UNIVERSIDADE DE CAXIAS DO SUL
CENTRO DE COMPUTAÇÃO E TECNOLOGIA DA INFORMAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO BARCAROL GUIMARÃES

Mineração de Dados baseada nos Sistemas Imunológicos: um estudo de caso na Detecção de Fraude

Carine Geltrudes Webber
Orientador

Caxias do Sul, fevereiro de 2021

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTAGENS

LISTA DE SIGLAS

CLI Command-line Interface

FP Taxa de falsos positivos

GNU GPL GNU General Public License

GUI Graphical User Interface

IA Inteligência Artificial

MD Mineração de Dados

TP Taxa de positivos verdadeiros

WEKA Waikato Environment for Knowledge Analysis

RESUMO

Os Sistemas Imunológicos Artificiais são um campo da Inteligência Artificial que se desenvolveu no início dos anos 90, e continua sendo objeto de pesquisa até hoje. Iniciando na área de segurança de computadores, os algoritmos baseados no sistema imunológico foram utilizados em diversas áreas da computação. O poder de abstração da modelagem dos elementos do sistema como componentes do sistema imunológico natural ajudaram no desenvolvimento desse sistemas. Utilizando um pacote de algoritmos imunológicos e a ferramenta WEKA, esse trabalho tem o objetivo de verificar como a utilização de Sistemas Imunológicos Artificiais influencia um sistema computacional, comparando-os com técnicas mais tradicionais da Mineração de Dados e da Inteligência Artificial.

Palavras-chave: Mineração de dados, inteligência artificial, sistemas imunológicos artificiais, fraude, detecção de fraude, segurança da informação.

Immune System based Data Mining: a case study on Fraud Detection

ABSTRACT

Artificial Immune Systems are a field of Artificial Intelligence that developed on the early 90's, and remains subject to researches event today. Beginning on computer security systems, algorithms based on the immune system have been used on many areas of computing. The power of abstraction the the design of these systems as components of the natural immune system helped their development. Using a package of immune algorithms and the WEKA environment, this study will verify how the application of Artificial Immune Systems influences a computational system, comparing them with more traditional techniques from Data Mining and Artificial Intelligence.

Keywords: data mining, artificial intelligence, artificial immune systems, fraud, fraud detection, information security.

1 INTRODUÇÃO

O crescente avanço na tecnologia de armazenamento de dados, principalmente em termos de velocidade e tamanho, permite a criação de bancos de dados complexos e detalhados. Com isso, o desenvolvimento de programas de computador que há anos atrás seriam computacionalmente inviáveis torna-se possível. Uma área fortemente influenciada por esse fator é a mineração de dados.

A Mineração de Dados é associada à ideia do aproveitamento de grandes bases de dados para auxiliar o profissional em alguma tarefa, através da apresentação de padrões e relações que seriam difíceis ou impossíveis de serem encontrados pelo observador humano. Ela foi definida como "análise de bases (geralmente extensas) de dados previamente coletados para estabelecer relações e apresentar a informação de forma mais clara e útil ao proprietário dos dados" (HAND, MANILLA, SMYTH, 2001, p. 1)[1] As atividades envolvidas na mineração de dados são:

a) **Determinar como os dados a serem usados serão representados**. As estruturas do problema devem ser codificadas para que possam ser utilizadas por algoritmos computacionais. Formas comuns de representação são cadeias de bits, números inteiros, números reais e valores categóricos.

b) **Decidir a função de aptidão**. Essa função é utilizada para quantificar a adequação de um modelo a um conjunto de dados, permitindo definir se um modelo é "melhor"que outro, ou seja, representa mais precisamente os elementos daquele conjunto. Essa função é altamente dependente da definição da representação dos dados, e é uma das partes mais importantes da definição do sistema, já que é ela que controla a convergência para uma solução ótima.

c) **Escolher um processo algorítmico para otimizar a função de comparação**. Através de um algoritmo específico, modelos diferentes (ou parâmetros diferentes para um mesmo tipo de modelo) são criados e testados utilizando a

[1] "Data mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner.".

função de aptidão. Esse processo é repetido até que a condição de término seja encontrada, geralmente após um determinado limite da função de aptidão ou um determinado número de iterações.

d) **Administrar o acesso aos dados de forma eficiente durante a execução dos algoritmos**. Grandes *datasets* geralmente excedem o tamanho dos dispositivos de armazenamento primário atuais. Dispositivos de armazenamento secundário ou terciário são utilizados, tornando o acesso a esses dados ineficiente. Um algoritmo corre o risco de tornar-se computacionalmente inviável caso o programador não considere esse fato.

Os exemplos da utilização da Mineração de Dados são inúmeros, estendendo-se virtualmente a qualquer atividade, de empresas comerciais à medicina e engenharia. A coleta e armazenamento de informações cresce constantemente, tornando a análise de dados impossível sem o uso de uma ferramenta automatizada. A mineração extrai padrões ou modelos de fontes de informação que seriam difíceis ou impossíveis de serem visualizadas apenas observando-se os dados. Isso tem motivado o desenvolvimento de algoritmos capazes de identificar padrões com mais detalhamento e significado.

Como área interdisciplinar, a mineração atrai estudiosos de diversas áreas da computação como, em especial, a Inteligência Artificial. A combinação de técnicas tradicionais de mineração com os algoritmos da Inteligência Artificial estende em muito o seu poder de análise. Toma-se a definição básica de Luger (LUGER, 2009, p. 1):

> "Inteligência Artificial (IA) pode ser definida como a área da ciência da computação que se dedica a automação de comportamento inteligente."[2]

Analisar informações, raciocinar e tirar conclusões são também objeto de estudo da Inteligência Artificial. Nessa área, alguns dos métodos mais aplicados são redes neurais e bayesianas, *clustering* e lógica nebulosa (*fuzzy*). De fato, as grandes bases de dados, que geralmente são objeto de atuação da mineração, constituem excelente campo de testes para os algoritmos da IA.

Uma área que têm recebido crescente atenção de estudos de mineração de dados e Inteligência Artificial é a de segurança da informação. O desenvolvimento de ferramentas de mineração é visto como um forte candidato à solução de problemas nessa área, já que elas são capazes de extrair modelos e comportamentos da base de dados que dificilmente seriam percebidos por um observador humano. Esses modelos podem ser usados para identificar padrões em novas instâncias, automatizando o

[2]"Artificial intelligence (AI) may be defined as the branch of computer science that is concerned with the automation of inteligent behaviour.".

processo de monitoramento. O ganho que um profissional tem ao utilizar uma ferramenta dessa natureza é enorme: a tarefa de análise de bases de dados extensas é repassada dele para o computador.

O reconhecimento de padrões e a capacidade de investigação de grandes bases de dados são os maiores desafios que profissionais da segurança da informação enfrentam. Nesse sentido, o objetivo é repassar à maquina cada vez mais o trabalho mecânico, para que o profissional possa se dedicar à análise crítica do material compilado. Ou ainda um algoritmo de mineração de dados pode analisar a grande quantidade de dados, deixando para os especialistas humanos a análise apenas daquelas identificadas como suspeitas.

Dentro da segurança da informação, escolheu-se um contexto mais específico para delimitar o escopo do trabalho: a detecção de fraude. Na legislação brasileira (SILVA, 1982, p. 324), a fraude é definida como "o *engano malicioso* ou a *ação astuciosa*, promovidos de *má fé*, para *ocultação da verdade* ou *fuga ao cumprimento do dever*". Em seu artigo *Measuring the impact of fraude in the UK*, LEVI, BURROWS (2002) definem a fraude como um mecanismo através do qual o fraudador ganha uma vantagem ilegal ou causa perdas ilegais. Os tipos mais proeminentes de fraude ocorrem na área médica, nos seguros (de vida, casa, automóveis, etc), cartões de crédito e telecomunicações.

A fraude não restringe-se aos casos em que há perdas financeiras diretas. Em um ambiente competitivo, a fraude pode ser um problema crítico se as medidas preventivas e planos de contingência não forem planejados e aplicados. Segundo a agência americana de investigação FBI (FBI, 2010), o custo das fraudes na área de seguros, excluindo-se os seguros de saúde, apenas nos Estados Unidos, é estimado em 40 bilhões de dólares por ano. O aumento médio nas apólices de uma família comum devido à fraude é calculado em 400 a 700 dólares. A *National Health Care Anti-Fraud Association* estima que 3% de todos os gastos médicos dos Estados Unidos são perdidos por causa de fraudes, o que representa uma quantia de 68 milhões de dólares anualmente.

A detecção de fraude é uma das etapas de um processo maior, comumente denominado de *controle de fraude*. Ela consiste na identificação de padrões e comportamentos associados à atividades irregulares, prevenindo que elas se concretizem (PHUA et al., 2010). A aplicação de técnicas de mineração de dados em aplicações desse tipo proporcionou espaço para o desenvolvimento de incontáveis trabalhos (PHUA et al., 2010). Essa é uma das principais aplicações da mineração de dados em aplicações corporativas e governamentais.

Dados recentes mostram que a detecção de fraude tradicional, o processo manual conduzido por um especialista humano é caro, devido ao custo da contratação desse especialista e ao grande volume de dados presente nos bancos de dados. Ou-

tro problema associado a essa prática é a incapacidade de detecção de padrões de fraude dispersos entre os dados, além do trabalho exaustivo e gasto de recursos que poderiam ser aplicados em áreas mais produtivas. O tempo necessário para que essa informação seja reunida, compilada e analisada também torna-se um fator proibitivo, já que dá ao fraudador uma janela de tempo muito grande para agir e até mesmo ocultar seus feitos. Sistemas de detecção de fraude têm o objetivo de automatizar e reduzir as partes manuais desse processo de verificação.

Com frequência, nas ciências, pesquisadores se voltam a áreas diferentes das suas próprias, com o objetivo de encontrar soluções para os seus problemas adaptando outras soluções. Um exemplo disso, na Inteligência Artificial, são os Sistemas Imunológicos Artificiais, inspirados no sistema imunológico natural. Conforme definiu Castro:

> "Sistemas Imunológicos Artificiais (SIA) são sistemas adaptativos, inspirados na teoria da imunidade e funções, princípios e modelos imunológicos observadas, que são aplicados à resolução de problemas."[3]

O sistema imunológico natural é um sistema robusto, complexo e adaptativo, que classifica as células do corpo como próprias e não-próprias. Isso é feito através de uma força de trabalho distribuída, capaz de agir de forma local ou global, usando uma rede de mensagens químicas para comunicar-se (AICKELIN, DASGUPTA, 2005).

