDA. A figura 3.13 mostra o código descrito aqui.

```
class MLP(nn.Module):

  def __init__(self, input_size, hidden_size, out_size):
    super(MLP, self).__init__()

    self.features  = nn.Sequential(
                       nn.Linear(input_size, hidden_size),
                       nn.ReLU(),
                       nn.Linear(hidden_size, hidden_size),
                       nn.ReLU()
                     )
    self.out     = nn.Linear(hidden_size, out_size)

  def forward(self, X):

    feature = self.features(X)
    output  = self.out(feature)

    return output

input_size  = len(train_set[0][0])    # quantidade de atributos *importantes*
hidden_size = 32
out_size    = 2                       # variaveis que serão preditas

net = MLP(input_size, hidden_size, out_size).to(args['device']) #cast na GPU
```

Figura 3.13: Implementação da classe que modela a arquitetura da rede

Fonte: compilação do autor

Em seguida é definida uma *loss function*(função de perda ou de custo). É criado um critério que mede o erro médio quadrático (norma matricial ao quadrado) entre cada elemento na entrada x e o destino y. O otimizador utilizado é o Adam, um algoritmo para otimização estocástica descrito em [**?**] passando para o algoritmo os valores da taxa de aprendizado e do decaimento de pesos mostrado anteriormente nesta seção.

O fluxo de treinamento desta arquitetura de rede neural multicamadas proposta neste experimento segue o algoritmo iterativo:

- Iterar nas épocas
- Iterar nos batches (a quantidade de instâncias simultâneas)
- Cast dos dados no dispositivo de hardware (GPU)

- Forward na rede e cálculo da *loss function*
- Cálculo do gradiente e atualização dos pesos

Esse conjunto de passos é responsável pelo processo iterativo de otimização de uma rede. A validação, entretanto, é apenas a aplicação da rede em dados nunca antes vistos para estimar a qualidade do modelo no mundo real.

Três funções, portanto, são criadas, uma para modelar o treinamento da rede neural, uma para o teste e validação e outra que agrega as duas primeiras em uma só para executar o algoritmo descrito anteriormente. Então, para finalizar o experimento com o TensorFlow, o PyTorch, CUDA e o Google Collaboratory, foi realizado um treinamento com 500 épocas da rede neural explanada nesta seção e os resultados tanto do treino quando do teste e validação foram armazenados.

As figuras 3.14, 3.15 e 3.16 mostram as três funções citadas anteriormente

Como esclarecimento e interpretação visual foi construída, pois, a figura 3.17 com os dados de armazenados de treino e teste (média da *loss function* ou função de custo de cada iteração dentro da época) que mostra um comparativo das épocas de teste e treino da rede neural e a convergência de ambas, retratando a acurácia e a validade do modelo deste experimento.

```
def train(train_loader, net, epoch):
  net.train()

  epoch_loss = []
  for batch in train_loader:

    dado, rotulo = batch

    # Cast na GPU
    dado   = dado.to(args['device'])
    rotulo = rotulo.to(args['device'])

    # Forward
    pred = net(dado)
    loss = criterion(pred, rotulo)
    epoch_loss.append(loss.cpu().data)

    # Backward
    loss.backward()
    optimizer.step()

  epoch_loss = np.asarray(epoch_loss)

  print("Epoca %d, Loss: %.4f +\- %.4f" % (epoch, epoch_loss.mean(), epoch_loss.std()) )

  return epoch_loss.mean()
```

Figura 3.14: Implementação da função de treino da rede

Fonte: compilação do autor

3.0.5 Análise dos resultados

Os experimentos realizados na seção 3 corroboram a hipótese de que a detecção de conflitos pode ser transformada em um problema da tarefa de classificação da mineração de dados

```python
def test(test_loader, net, epoch):

  net.eval()
  with torch.no_grad():
    epoch_loss = []
    for batch in test_loader:

      dado, rotulo = batch

      # Cast na GPU
      dado   = dado.to(args['device'])
      rotulo = rotulo.to(args['device'])

      # Forward
      pred = net(dado)
      loss = criterion(pred, rotulo)
      epoch_loss.append(loss.cpu().data)

    epoch_loss = np.asarray(epoch_loss)

    print('********** Validate **********')
    print("Epoca %d, Loss: %.4f +\- %.4f" % (epoch, epoch_loss.mean(), epoch_loss.std()) )

    return epoch_loss.mean()
```

Figura 3.15: Implementação da função de teste da rede

Fonte: compilação do autor

```python
def forward(loader, net, epoch, mode):
  if mode == "train":
    net.train()
  else:
    net.eval()

  epoch_loss = []
  for batch in loader:
    dado, rotulo = batch

    # Cast na GPU
    dado   = dado.to(args['device'])
    rotulo = rotulo.to(args['device'])

    # Forward
    pred = net(dado)
    loss = criterion(pred, rotulo)
    epoch_loss.append(loss.cpu().data)

    if mode == "train":
      # Backward
      loss.backward()
      optimizer.step()

  epoch_loss = np.asarray(epoch_loss)
  print("Epoca %d, Loss: %.4f +\- %.4f" % (epoch, epoch_loss.mean(), epoch_loss.std()) )
```