Modelos computacionais foram criados para tentar adaptar o funcionamento do sistema imunológico natural a problemas da computação, e dá-se a esses modelos o nome de Sistemas Imunológicos Artificiais. A utilização desse tipo de sistema é relativamente nova: os primeiros trabalhos nessa área foram desenvolvidos a partir da metade dos anos 90. Mesmo assim é uma técnica que apresenta bons resultados e tem despertado o interesse de desenvolvedores de sistemas de segurança da informação. A modelagem desse tipo de sistema como um Sistema Imunológico Artificial oferece diversas vantagens:

a) O sistema imunológico natural protege o corpo de invasões externas. Essa metáfora pode ser facilmente estendida para problemas de segurança, facilitando a modelagem dos componentes do sistema.

b) A natureza distribuída do sistema imunológico facilita a implementação do sistema seguindo o paradigma dos sistemas distribuídos da computação.

[3] "Artificial Immune Systems (AIS) are adaptive systems, inspired by theoretical immunology and observed immune functions, principles and models, which are applied to problem solving.". CASTRO, L. N.; TIMMIS, J. 2002. *Artificial Immune Systems: A New Computational Intelligence Approach.* p. 57.

c) O sistema imunológico tem características que são muito importantes nos sistemas de segurança da informação. Ele contém uma memória, que reconhece agentes que o corpo já teve contato e apresenta um comportamento adaptativo, capaz de reconhecer e combater novas ameaças que ainda não haviam sido tratadas.

DASGUPTA, YU, NINO (2010) apresentam os trabalhos recentes na área de Sistemas Imunológicos Artificiais. A tabela 1.1 mostra as áreas onde esses sistemas foram aplicados.

Tabela 1.1: Aplicações recentes (DASGUPTA, YU, NINO, 2010)

Mineração de dados
Redes e segurança
Otimização
Automação e design
Detecção de anomalia
Bioinformática
Processamento de texto
Reconhecimento de padrões, clustering e classificação

Uma adição ainda mais recente foi a da Teoria do Perigo, proposta por Matzinger, em 1994. A principal revolução na Teoria do Perigo é a mudança na forma como as células tomam conhecimento e reagem, introduzindo a noção de perigo e tolerância. Essa mudança não é tanto na modelagem e representação dos dados, mas sim na decisão do que deve ser modelado, e o que deve gerar respostas do sistema (AICKELIN, DASGUPTA, 2005). A adição da Teoria do Perigo a um sistema, além das vantagens proporcionadas pela modelagem como Sistema Imunológico Artificial, restringe o domínio do problema, identificando o subconjunto de atributos mais importantes à detecção, e facilita a implementação em ambientes onde a definição de perigo muda com facilidade, características importantes em sistemas de segurança da informação.

2 DETECÇÃO DE FRAUDE

A fraude pode ter origem tanto interna quanto externa a uma organização. Por exemplo, uma empresa está sujeita a fraude por seus administradores (denominada de alto-nível) ou empregados que não sejam gestores (baixo-nível) (PHUA et al., 2010). Em um documento de 2012, a Associação de Investigadores Certificados de Fraude (Association of Certified Fraud Examiners, ACFE) definiu a fraude interna como a exploração ilegal dos recursos e bens de uma empresa, por um empregado, para enriquecimento próprio (ACFE, 2012).

Já na fraude externa, os seus autores dividem-se em três perfis: casual, criminal e crime organizado (PHUA et al., 2010). Criminosos casuais apresentam comportamento aleatório, transgredindo as leis quando têm oportunidade, tentação ou em períodos de dificuldades financeiras. Por outro lado, indivíduos ou grupos organizados são mais perigosos porque tentam esconder ou dissimular sua verdadeira identidade, além de evoluir seu *modus operandi* com o tempo, tentando burlar os sistemas de detecção e evitar a sua identificação. Assim, é importante levar-se em consideração essa constante interação entre os sistemas de detecção e os fraudadores profissionais. Essas categorias de fraudadores geralmente atuam em um setor específico: as fraudes internas e de seguro são mais frequentemente exploradas por criminosos comuns, enquanto fraudes de cartão de crédito e telecomunicações são vítimas de fraudadores profissionais.

O monitoramento de sistemas com o objetivo de encontrar comportamentos fraudulentos já existia muito antes da utilização dos sistemas computacionais tornarem-se ferramentas comuns. Antes havia um processo denominado auditoria: gerar, armazenar e revisar um registro cronológico de eventos de um sistema (BACE, 2000). Os principais objetivos dos sistemas de auditoria são identificar os usuários do sistema, impedir o uso impróprio, e auxiliar na reconstrução de eventos e na estimativa, quantificação e qualificação de danos.

O primeiro trabalho a considerar necessária a auditoria automática de sistemas foi ANDERSON (1972). Nesse trabalho, Anderson classifica os riscos e ameaças a sistemas, diferenciando fontes internas e externas, como na figura 2.1. Ele também

cita diversos objetivos para um sistema de auditoria:

a) Prover informações suficientes para que o problema possa ser localizado, mas que não exponham detalhes que possibilitem um ataque.

b) Obter dados de diversas fontes para otimizar o conteúdo do banco de dados.

c) Discernir uma atividade "normal" do sistema, para que se possa detectar abusos interno.

d) Levar em consideração as estratégias dos invasores no projeto do sistema.

A detecção de fraude é apenas uma das etapas de um sistema chamado de *controle de fraude*. Nesse contexto, a detecção automática ajuda a reduzir o trabalho manual de verificação das instâncias (PHUA et al., 2010). O objetivo principal desses sistemas é identificar padrões de transações suspeitas em meio às transações comuns de uma organização. O fraudador pode, por exemplo, contratar um seguro usando informações de outra pessoa ou informações falsas. O sistema procura detectar e impedir a fraude o mais cedo possível.

Tabela 2.1: Matriz de ameaças

	Uso não autorizado dos dados/programa	Uso autorizado dos dados/programa
Uso não autorizado do computador	Invasão externa	
Uso autorizado do computador	Invasão interna	Abuso de poder

Fonte: (ANDERSON, 1972).

Geralmente não é possível ter absoluta certeza sobre a legitimidade das transações de um negócio a partir dos dados disponíveis. Não seria possível verificar todas as entidades com as quais uma empresa mantém relações, que em algumas empresas podem ser milhares. Assim, não existe uma técnica infalível para a detecção de fraudes. Isso não significa que elas não possam ser detectadas. A melhor alternativa, na prática, é uma busca por possíveis evidências de fraude nos dados disponíveis. Métodos matemáticos e estatísticos são utilizados para comparar um banco de dados de transações existentes com as novas, com o objetivo de encontrar evidências de possíveis fraudes. Mesmo assim, geralmente há um processo de análise e revisão caso a caso por um especialista.

Ainda, segundo a pesquisa apresentada no mesmo trabalho, o motor analítico desse tipo de sistema pode ser composto de um ou mais métodos, tais como: Sistemas Imunológicos Artificiais, inteligência artificial, auditoria, bancos de dados, computação distribuída e paralela, econometria, sistemas especialistas, lógica nebulosa, algoritmos genéticos, aprendizagem de máquina, redes neurais, reconhecimento de padrões, estatística, visualização, entre outros.

Dois conceitos relacionados à detecção em geral são falsos positivos e falsos negativos. Falsos positivos são instâncias erroneamente classificadas, por exemplo, uma transação comum que é classificada como fraudulenta. Falsos negativos são o oposto: uma transação fraudulenta que é classificada como comum. O número de falsos positivos aumenta o trabalho desnecessário na fase de revisão, enquanto os falsos negativos reduzem a eficácia da detecção.

A redução dos falsos positivos é um dos objetivos principais de um sistema de detecção. Os falsos negativos, no entanto, são quase impossíveis de serem eliminados completamente, em qualquer sistema de detecção (MICHIE, SPIEGELHALTER, TAYLOR, 1994). Mesmo um sistema capaz de reconhecer sinais de fraudes existentes não é suficiente para um ambiente real. Fraudadores tentam constantemente superar os sistemas de detecção, evoluindo o seu *modus operandi* com o tempo, novos métodos são criados, novos fraudadores entram em atividade. Um sistema que almeje deter o maior número possível deve ser constantemente atualizado para se adaptar às mudanças de um ambiente tão dinâmico.

Dependendo do domínio da organização, para que um sistema possa prever fraudes com antecedência suficiente para que elas sejam evitadas, ele deve monitorar constantemente as novas transações em andamento. Esse fator reduz a utilização de sistemas que necessitam de um longo tempo de treinamento ou de análise. O ideal seria que ele estivesse em constante execução, analisando as transações conforme elas ocorrem. Para aplicações grandes ou descentralizadas, pode ser muito difícil conseguir isso sem afetar a performance geral das transações.

Para cada domínio, tipos diferentes de fraude podem existir, e mais de um tipo pode ocorrer simultaneamente, sem uma ordem definida.

2.1 Dados

Existem alguns conceitos sobre as bases de dados que são característicos da área de mineração de dados e aprendizagem de máquina. O número de instâncias na base é chamado de número de *amostras* e o número de atributos de cada instância é chamado de número de *características* (em inglês, *samples* e *features*).

Os atributos das instâncias em um banco de dados usado para detecção de fraude geralmente limitam-se a valores binários, numéricos, categóricos ou uma mistura

desses três. Os atributos específicos usados geralmente são semelhantes. Aplicações de seguro utilizam o histórico do cliente: tempo de contrato, histórico e total de pagamentos, lucro anual valor médio depositado em conta bancária. Para fraudes de crédito, utilizam-se informações sobre as transações: valor, data, localização geográfica, conta de destino, tempo de conta, etc. Fraudes em seguros de automóveis utilizam valores binários para atributos como acidente e tratamento hospitalar, além de dados do motorista, custo, tipo de ferimento, etc.

O número de instâncias positivas nas bases de dados de fraude é geralmente muito reduzido: as fraudes representam um percentual muito pequeno em relação ao número de transações legítimas de uma organização, geralmente menor do que 20%. Métodos de detecção de fraude nunca são perfeitos: deve existir um mecanismo para lidar com as fraudes que não são identificadas a tempo de serem impedidas.

A obtenção de bancos de dados reais para teste é difícil, já que empresas e organizações, por razões legais e competitivas não disponibilizam informações desse tipo. Assim, é difícil encontrar bases de dados públicos para que os testes sejam realizados. Outra desvantagem comumente encontrada nesses bancos de dados é o fato dos dados estarem alterados, para a preservação da confidencialidade dos clientes das empresas fornecedoras. Apesar dos dados ainda poderem ser utilizados normalmente em ambientes de teste, não é possível, como seria caso se tivesse acesso aos dados originais, derivar regras de classificação dos resultados dos testes. Por exemplo, observando os resultados, seria possível perceber que um valor alto em um atributo leva a instância ser classificada como fraudulenta na maioria dos casos. Em uma base de dados esse atributo teria uma descrição informativa, mas isso é perdido em uma base alterada, onde ele seria descrito por um identificador sem significado, como “atributo 5”.

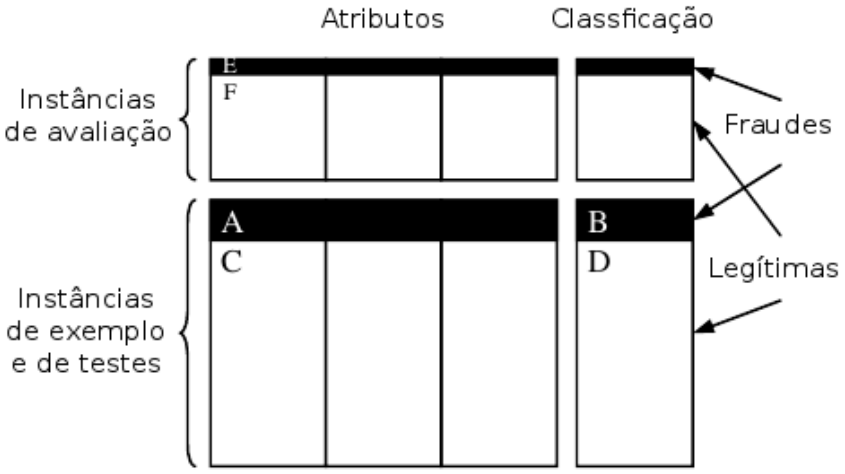
Uma alternativa é a criação de um banco de dados artificial, inspirado em dados reais. A eficácia dessa técnica é limitada pela capacidade do criador do banco em prever o maior número de casos possíveis, o que geralmente é muito inferior à variedade das situações reais. Mesmo assim, essa técnica é comumente empregada na fase de concepção e teste, devido à dificuldade de obtenção de dados.

2.2 Implementação

A maioria dos sistemas de detecção de fraude opera usando listas negras (*black-lists*) de dados (transações, contas, etc.), que são comparadas com as nova instâncias. Algumas utilizam regras fixas para a classificação. A figura 2.1 mostra a organização dos dados nesses sistemas. Uma parte do banco de dados é usada para o treinamento. Esses são os dados onde o sistema aprenderá os padrões e regras utilizados para a detecção. O restante das instâncias é usada para a avaliação do treinamento.

Também é comum que haja bancos de dados distintos para o treinamento e avaliação.

Figura 2.1: Dados para a análise



Fonte: (PHUA et al., 2010).

A maioria dos estudos relacionados à detecção de fraude considera a detecção de *outliers* como uma ferramenta principal de detecção (ARAL et al., 2011). Existem muitos métodos aplicados à detecção de fraude: auditoria, sistemas especialistas, lógica nebulosa (*fuzzy*), redes neurais, reconhecimento de padrões, árvores de decisão, regressão, etc (HUANG, TAWFIK, NAGAR, 2010). Considerando os dados divididos conforme a figura 2.1, as duas técnicas mais usadas são:

a) Dados para treinamento classificados (A + B + C + D) processados por um algoritmo supervisionado.

b) Instâncias legais (C) processadas por um algoritmo semi-supervisionado.

Algoritmos supervisionados examinam as instâncias previamente classificadas (A + B + C + D) para identificar matematicamente os padrões presentes nas classificadas como fraudulentas. Os algoritmos mais utilizados nessa categoria são as redes neurais. Outros algoritmos incluem máquinas de vetores de suporte (*support vector machine*, SVM), árvores de decisão e raciocínio baseado em casos. Para aumentar a eficácia dos métodos supervisionados, esses algoritmos podem ser aplicados em sequência. Também podem ser combinados resultados de bancos de dados distintos.

Algoritmos não-supervisionados atuam sobre dados não classificados (A + C + E + F), e o seu objetivo é agrupar os dados em padrões, para que estes sejam mais facilmente analisados, combinando a detecção humana e a computação da

máquina. Exemplos desses algoritmos são redes neurais não supervisionadas, análise de ligações (*link analysis*) e mineração de grafos (*graph mining*).

A combinação de dois ou mais algoritmos supervisionados, ou de algoritmos supervisionados e não-supervisionados, chamados de algoritmos híbridos, é uma grande área de pesquisa. Também são utilizados algoritmos supervisionados em bancos de dados que contêm apenas instâncias legais (C). Regras são geradas e testadas na base de dados, e aquelas que identificam padrões nesses dados são descartadas. Esses algoritmos são chamados de semi-supervisionados, porque apesar de não fazerem distinção entre dados legais e ilegais, os dados ainda necessitam ser classificados para que sejam utilizados no seu treinamento.

Autores como PHUA et al. fazem muitas críticas ao uso de dados previamente classificados para o treinamento dos sistemas (PHUA et al., 2010). A classificação atrasa o processo de detecção, aumenta o tempo de reação a novos tipos de fraudes e pode ser cara e difícil de se obter. Pode ainda ser incorreta, tendenciosa e expor dados sigilosos, dependendo do tipo de aplicação. Assim, instâncias de treinamento e avaliação (A + C + E + F, sem classificações) devem ser combinadas e processadas por um algoritmo não-supervisionado, detectando regras, pontuações ou anomalias visuais nos dados avaliados.

Um sistema que detecte e reporte uma fraude muito tempo depois de ela ter ocorrido permite que o fraudador consiga causar um dano substancial. Em geral, esse tempo de resposta de um sistema a partir do momento em que uma fraude é concretizada até a sua detecção é crucial para a eficácia do sistema em de fato proteger o ambiente em que é inserido.

Os sistemas de detecção podem ser divididos em dois tipos gerais, denominados *on-line* e *off-line*, enquanto alguns incorporam características dos dois modelos e outros têm um processo distinto para ambos, que interagem entre si para gerar o resultado final.

2.3 Considerações finais

A detecção de fraude é uma área que pode ser muito aprimorada através da utilização de sistemas computacionais. No entanto, a implementação de um sistema para detecção de fraude é complexa, devido às peculiaridades dessa área: os dados utilizados são sensíveis e esparsos, ocorrências de fraude são raras, comparadas ao grande volume de dados de transações legítimas nas bases de dados. Outro fator importante é que o sistema deve ser adaptável, já que os fraudadores aprimoram constantemente os seus métodos.

Além disso, a detecção, como parte do processo de controle de fraude, deve estar integrada aos outros processos, como a obtenção e segurança dos dados e a divulga-

ção dos resultados. Todos esses processos devem estar em constante funcionamento para que a detecção e a contenção da fraude sejam efetivas, evitando o máximo de danos possível. O sistema de detecção deve fornecer informações suficientes para que a instância possa ser analisada por um especialista para atestar se trata-se realmente de uma fraude. Também é importante que possa haver uma forma de otimização para que o sistema possa ser adaptado aos requisitos da detecção de fraude. Em especial, o número de falsos negativos deve ser o mínimo possível.

Busca-se constantemente aprimorar e adaptar as técnicas tradicionais da mineração de dados para a detecção de fraude. Uma área que oferece uma interessante inspiração é a biologia. Em especial, a imunologia trabalha com cenários muito semelhantes aos sistemas de detecção. Nos próximos dois capítulos, serão apresentados os algoritmos inspirados no sistema imunológico natural e os benefícios que eles podem trazer para os sistemas de detecção.

3 O SISTEMA IMUNOLÓGICO

Uma das funções principais do sistema imunológico é a proteção contra infecções (CAYZER, SULLIVAN, 2007). O reconhecimento de agentes infecciosos é feito através de fragmentos característicos de proteínas chamados de *antígenos*.

A arquitetura do sistema imunológico natural apresenta diversas camadas: um agente patogênico que consiga infiltrar-se no corpo enfrentará os sistemas imunológicos *inato* e *adaptativo*. Esses dois sistemas interrelacionados são constituídos de diversos tipos de moléculas e células, produzidas por órgãos e processos especializados para lidarem com o problema da discriminação do próprio e não-próprio. Essa discriminação é feita no nível mais baixo, com a interação entre as superfícies das células e moléculas e dos agentes patogênicos.

A primeira linha de defesa do corpo é o sistema imunológico inato, que existe desde o nascimento. Esse sistema não gera respostas específicas para cada tipo de invasor. Ele é constituído de proteções básicas como a pele, cílios, lágrimas, saliva e células como os macrófagos, que neutralizam os agentes externos em áreas de infecção.

No entanto, alguns agentes infecciosos são capazes de ultrapassar essas barreiras básicas. Nesse caso, entra em ação o sistema imunológico adaptativo, que recebe esse nome por prover respostas imunológicas específicas para cada tipo de antígeno ao qual é exposto. A resposta imunológica é gerada pela identificação de um agente externo, o que leva à ativação de certas células que removem o agente do corpo. Uma memória de exposições é mantida, e é utilizada quando o mesmo agente infecta o corpo novamente, mostrando um comportamento de aprendizagem na identificação (BROWNLEE, 2011).

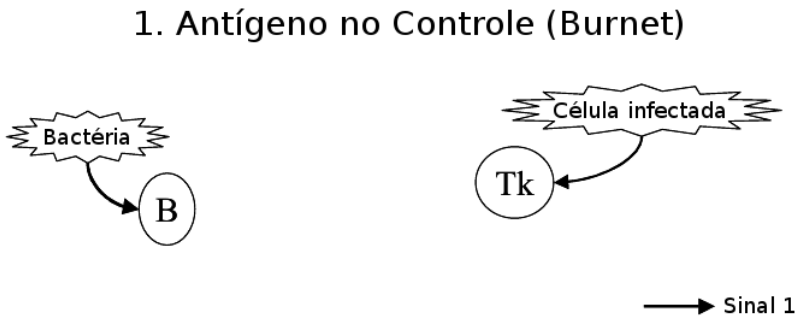
A primeira resposta, ou *resposta primária*, é lenta, levando até algumas semanas para remover a infecção. Caso o agente patogênico seja encontrado novamente, ocorre uma *resposta secundária*, onde é aplicado o que foi aprendido na resposta primária, tornando a resposta imunológica mais rápida e eficiente. A memória adquirida tem duração longa, geralmente provendo imunidade por toda a vida do hospedeiro (dois exemplos são varicela e sarampo).

O primeiro modelo do sistema imunológico foi o da distinção entre o próprio e o não-próprio, proposto por BURNET (1959). Ao longo do tempo, novos modelos foram criados, tentando resolver as questões que os outros modelos não explicavam. Entre esses, destacam-se o modelo do não-próprio infeccioso (JANEWAY, 1989), sendo aceito atualmente como o mais completo pelos imunologistas. As figuras 3.1 a 3.4 mostram a evolução dos modelos, e foram retiradas de AICKELIN, CAYZER (2002).

3.1 Discriminação do próprio e não-próprio

No modelo de discriminação do próprio e não próprio (Self Non-self Discrimination, SNSD) de Burnet, existia apenas um tipo de linfócito, o linfócito B, responsável por identificar os antígenos e produzir os respectivos anticorpos. A resposta imune adaptativa é gerada apenas pela identificação desses antígenos, sem qualquer mecanismo de controle.