Figura 3.16: Implementação da função que mescla o treino e o teste em uma só

Fonte: compilação do autor

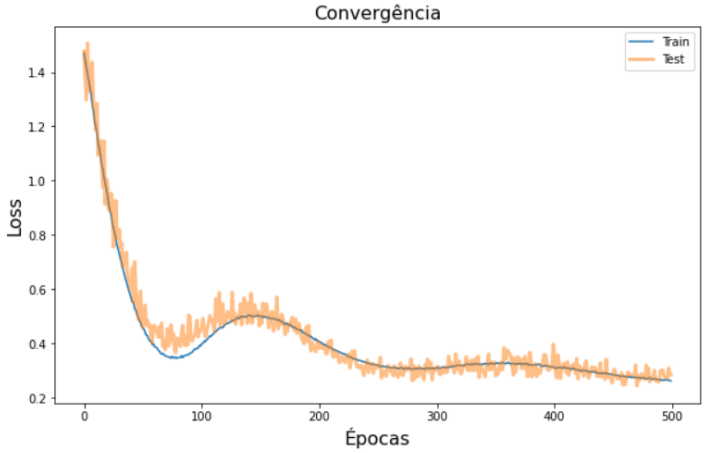


Figura 3.17: Convergência das épocas entre o treino e o teste da MLP

Fonte: compilação do autor

e do aprendizado de máquina. As vantagens sobre as outras abordagens relatadas na literatura são:

- A detecção do conflito pode ser realizada em tempo de execução, pois os modelos já estão treinados;
- As políticas são verificadas "em lote" e não em pares já que a análise em pares é um problema computacionalmente custoso da classe NP-Completo como demonstrado por [?];
- Ambas as técnicas analisadas neste trabalho mostraram-se eficazes com acurácias acima da proposta inicialmente (95%);
- os modelos de machine learning desenvolvidos nos experimentos podem ser aplicados em outros contextos, pois são suficientemente genéricos para tal;
- Ao analisar uma nova política, não é necessário "varrer" ou consultar todas as instâncias do *dataset* novamente já que os modelos já estão treinados;
- os modelos de machine learning tem a tendência a melhorarem a eficácia à medida que a quantidade de instâncias cresce

Todas os modelos e algoritmos demonstrados nos experimentos tiveram acurácia acima de 95% o que mostra que a detecção de conflitos em políticas (ou normas) pode ser colocada como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.

Capítulo 4

Cronograma e propostas para o texto final

Para a pesquisa que resultará na dissertação de mestrado os seguintes pontos serão levantados, estudados e melhor definidos em termos dos objetivos do trabalho:

1. Finalização da pesquisa teórica sobre o relacionamento de entidades;
2. Finalização da pesquisa teórica sobre funções de custo, SVM, descida do gradiente, topologias de redes neurais, suporte matemático aprofundado para o SVM;
3. estudo de outras redes, como Adaline, Madaline, redes de Kohonen, redes RBF e compará-las com as MLP e o SVM ;
4. Análise matemática profunda das outas função de ativação no classificador;
5. Análise teórica e implementação de outros otimizadores;
6. Realizar mais experimentos redes neurais e SVM;
7. Comparação final com outros classificadores (preferencialmente, geométricos, como o KNN e o SVM, avaliando suas acurácias e eficiência (para os conflitos indiretos).

Para isso, propõem-se o seguinte cronograma descrito na tabela 4.1

Tabela 4.1: Cronograma de finalização da dissertação

Atividades	**1ª sem ago**	**2ª sem ago**	**1ª sem set**	**2ª sem set**	**1ª sem out**	**2ª sem out**	**1ª sem nov**
Finalização da pesquisa teórica sobre o relacionamento de entidades	X	X					
Finalização da pesquisa teórica sobre funções de custo, SVM, descida do gradiente, topologias de redes neurais, suporte matemático aprofundado para o SVM;	X	X					
Estudo de outras redes, como Adaline, Madaline, redes de Kohonen, redes RBF e compará-las com as MLP e o SVM		X	X	X			
Análise matemática profunda das outras função de ativação no classificador			X	X			
Análise teórica e implementação de outros otimizadores			X	X	X		
Realizar mais experimentos redes neurais e SVM			X	X	X		
Comparação final com outros classificadores (preferencialmente, geométricos, como o KNN e o SVM, avaliando suas acurácias e eficiência (para os conflitos indiretos)				X	X		
Escrita, revisão e entrega de resultados iniciais					X	X	X
Texto final com resultados						X	X

Capítulo 5

Conclusões

- Esta pesquisa mostrou que é possível *converter a detecção de conflitos a um problema de classificação* especificamente, para os conflitos ***diretos***;
- O classificador mais acurado, nos experimentos, foi, como se imaginava pela hipótese, o *MultiLayer Perceptron* que é um classificador que usa *backpropagation* para aprender usando perceptron de várias camadas para classificar instâncias desconhecidas [**?**];
- Será realizada uma comparação entre o MLP e o SVM (e outros classificadores geométricos). Suas acurácias serão devidamente comparadas juntamente com a eficiência das soluções propostas.
- Todas os modelos e algoritmos demonstrados nos experimentos tiveram acurácia acima de 95% o que mostra, portanto, que a detecção de conflitos em políticas pode ser colocada como uma classe de problemas a serem resolvidos de forma eficiente por técnicas de aprendizagem de máquina.