Figura 3.1: SNSD: Antígeno no controle

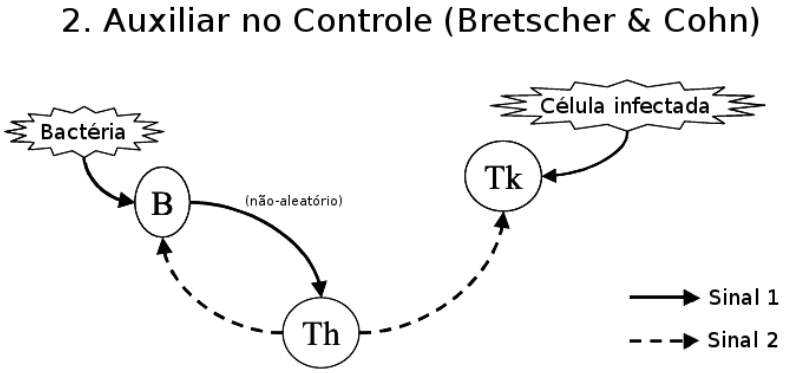


Fonte: AICKELIN, CAYZER (2002).

Em algum ponto no início da vida os linfócitos aprendem a diferenciar o próprio, células pertencentes ao corpo, do não-próprio, células estranhas ao corpo, que devem ser eliminadas. As células que geram respostas autoimunes (reações contra as células próprias) são removidas nesse ponto, restando apenas as capazes de identificar o não-próprio.

Mais tarde, Bretscher e Cohn introduziram uma segunda célula, o linfócito T auxiliar, ou T_h (T *helper*), que regulava a ativação dos linfócitos B (figura 3.2). O linfócito B apresenta o antígeno ao linfócito T e aguarda sua confirmação para iniciar a resposta. A introdução dessa célula tem o objetivo de evitar uma reação

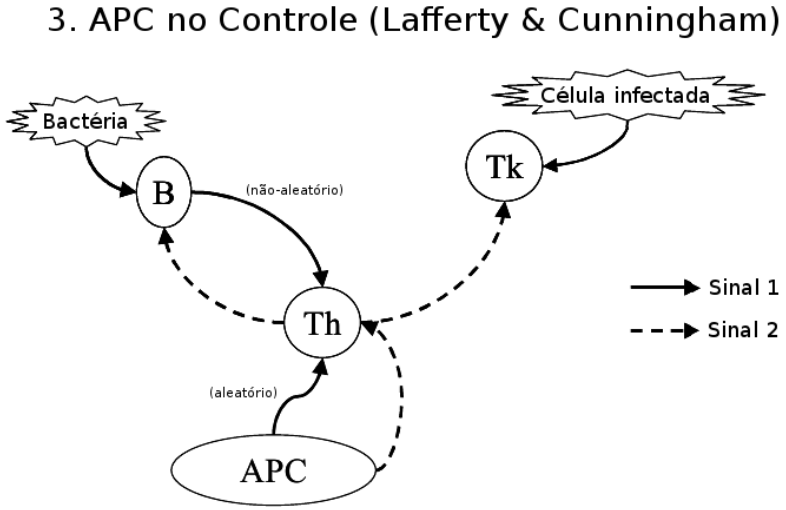
Figura 3.2: Auxiliar no controle



Fonte: AICKELIN, CAYZER (2002).

autoimune sem controle. Lafferty e Cunningham introduziram uma terceira célula, a Célula Apresentadora de Antígeno (APC, *Antigen Presenting Cell*), cuja função é decompor os antígenos e apresentá-los aos linfócitos T. Dessa forma, o funcionamento das células do modelo anterior agora depende da ativação do linfócito T pela APC (figura 3.3).

Figura 3.3: APC no controle

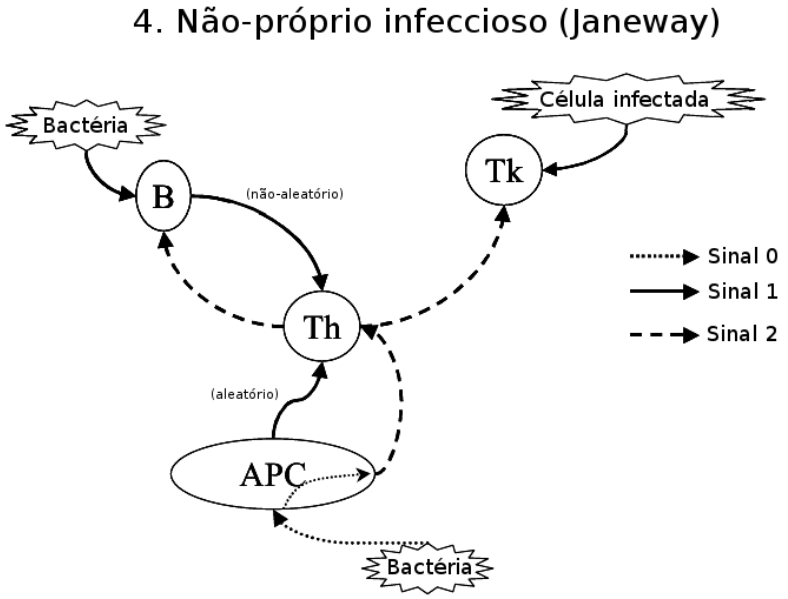


Fonte: AICKELIN, CAYZER (2002).

3.2 Não-próprio infeccioso

O modelo mais aceito na imunologia atualmente é o modelo não-próprio infeccioso (Infectious Non-Self, INS), que foi proposto por Janeway (JANEWAY, 1989). Nele, as APCs também têm de ser ativadas: elas só enviam o sinal dois aos linfócitos T quando tiverem reconhecido padrões patológicos no antígeno (figura 3.4). Assim, da mesma forma que no modelo anterior, o funcionamento de todo o sistema depende de um fator externo. No modelo anterior, a APC ativava o sistema, no modelo de Janeway, a APC tem que ser ativada através do reconhecimento de padrões no antígeno, e então ativar o sistema.

Figura 3.4: INS: Não-próprio infeccioso no controle



Fonte: AICKELIN, CAYZER (2002).

3.3 A Teoria do Perigo

Uma adição ainda mais recente foi a da Teoria do Perigo (MATZINGER, 1994). A teoria do próprio e não-próprio não condizia com os os experimentos realizados: não era observada resposta imune às bactérias no intestino ou no ar, haviam mudanças no conceito de próprio durante a vida de um indivíduo, por exemplo. Além disso, havia a própria problemática da definição de próprio e não próprio.

Matzinger sugeriu então uma teoria radicalmente diferente: não seria a distinção do próprio e do não-próprio a força que impulsiona o sistema imunológico, mas sim

o que ela caracteriza como "perigo", ou seja, qualquer coisa que cause estresse ou morte não-apoptética (não-natural) da célula. A Teoria do Perigo baseia-se em sinais gerados pelos tecidos danificados para diferenciar entre eventos malignos e benignos (CAYZER, SULLIVAN, 2007).

Embora não seja completa, essa teoria explica por que não existe resposta imune a células não-próprias inofensivas, mas existe resposta imune a células próprias danosas, como células cancerígenas. Esses são fenômenos que as teorias baseadas na discriminação do próprio e não-próprio não explicavam. Mais do que isso, a Teoria do Perigo muda a forma como se enxerga o sistema imunológico como um todo: um sistema responsável por manter o corpo em estado de equilíbrio. Dessa forma, a distinção explícita do próprio se torna desnecessária. Discriminação ainda existe, mas seu foco é o perigo, não mais o estranho.

Uma explicação detalhada do sistema imunológico, e seu funcionamento conforme a Teoria do Perigo, é apresentado na figura 3.5. A morte celular pode se dar de duas maneiras: via apoptose, a morte normal de uma célula, desencadeada pelo corpo deliberadamente; ou via necrose, uma morte inesperada e não planejada causada por alguma situação excepcional.

A teoria de Matzinger sugere que o sistema imunológico é ativado pela segunda forma de morte celular. Dessa forma, na ocasião de uma invasão, é gerada uma resposta imunológica no contexto do local onde a necrose ocorre. Apesar desses novos conceitos, a Teoria do Perigo ainda se baseia nos dois sinais de ativação da teoria de Janeway (citada acima) para a geração da resposta imunológica, evitando assim falsos positivos (reações autoimunes).

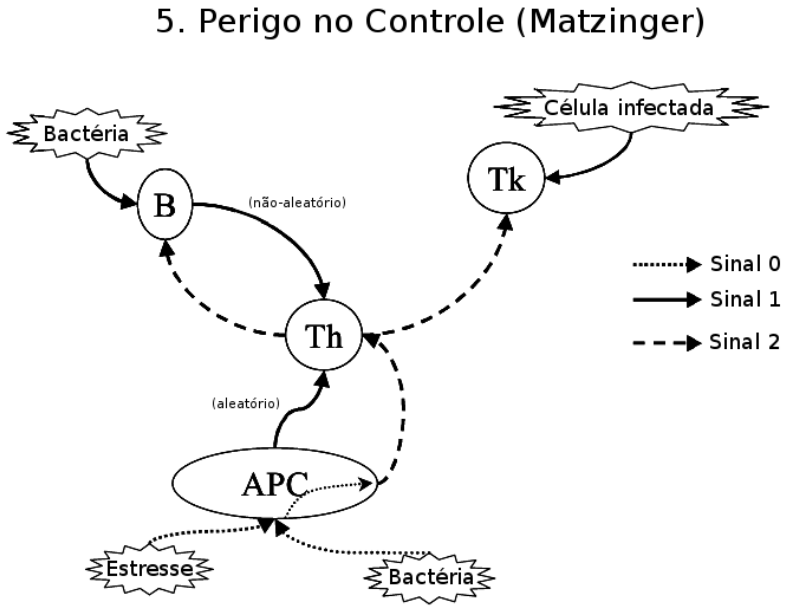
Os linfócitos B são os responsáveis pela identificação de antígenos através de receptores em sua superfície. Durante uma infecção, essas células se multiplicam e produzem anticorpos para eliminar os antígenos identificados. Elas são capazes de adaptar-se a virtualmente qualquer tipo de antígeno, gerando a resposta adequada. Um outro tipo de linfócito, os linfócitos T, quando ativados, podem exercer uma de duas funções: os linfócitos T citotóxicos (CTL) são responsáveis pela eliminação de células infectadas, enquanto que os linfócitos T auxiliares (T_h) são responsáveis pela ativação de outras células, entre elas os linfócitos B.

Além disso, alguns linfócitos T são mantidos como células de memória. Durante a resposta imunológica, os linfócitos B e T_h se multiplicam para combater os antígenos e são removidos quando a resposta termina. No entanto, alguns linfócitos T são mantidos, para que possam ser usados em futuras respostas ao mesmo tipo de antígeno. A combinação de linfócitos T virgens (sem um tipo de antígeno associado) e de memória permite que o sistema imunológico gere respostas tanto a novas ameaças quanto a ameaças recorrentes.

Existe ainda um outro tipo de linfócito, o T *killer* (T_k), que não tem receptores

antígeno-específicos, mas é capaz de reconhecer células infectadas e algumas células anormais. O objetivo dessas células é atuar como a primeira defesa contra infecções, e são muito importantes nos começo da vida.

Figura 3.5: Perigo no controle



Fonte: AICKELIN, CAYZER (2002).

O comportamento dos linfócitos T e B se baseia em três regras:

a) Linfócitos T e B entram em atividade ao receber os sinais um e dois, morrem ao receber apenas o sinal um e ignoram o sinal dois sem o recebimento do sinal um.

b) Linfócitos T aceitam o sinal dois apenas de APCs, enquanto os linfócitos B, apenas de linfócitos T ativos ou células de memória. O sinal um pode ter origem em qualquer célula.

c) Linfócitos T ativados não precisam do sinal dois para entrarem em ação. Após um período de tempo, eles voltam ao estado de repouso, necessitando novamente dos dois sinais.

Células Apresentadoras de Anítgeno são as células responsáveis por apresentar os antígenos às células T. Essas células podem ser os linfócitos B, macrófagos e as células dendríticas. Quando as células se encontram em estado de estresse ou morrem de forma não programada, enviam um sinal para as células APC, representado pela

seta Sinal 0 (figura 3.5. Isso desencadeia o envio do sinal dois para os linfócitos T_h. Enquanto isso, linfócitos B usam seus receptores para reconhecer antígenos nas suas redondezas, enviando o sinal um para os linfócitos T_h. É o par de sinais um (reconhecimento do antígeno) e dois (sinal enviado pelo linfócito T, ativado pelo sinal de perigo) que faz com que a reação imunológica tenha início.

Um conceito importante da Teoria do Perigo é a tolerância. Linfócitos B que reconhecem células próprias continuamente recebem o sinal um (identificação através dos receptores em sua superfície). No entanto, essas células não apresentam perigo (não causam danos a outras células), logo o sinal dois nunca será gerado. Sem receber o sinal dois de outras células, o linfócito será removido, conforme a primeira regra das três regras apresentadas acima. Dessa maneira, corpos estranhos mas que não causem dano (perigo) ao corpo não geram respostas imunes.

3.4 Seleção clonal

Ao contrário da seleção negativa e da Teoria do Perigo, que são centradas na detecção do não-próprio ou dos sinais de perigo, o foco da seleção clonal é o processo de adaptação das células B e T, para que possam identificar em remover os invasores. A capacidade do sistema imunológico de adaptar essas células para novos tipos de antígenos se deve a dois processos conhecidos como seleção clonal e maturação de afinidade por hipermutação (comumente denominados apenas "seleção clonal").

Segundo a teoria da seleção clonal, quando um linfócito identifica um antígeno, ele prolifera, criando milhares de cópias de si mesmo, e diferenciando-se em diferentes tipos de células: de plasma e de memória. Segundo Burnet (BURNET, 1959), essa clonagem ocorre de acordo com o grau de correspondência do anticorpo ao antígeno. Uma correspondência forte resulta em um grande número de clones, enquanto uma correspondência mais fraca resulta em um número menor de clones. Da mesma maneira, o grau de mutação dos clones é inversamente proporcional a esse grau de correspondência: quanto menor a correspondência entre um anticorpo e um antígeno, maior será o grau de mutação que seus clones sofrerão.

Células de plasma têm vida curta e produzem grandes quantidades de anticorpos moleculares. Células de memória vivem por um longo período, fazendo parte das respostas secundárias caso o antígeno venha a ser identificado novamente.

A clonagem de células B causa um aumento na afinidade do antígeno que provocou a clonagem, através de um processo denominado maturação de afinidade. Esse processo é composto de duas partes: hipermutação somática, que diversifica os anticorpos introduzindo mudanças aleatórias nas novas gerações; e um mecanismo de seleção, que garante que apenas os clones com maior afinidade sobrevivam. Assim, uma geração de células inclui a inicialização de soluções candidatas, seleção, clona-

gem, mutação, reseleção e substituição de população, semelhante a um algoritmo genético.

A parte essencial dessa teoria é que, quando um linfócito é clonado, sofre pequenas alterações em sua estrutura (hipermutação somática), que alteram seus receptores e sua capacidade de reconhecimento, assim como os anticorpos que são gerados por ele. Assim, essa teoria sugere que um repertório inicial de células imunes genéricas é capaz de evoluir para responder a mudanças no ambiente, sem ter informações detalhadas sobre os agentes que compõem esse ambiente.

3.5 Redes imunológicas

O foco das teorias vistas até agora era na ligação entre uma parte do anticorpo (conhecida como paratopo) com uma parte do antígeno (conhecida como epitopo). O modelo das redes imunológicas, desenvolvido por Jerne (JERNE, 1974) vai além, introduzindo anticorpos que também contém epitopos, que podem ligar-se a epitopos de outros anticorpos, formando uma rede.

Quando há uma ligação, a entidade da parte do epitopo é eliminada ou reprimida, e a da parte do paratopo é proliferada. Essa rede de estímulo e supressão que existe entre os anticorpos, e que afeta a concentração de cada tipo de anticorpo, forma um tipo de memória associativa.

3.6 Considerações finais

A imunologia como ciência existe desde o século 18, e o sistema imunológico ainda é uma grande fonte de pesquisa, em particular a natureza distribuída de seus mecanismos de memória, auto-tolerância e controle descentralizado. Baseando-se no funcionamento do sistema imunológico biológico, foram criados Sistemas Imunológicos Artificiais: algoritmos, estruturas e sistemas computacionais que inspiram-se nos modelos do sistema imunológico. Esses sistemas podem ser usados por imunologistas para explanação, experimentação ou previsão de situações que seriam difíceis ou impossíveis de serem reproduzidas em testes de laboratório (GARRETT, 2005). Essa técnica é conhecida como “imunologia computacional”.

No entanto, a maioria dos modelos de Sistemas Imunológicos Artificiais também são utilizados como abstrações dos processos imunológicos para o desenvolvimento de sistemas computacionais. O sistema imunológico é um sistema adaptativo complexo, que evoluiu nos seres vertebrados para protegê-los de invasores. O objetivo do estudo dos Sistemas Imunológicos Artificiais é aplicar na concepção de sistemas computacionais estruturas e funcionalidades inspiradas nos modelos biológicos. A importância dos Sistemas Imunológicos Artificiais é definida da seguinte forma

(AICKELIN, DASGUPTA, 2006):

> Do ponto de vista do processamento de informação, o sistema imunológico é um sistema adaptativo notavelmente paralelizado e distribuído com um mecanismo de controle (parcialmente) descentralizado. Ele usa extração de características, sinalização, aprendizagem, memória e recuperação associativa para solucionar tarefas de reconhecimento e de classificação. Em particular, ele aprende a reconhecer padrões relevantes, lembrar padrões que foram já foram vistos e usar análise combinatória para construir detectores de padrão eficientes. Além disso, o funcionamento do sistema como um todo é uma propriedade emergente de muitas interações locais. Essas notáveis habilidades de processamento de informações do sistema imune representam aspectos importantes no campo da computação. (AICKELIN, DASGUPTA, 2006)[1]

Durante a história da Inteligência Artificial, os modelos biológicos inspiraram muitos modelos de algoritmos imunológicos. Tipos diferentes de modelos podem ser utilizados em domínios diferentes de problemas e existem modelos que podem ser aplicados em mais de um domínio. Dentre estes modelos, destacam-se o algoritmos de seleção negativa, as redes imunológicas artificiais, o algoritmo de seleção clonal, os algoritmos genéticos e os baseados na Teoria do Perigo. As definições, características, funções e implementações desses sistemas são apresentadas no capítulo seguinte.

[1] "From an information-processing perspective, the immune system is a remarkable parallel and distributed adaptive system with (partial) decentralized control mechanism. It uses feature extraction, signaling, learning, memory, and associative retrieval to solve recognition and classification tasks. In particular, it learns to recognize relevant patterns, remember patterns that have been seen previously, and use combinatorics to construct pattern detectors efficiently. Also, the overall behavior of the system is an emergent property of many local interactions. These remarkable information-processing abilities of the immune system provide several important aspects in the field of computation."(AICKELIN, DASGUPTA, 2006)

4 SISTEMAS IMUNOLÓGICOS ARTIFICIAIS

O estudo sobre os Sistemas Imunológicos Artificiais (um subcampo da área de Inteligência Computacional) iniciou nos anos 90, baseado na proposição de aplicar modelos teóricos da Imunologia a problemas de aprendizagem de máquina e automação, procurando estudar e desenvolver abstrações computacionalmente interessantes do sistema imunológico natural. Como área, os Sistemas Imunológicos Artificiais são parte de uma área maior de algoritmos inspirados na biologia, junto com paradigmas como a computação genética e evolutiva (*Genetic and Evolutionary Computation*, GEC) e redes neurais artificiais (*Artificial Neural Networks*, ANN).

Conforme explicado anteriormente, os Sistemas Imunológico Artificiais podem ser divididos em dois tipos. No entanto, os sistemas que buscam simular fielmente as estruturas da Imunologia não são tão interessantes para a computação. A maioria dos estudos é baseada em modelos que se inspiram nos modelos naturais, mas que têm como objetivo final a construção de sistemas computacionais.

Os primeiros trabalhos basearam-se em paradigmas como o das Redes Imunológicas Artificiais, os Algoritmos Genéticos, Aprendizagem por Reforço e Sistemas de Classificação. Os primeiros trabalhos a formarem uma identidade própria desses algoritmos foram aqueles que relacionavam o sistema imunológico como uma analogia aos sistemas computacionais de proteção da informação, tais como FORREST et al. (1994) e FORREST, BEAUCHEMIN (1997).

Os algoritmos modernos são inspirados em três campos principais: seleção clonal, seleção negativa e redes imunológicas. Esse tipo de sistema é comummente aplicado a problemas de detecção de padrões, classificação, otimização, *clustering* e outros domínios de aprendizagem de máquina.

A última década mostrou um grande aumento no interesse pelos algoritmos baseados em sistemas imunológicos por serem uma boa fonte de inspiração para novas abordagens para solução de problemas complexos. O sistema imunológico natural oferece metáforas ricas pelo fato de ser um sistema altamente distribuído, adaptativo e auto-organizável, além de suas capacidades de aprendizado, memorização, extração de características e reconhecimento de padrões. A tabela 4.1 (inspirada

Tabela 4.1: Trabalhos recentes na área de Sistema Imunológico Natural

Algoritmos	Trabalhos
Seleção negativa	27
Células dendríticas	14
Redes imunológicas	10
Teoria do Perigo	8
Seleção clonal	5
Abordagens híbridas	18
Outros modelos	3

Fonte: DASGUPTA, YU, NINO (2010).

em DASGUPTA, YU, NINO (2010)) mostra uma lista dos algoritmos utilizados nos trabalhos recentes na área.

4.1 Algoritmos

Em DASGUPTA, YU, NINO (2010), os autores apresentam pesquisas recentes na área de Sistemas Imunológicos Artificiais, e mostram que as pesquisas recentes nessa área têm se focado em cinco algoritmos principais. Esse algoritmos são: (1) seleção negativa, (2) redes imunológicas artificiais, (3) seleção clonal, (4) Teoria do Perigo e (5) algoritmos de células dendríticas, apresentados nas seções seguintes.

4.1.1 Seleção negativa

Forrest et al (FORREST et al., 1994) publicou um artigo chamado "Self-Nonself Discrimination in a Computer"(Discriminação do Próprio-Não-Próprio em um Computador), que propunha um método chamado de *algoritmo de seleção negativa.* Esse algoritmo é baseado na capacidade de discriminação entre o próprio e o não-próprio no sistema imunológico adaptativo através do processo de seleção negativa na geração de células T no sistema imunológico. Foi projetado para aplicações de detecção de alteração, detecção de intrusão e outros problemas de reconhecimento de padrões e classificação binária, aplicado inicialmente a um problema de detecção de vírus de computador.

O princípio da técnica da seleção negativa é a modelagem do que é desconhecido como um complemento do que é conhecido. A seleção negativa ocorre no timo,

quando as células T (assim chamadas porque se desenvolvem no timo) que reagem a células próprias são eliminadas, mantendo apenas aquelas células com ligação fraca ou nula com as células próprias. No sistema imunológico natural, as precursoras das células T deslocam-se da medula óssea para o timo, onde ocorre o seu desenvolvimento. As células proliferam-se e diferenciam-se, através de mutação genética. A seleção negativa é aplicada sobre aquelas células T que são ativadas por células próprias. Essas células, no futuro, respondem aos diferentes tipos de agentes não-próprios invasores que precisam ser removidos da corrente sanguínea e linfática, tecidos, etc.

Na área dos Sistemas Imunológicos Artificiais, é comum referir-se à seleção negativa como "detecção de vírus". Na verdade, o conceito refere-se à detecção de qualquer mudança no "próprio", seja ela causada por vírus ou outro tipo de agente. É esse processo que foi abstraído para formar um algoritmo para detecção de mudanças em um conjunto de objetos.

Os algoritmos inspirados na seleção negativa são baseados nesse comportamento, e suas principais características são a representação negativa da informação, geração distribuída do conjunto de detectores e classificação dos dados em apenas uma classe. A representação dos dados é um dos principais aspectos desse algoritmo: geralmente são usados *strings* ou vetores de valores reais. Existe também uma regra de combinação particular de cada algoritmo, tipicamente baseada em distância ou alguma medida similar. Muitas variações desse algoritmo foram desenvolvidas, mas todas compartilham essas características básicas.

O modelo para um algoritmo baseado na seleção natural é:

a) Criar um conjunto de *strings* próprias, S.

b) Criar um conjunto de *strings* aleatórias, R_0.

c) Para cada $r_0 \in R_0$, criar um conjunto, R, dos elementos de r_0 que não são fortemente similares a qualquer $s \in S$. A similaridade é definida por uma função $m(r_0, s)$, tal que: (i) $m(r_0, s) \bowtie \theta$, (ii) $\bowtie$ é um operador (como $\geq$) que define se valores altos ou baixos de $m(r_0, s)$ indicam maior similaridade entre *strings* e (iii) θ define um limiar.

d) Para cada $r \in R$, garantir que nenhum $s \in S$ está acima (ou abaixo, dependendo de $\bowtie$) do limiar θ.

e) Retornar ao item d) enquanto a detecção de mudanças em S seja necessária.

O conjunto R é conhecido como conjunto de detectores, e é ele que será o responsável por detectar mudanças nos elementos próprios. Uma alteração nesse conjunto

faz com que os elementos aproximem-se do conjunto de elementos não-próprios, aumentado a similaridade com os elementos de R (GARRETT, 2005).

Aplicações típicas da seleção negativa são:

a) Detecção de alterações em códigos de máquina de programas causadas por vírus de computador, detecção de alterações em redes de sistemas distribuídos e detecção de intrusão em redes de computadores.

b) Detecção e diagnósticos de falhas, como em máquinas automatizadas, sensores, monitoramento de usinas químicas e nucleares e desenvolvimento de hardware tolerante a falhas.

4.1.2 Redes imunológicas artificiais

Redes imunológicas artificiais (*Artificial immune networks*, AINs) são inspiradas no modelo de redes imunológicas de Farmer et. al (FARMER, PACKARD, PERELSON, 1986), foram propostas por ISHIDA (1990) e foram redefinidos e reimplementados em TIMMIS, NEAL, HUNT (2000). Esses trabalhos foram nomeados conjuntamente como AINEs (*Artificial Immune NEtworks*, Rede Imunológicas Artificiais). Uma rede imunológica artificial consiste de um conjunto de células B e ligações entre essas células. Essas redes permitem ao sistema imunológico ter uma memória imunológica: as células estimulam ou reprimem umas às outras, atingindo uma memória estável. Sua utilização é ampla em campos como a mineração de dados e aprendizagem de máquina.

O sistema proposto por TIMMIS, NEAL, HUNT era composto de uma medula óssea, uma rede de células B e uma população de antígenos. A população de células B é inicializada aleatoriamente e, uma a uma, são inseridas em um ponto também aleatório na rede. Caso a célula seja capaz de ligar-se à população de antígenos, clones das células B são gerados, e aqueles com maior afinidade com as células da rede são mantidos. Os algoritmos de redes imunológicas artificiais incorporaram algumas ideias das teorias da seleção clonal: as células B nas AINE sofrem clonagem, mutação e seleção quando são estimuladas pela rede.

Aplicações típicas das redes imunológicas são:

a) Mineração de dados

b) Diagnósticos

c) Análise de clusters

d) Detecção de sequências de genes promotores.

4.1.3 Seleção clonal

Em 2000, Castro et al.(CASTRO, ZUBEN, 2002a) propôs o algoritmo de seleção clonal (CSA), mais tarde conhecido como CLONALG. Esse algoritmo é baseado nos princípios da imunidade adquirida de seleção clonal e maturação de afinidade, que são por sua vez baseados nos princípios da teoria de Darwin da seleção natural.

As etapas do CLONALG são:

a) Criar uma população inicial aleatória de anticorpos, P_0.

b) Repetir as etapas seguintes enquanto a condição de término não for atingida.

c) Escolher um subconjunto de P_t, F, contendo os anticorpos com maior aptidão de acordo com uma função de aptidão $f(ab_i)$.

d) Para cada $ab \in F$, criar um conjunto de clones, C_i, tal que $|C_i| = nc(ab_i)$. O conjunto de todos os clones, $C = \bigcup_i C_i$.

e) Aplicar mutação em cada clone $c \in C$ através de uma função $am(ab_i)$. Adicionar esses novos clones que sofreram mutação, C', ao conjunto P_t, gerando P'_t.

f) Selecionar um subconjunto de P'_t, F', contendo os anticorpos com maior aptidão.

g) Adicionar r células B geradas aleatoriamente a F', gerando uma nova população P''_t.

h) Reter apenas os $|P_t|$ melhores membros de P''_t, gerando P_{t+1}. Todas as outras células B são consideradas mortas.

O resultado de cada nova geração é que as células B tendem a aproximarem-se ao ponto ótimo do espaço de estados, em relação à geração anterior. A adição de membros aleatórios em cada geração faz com que o espaço seja melhor explorado.

Aplicações típicas da seleção clonal são:

a) Diversos tipos de reconhecimento de padrões.

b) Problema da coloração de grafos, reconhecimento de caracteres e alguns problemas NP-completo.

c) Diversos tipos de escalonamento.

d) Classificação de documentos.

e) Otimização de funções combinatoriais, uni-modais e multi-modais e o problema da inicialização dos centros em funções de base radial.

4.1.4 Teoria do Perigo

No primeiro artigo a propor a aplicação da Teoria do Perigo em um sistema computacional, AICKELIN, CAYZER (2002) apresentam como a Teoria do Perigo pode auxiliar na aplicação de Sistemas Imunológicos Artificiais em problemas complexos de detecção de anomalia, citando algumas analogias dos sistemas imunológicos presentes na Teoria do Perigo:

a) Uma APC é necessária para apresentar o sinal de perigo.

b) O "sinal de perigo"pode não ter relação nenhuma com perigo.

c) O sinal de perigo pode ser positivo (presença de sinal) ou negativo (ausência de sinal).

d) Uma medida de proximidade pode ser usada para simular uma zona de perigo.

e) Uma resposta imunológica não deve gerar novos sinais de perigo.

A contribuição da Teoria do Perigo para sistemas de detecção é a habilidade de focar-se apenas naqueles eventos que têm mais chance de serem nocivos, o que geralmente representa um subconjunto reduzido do conjunto não-próprio. A mudança da teoria do conceito de “próprio” e “não-próprio” para o de “perigo” pode parecer sem sentido, mas existem diferenças importantes: o conceito de próprio é relativo às próprias conhecidas, que podem não representar o conjunto próprio inteiro, além de poderem mudar com o passar do tempo e podem conter muitas características ou atributos.

O conceito de perigo, por outro lado, é baseado em eventos indesejados, possibilitando ao sistema de detecção utilizar apenas características ou atributos relacionados à situação de perigo. Representações dos sinais de perigo em um ambiente computacional podem ser positivas ou negativas (ou ambas). Sinais positivos podem representam a presença de eventos como grande utilização de memória ou disco, alterações frequentes em arquivos ou sinais incomuns do sistema operacional (como SIGABRT em sistemas UNIX). Sinais negativos podem representar a ausência de eventos como a falta de resposta de um servidor, ausência de um arquivo ou falta de espaço em disco ou memória.

O sistema imunológico reage aos antígenos dentro de uma zona de perigo centrada no local da origem do sinal, que pode ser modelada como uma medida de similaridade ou relação de causalidade. Aqueles anticorpos que combinam-se aos antígenos (sinal um) dentro de uma zona de perigo (sinal dois) proliferam, gerando células de memória.

4.1.5 Algoritmo de células dendríticas

O algoritmo de células dendríticas é inspirado pela Teoria do Perigo, mais especificamente na função das células dendríticas. As células dendríticas foram identificadas por STEINMAN, COHN (2002) e o seu principal papel é o de célula apresentadora de antígeno (APC). O objetivo do algoritmo de células dendríticas é preparar um conjunto de células dendríticas maduras que forneçam informações dependentes de contexto sobre como classificar como normais ou anômalos os padrões de entrada.

As células dendríticas encontram-se distribuídas em todos os tecidos e executam funções diferentes de acordo com o tecido em que atuam. Quando ainda estão em um estado de imaturidade viajam na corrente sanguínea, até que sofrem diferenciação e maturação quando propriamente estimuladas. Elas então migram para tecidos periféricos, onde apresentam antígenos para as células T, iniciando as respostas imunológicas. As células dendríticas são de três tipos principais:

Imaturas : coletam partes dos antígenos e sinais

Semi-maduras : decidem que os sinais locais não representam perigo e apresentam um sinal de tolerância às células T

Maduras : decidem que os sinais locais são de perigo e apresentam um sinal de reposta imune às células T

O primeiro algoritmo baseado no comportamento dessas células foi desenvolvido em GREENSMITH, AICKELIN, CAYZER (2005) e definido formalmente em GREENSMITH, AICKELIN, TWYCROSS (2006). Esse algoritmo combina sinais múltiplos para estimar o estado corrente do ambiente e amostrar outro antígeno assincronamente. Sua execução consiste de três etapas principais: inicialização, atualização e agregação. A inicialização configura diversos parâmetros. A fase de atualização é dividida em uma fase de atualização do tecido, onde o valor dos sinais é calculado de acordo com os dados de entrada, resultando nos sinais de entrada das células, e uma fase de atualização das células. O último estágio da execução é a agregação: os antígenos coletados são analisados e classificados.

O algoritmo aplica pesos pré-definidos aos sinais de entrada para produzir três sinais de saída: sinal de coestimulação, sinal de semi-maturação e sinal de maturação. Se a soma dos valores dos sinais de maturação for maior que a somas dos sinais de semi-maturação, a célula é diferenciada para um estado maduro e tem assinalado um valor de contexto. O algoritmo calcula a proporção de valores de contexto maduro de um antígeno para o total de antígenos, chamada de MCAV (*Mature Context Antigen Value*, Valor de Contexto Maduro do Antígeno), e aqueles antígenos cujo MCAV excede um limite pré-definido são classificados como anômalos.

4.1.6 AIRS

Esse algoritmo foi criado por Andrew Watkins (WATKINS, TIMMIS, BOGGESS, 2004) e tinha como objetivo implementar um Sistema Imunológico Artificial que utilizasse aprendizagem supervisionada. Na época de seu desenvolvimento, a área de algoritmos imunológicos supervisionados não havia sido explorada, apesar da pesquisa intensa na área de algoritmos imunológicos não-supervisionados [1].

Por esse motivo, esse foi o primeiro algoritmo supervisionado a implementar a maior parte das técnicas inspiradas nos sistemas imunológicos, como: modelagem dos dados como antígenos e anticorpos, expansão clonal dos linfócitos, mutação e maturação de afinidade e memória imunológica.

Na definição desse algoritmo também foi aplicado o formalismo do espaço de formas. Conforme apresentado na seção 4.3, esse é um formalismo bastante apropriado para a modelagem de Sistemas Imunológicos Artificiais, devido a forte semelhança entre o espaço de formas e as diferentes formas que os receptores dos linfócitos no sistema imunológico apresentam. Os antígenos identificados por esses receptores formam a região de reconhecimento ao redor de cada ponto no espaço de formas. Em um algoritmo de aprendizagem, isso representa a correspondência entre as instâncias de treinamento (antígenos) e as possíveis soluções (células B).

4.1.7 Immunos

Esse algoritmo foi apresentado por Jerome Carter (CARTER, 2000). O algoritmo Imunnos foi desenvolvido em oposição aos algoritmos que tentavam simular fielmente o comportamento do sistema imunológico. Segundo o autor, do ponto de vista da computação, construir um sistema que utilize equações cuidadosamente derivadas dos estudos teóricos a imunologia seria um feito notável, mas não ideal. Os elementos do sistema imunológico foram reduzidos ao nível mais fundamental para que fossem introduzidos no sistema.

O principal elemento na modelagem do algoritmo foi a utilização de pequenas e simples unidades de processamento conectadas em paralelo, como pode ser observado nos nodos das redes neurais e linfócitos do sistema imunológico. As principais metas da modelagem eram:

a) Representação interna simples de ser entendida,

b) Capacidade de generalização sobre os dados de entrada,

c) Tempos de treinamento previsíveis,

d) Aprendizagem *online*,

[1] De acordo com o autor, o único outro algoritmo imunológico supervisionado era o Immunos, apresentado na próxima seção.

e) Potencial para atuar como memória associativa,

f) Suporte a atributos contínuos e qualitativos,

g) Capacidade de aprendizagem e recuperação de um grande número de padrões,

h) Aprendizagem baseada em experiência e

i) Aprendizagem supervisionada.

4.2 Implementação

A modelagem de um problema como um Sistema Imunológico Artificial requer a definição de quatro componentes: codificação, medida de similaridade, seleção e mutação. De maneira geral, o funcionamento de um sistema baseado no modelo imunológico é: uma vez que a *codificação* e uma *medida de similaridade* tenham sido escolhidas, o sistema executa a *seleção* e *mutação*, ambas baseadas na medida de similaridade, até que o critério de parada seja satisfeito (AICKELIN, DASGUPTA, 2005). Essas quatro etapas são detalhadas nas próximas seções.

4.2.1 Codificação

A codificação é uma etapa essencial na definição de um Sistema Imunológico Artificial. Ela afeta toda modelagem do sistema, em especial a medida de similaridade, e pode ser responsável pelo seu sucesso ou falha. Os dois principais agentes do sistema imunológico, antígenos e anticorpos, são também os principais elementos que devem ser modelados. Eles são comumente representados da mesma forma.

Tabela 4.2: Mapeamento das estruturas do sistema imunológico

Sistema imunológico	SIA
Anticorpo	Vetor de características
Antígenos	Dados de treinamento
Expansão clonal	Reprodução de instâncias que combinam com os antígenos
Maturação de afinidade	Mutação aleatória, eliminação de anticorpos
Memória imunológica	Conjunto de anticorpos de memória
Espaço de formas	Tipo e valores possíveis dos vetores de características

Fonte: (WATKINS, TIMMIS, BOGGESS, 2004).

Um antígeno é uma parte do objetivo ou solução da aplicação, que pode ser único ou um parte de um conjunto-solução. Os anticorpos são o resto dos dados, normalmente a base de dados já existente, e geralmente existem em grande quantidade. Em uma aplicação de detecção de intrusão, por exemplo, o antígeno poderia ser o conjunto de dados que define um tráfego de dados, e os anticorpos, os conjuntos de dados que já foram identificados como transações legais. Na maioria dos problemas, a representação desses dois elementos é feita através de um vetor de números ou de características. Cada posição desse vetor representa uma característica da instância. Os tipos mais comuns de dados são números (inteiros ou reais), *strings* e valores binários.

A tabela 4.2 mostra as formas mais comuns do mapeamento das estruturas do sistema imunológico para as estruturas computacionais nos Sistemas Imunológicos Artificiais.

4.2.2 Medida de similaridade

A medida de de similaridade (também chamada de medida de afinidade, principalmente na área de Sistemas Imunológicos Artificiais) é usada para comparar duas instâncias e medir o quanto uma é semelhante à outra. É usada principalmente para agrupar instâncias em grupos e nas condições de término (o algoritmo termina quando o modelo descreve as instâncias com uma medida de similaridade satisfatória, que varia de acordo com a aplicação, o tempo de execução e o nível de similaridade necessário).

Uma função de similaridade simples é o número de *bits* que são idênticos nas duas sequências. Por exemplo, para os *strings* (00011) e (00000), a função retornaria 3. Essa função é oposta à distância de Hamming, que mede quantos *bits* devem ter seus valores trocados para tornar as sequências iguais. Em alguns problemas, essa medida não é suficiente, já que ela não tem a noção de continuidade. Uma alternativa é calcular o número de posições iguais contínuas, retornando o maior valor. Assim, o exemplo anterior também receberia o valor 3, mas as sequências (00000) e (01010) receberia 1. Essa diferenciação pode ter ser apropriada ou não, dependendo do problema. Para variáveis não-binárias, existem ainda mais possibilidades de medida de distância, como a distância Euclideana.

Para os problemas de mineração de dados, a aptidão geralmente significa correlação, e uma medida simples de correlação é o coeficiente de correlação de Pearson. A correlação de duas instâncias u e v é definida como:

$$r = \frac{\sum_{i=1}^{n}(u_i - \overline{u})(v_i - \overline{v})}{\sqrt{\sum_{i=1}^{n}(u_i - \overline{u})^2 \sum_{i=1}^{n}(v_i - \overline{v})^2}} \tag{4.1}$$

Aqui u e v são duas instâncias, n é o número de variáveis comuns a u e v, u_i é o valor da variável i na instância u e $\overline{u}$ é a média dos valores de todas as variáveis de u (não apenas das variáveis comuns a u e v). A média é modificada para que o valor 0 seja atribuído a instâncias sem nenhuma variável em comum. O resultado são valores de -1 a 1, onde 1 significa forte concordância e -1 forte discordância.

Para algumas aplicações, aptidão pode não ser benéfica e as instâncias que são mais semelhantes são na verdade descartadas. Esse processo é conhecido como seleção negativa. Ele é equivalente ao processo que acredita-se que ocorre durante a maturação dos linfócitos B, onde eles aprendem a não identificar os tecidos próprios, para que não se inicie uma reação autoimune.

A seleção negativa é muito aplicada a sistemas de segurança. Cria-se um ambiente seguro, formado apenas por componentes confiáveis. Na inicialização do sistema é gerado um grande número de detectores randômicos, que são aplicados aos dados gerados por esse ambiente. Aqueles que identificarem os dados legais são eliminados, restando ao final apenas aqueles que não identificarem. Esses detectores formarão o sistema de detecção que irá monitorar constantemente o ambiente. Caso algum detector identifique os dados no ambiente é gerado um alerta de "possível não-próprio".

A otimização da aptidão dos modelos em muitos casos não é realmente o objetivo do sistema se o objetivo é criar generalizações através de um subconjunto dos dados existentes (HAND, MANILLA, SMYTH, 2001). Um modelo com grande aptidão pode não se adaptar a novas instâncias que venham a ser analisadas pelo sistema.

4.2.3 Seleção

O papel da seleção é explicado a seguir, no contexto do funcionamento do sistema. Esse processo é descrito em pseudocódigo no trecho 4.1. Supondo um estado inicial onde sistema se encontra vazio, o antígeno então codificado (conforme a representação definida anteriormente) e adicionado ao sistema. Cada anticorpo é codificado e adicionado ao sistema, um a cada iteração. Os anticorpos iniciam com um determinado valor de concentração, que é análogo ao número de células presentes no Sistema Imunológico Artificial. Esse valor é constantemente decrementado conforme o tempo passa, representando a morte natural de parte das células.

Listagem 4.1: Pseudo código de um Sistema Imunológico Artificial

```
1  Inicializar o sistema
2  Codificar o antígeno Ag
3  ENQUANTO (sistema não cheio) E (ainda existem anticorpos) FACA
4      Adicionar próximo anticorpo Ab
5      Calcular a medida de similaridade entre Ag e Ab
6      ENQUANTO (sistema cheio) E (sistema não estabilizado) FACA
```

```
 7          Reduzir a concentração de todos os Abs
 8          Estimular os Abs conforme a medida de similaridade
 9      FIMENQUANTO
10  FIMENQUANTO
```

Fonte: (AICKELIN, DASGUPTA, 2005)

Anticorpos cuja concentração ultrapassa um certo limiar mínimo são removidos do sistema. Um anticorpo aumenta sua concentração de acordo com a sua similaridade com o antígeno. Quanto maior a similaridade, mais sua concentração aumenta, em um processo similar à expansão clonal que ocorre com as células imunológicas quando identificam um antígeno. Após todos os antígenos serem adicionados ao sistema, começa um processo iterativo de decremento de concentração e clonagem, até que o sistema entre em equilíbrio: não ocorram remoções por um determinado período de tempo. Assim, conforme o número de iterações do processo de seleção, apenas aqueles antígenos que apresentarem uma maior taxa de similaridade com o antígeno são mantidos no sistema.

4.2.4 Mutação

A mutação utilizada nos sistemas imunológicos é similar àquela encontrada em Algoritmos Genéticos. *Strings* binárias têm seus dígitos invertidos, valores reais são alterados aleatoriamente e os outros tipos são trocados de posição. Além disso, geralmente usa-se a mutação somática, inspirada na reprodução de células do sistema imunológico, onde a mutação é mais severa conforme o grau de similaridade entre o anticorpo e o antígeno aumenta (ou diminui, caso esteja-se aplicando a seleção positiva).

A estratégia de mutação deve ser planejada com cuidado, porque nem todas as técnicas funcionam para qualquer tipo de dados. Um exemplo de um sistema de recomendação de filmes é apresentado em AICKELIN, DASGUPTA (2005), onde uma base de dados é utilizada para recomendar filmes aos usuários, analisando usuários que recomendaram filmes semelhantes àqueles recomendados por ele. Nesse caso, não faria sentido aplicar qualquer tipo de mutação: ao fim, os anticorpos convergiriam para o próprio usuário.

Mesmo assim, a mutação pode ser muito útil quando aplicada a alguns problemas, como detecção de intrusão e mineração de dados (CASTRO, ZUBEN, 2002b).

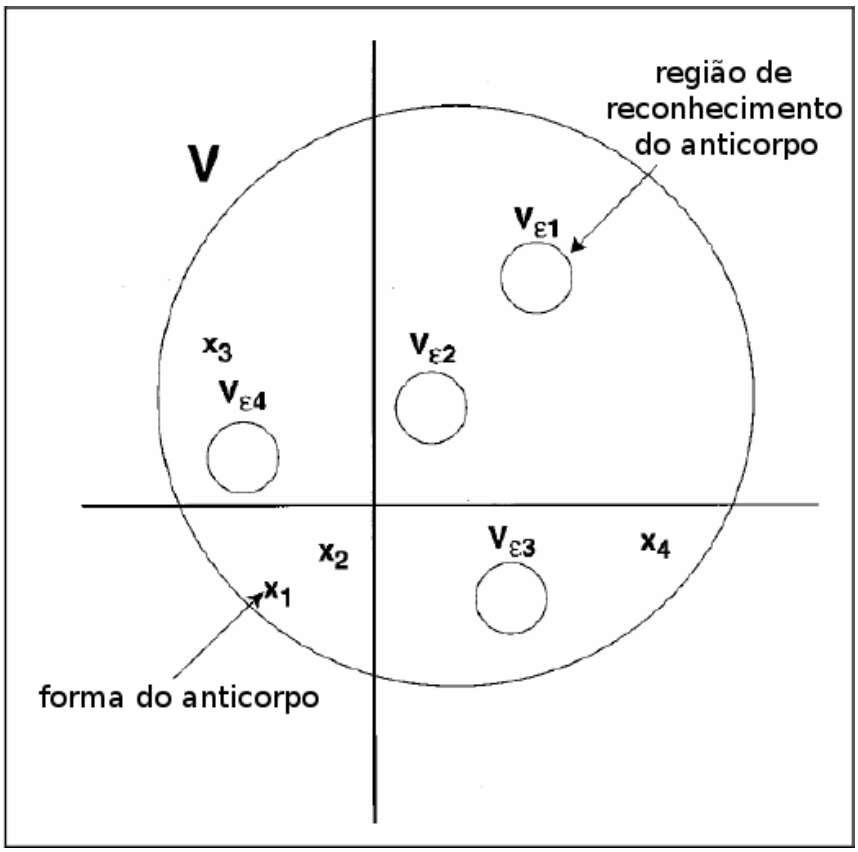
4.3 Espaço de formas

Os formalismos do espaço de formas e topologia de afinidade são dois paradigmas geométricos populares, originados na imunologia teórica e computacional,

muito utilizados em aplicações recentes do campo de Sistemas Imunológicos Artificiais (BROWNLEE, 2007). PERELSON, OSTER apresentaram o formalismo do espaço de formas em PERELSON, OSTER (1979), onde os autores fizeram uma investigação teórica das diferentes formas e capacidades de reconhecimento dos anticorpos. As ideias apresentadas nesse trabalho influenciaram muitos trabalhos no fim dos anos 80 e início dos anos 90, principalmente em trabalhos teóricos sobre seleção clonal e redes imunológicas.

Dados um anticorpo *ab* e um antígeno *ag* é criado um vetor onde as características relacionadas à ligação entre esses dois são discretizadas e transformadas em valores reais. Os parâmetros no trabalho original representavam características físicas, como a estrutura geométrica, carga eletrostática, entre outras. Assim, a interação *ab*-*ag* pode ser definida como um ponto em um espaço vetorial Euclidiano S de n dimensões, onde n é o número de características que compõem o vetor. Uma representação do espaço de estados, adaptada de BROWNLEE (2007) é apresentada na Figura 4.1.

Figura 4.1: Diagrama do formalismo do espaço de estados



Fonte: (BROWNLEE, 2007).

O espaço de formas é um hipercubo de volume V. Cada anticorpo é definido como uma área de reconhecimento nesse espaço (ϵ). Na definição original, essas áreas de reconhecimento eram funções Gaussianas. A distância euclidiana entre *ab* e *ag* é considerada como a 'afinidade' ou 'medida de complementaridade'. Para isso, a natureza complementar dos anticorpos e antígenos é ignorada: uma ligação perfeita é considerada quando $ab == ag$, ao contrário de $ab \mathrel{!}= ag$. Anticorpos possuem uma hiper-região de reconhecimento, onde antígenos podem gerar uma resposta imune.

Uma crítica ao formalismo do espaço de formas foi feita em CARNEIRO, STEWART (1994), destacando principalmente a simplicidade da abstração do espaço de estados e as limitações das funções de afinidade simples.

4.4 Topologia de afinidade

O objetivo de um anticorpo, no sistema imunológico, é maximizar a sua afinidade com um antígeno específico. Assim, a maturação de um anticorpo pode ser considerada também geometricamente como uma movimentação na superfície da função de afinidade para um antígeno específico (BROWNLEE, 2007).

Essa superfície teórica é geralmente considerada como contínua e possui diversos ótimos locais, ou seja, um ponto com maior afinidade que seus vizinhos, mas que não é o ponto global com maior afinidade. Esse formalismo é útil na definição e formalização do processo de maturação por hipermutação dos anticorpos, parte principal dos algoritmos baseados na teoria da seleção clonal.

Os formalismos do espaço de formas e topologia de afinidades são a base da definição do espaço de formas binário usado na seleção negativa e nas redes imunológicas, das regiões e limiares de detecção nos algoritmos de mineração de dados e classificação, e da movimentação na topologia das funções de custo nos algoritmos de otimização.

A interpretação geométrica genérica proporcionada por esses dois formalismos constitui um *framework* para a investigação de princípios imunológicos, não limitada aos problemas de reconhecimento de padrões.

4.5 Comparação dos algoritmos

Garrett faz um estudo detalhado de comparação entre os principais algoritmos imunológicos (GARRETT, 2005). Nesse trabalho, o autor define o método de comparação entre esses algoritmos em dois termos: distintividade e eficiência. A ideia por trás dessa comparação é que um algoritmo pode ser distinto de outros, mas ineficiente; como também pode ser altamente efetivo, mas redutível a outro método. No entanto, caso um método seja tanto distinto quanto efetivo, ele é realmente

considerado computacionalmente útil.

Para que um algoritmo seja considerado distinto, é avaliado se seus símbolos e expressões são únicos, assim como os processos aplicados sobre estes dois, ou se podem ser transformados em símbolos, expressões ou processos de outros métodos, sem afetar a dinâmica do algoritmo. Por outro lado, para que seja considerado eficiente, é avaliado se ele obtém os resultados através de meios únicos, apresenta resultados melhores ou obtém um conjunto de resultados mais rapidamente que métodos existentes. Nessa comparação, é importante que um mesmo tipo de experimento seja executado em todos os casos, e que a mesma métrica seja aplicada na avaliação dos resultados.

4.5.1 Seleção negativa

Apesar de ser uma das mais importantes abstrações do sistema imunológico, análises de algoritmos baseados na seleção negativa demonstraram que eles sofrem de problemas de escalabilidade. Considerando o conjunto de cadeias de bits consideradas próprias (S), o número de cadeias aleatórias (R_0) aumenta exponencialmente com o crescimento de S. Soma-se a isso o fato de que pode haver sobreposição nas partes do espaço de estados coberta por cada detector, enquanto outras partes podem não ser cobertas. Ou ainda podem haver áreas do espaço não-próprio que não podem ser cobertas por um detector sem que ele cubra também uma área própria.

Já foi questionada a tentativa de cobrir, aleatoriamente, um espaço não-próprio infinito utilizando detectores de tamanho finito. Como autores como Dasgupta, Forrest e Nino apontam, esse tipo de modelagem permite que múltiplos detectores identifiquem um elemento não-próprio por aproximação, além de permitir a criação de detectores sem que se tenha conhecimento do espaço de estados. De fato, caso o conjunto próprio fosse desconhecido, e existisse apenas uma função que, dado um elemento, determinasse se ele pertence ou não ao conjunto próprio, a seleção negativa seria o único método, entre os discutidos, que poderia ser aplicado.

Conforme demonstrou D'haeseleer, a seleção negativa muitas vezes não é tão efetiva quanto outras técnicas, mas oferece um nível de segurança maior por ser menos suscetível à tentativas deliberadas dos agentes maliciosos de enganar o sistema. Outras vantagens apontadas incluem a habilidade de detecção de mudanças nos eventos do sistema, de verificar mudanças em partes de um objeto sem a necessidade de verificar o objeto como um todo e de aplicar diversos detectores diferentes sobre um mesmo objeto, aumentando a probabilidade de detecção conforme o número de detectores utilizado. O autor também aponta que subconjuntos dos detectores podem ser utilizados de maneira autônoma e distribuída.

Alguns trabalhos incorporam processos da seleção clonal na seleção negativa (GARRETT, 2005): os detectores são gerados aleatoriamente, como no algoritmo

original, e aqueles que identificam elementos próprios passam pelo processo de seleção clonal. O objetivo é que esses detectores sejam ajustados até que não cubram mais o espaço próprio. A vantagem, nesse caso, é a de utilizar o poder de exploração do espaço de estados da seleção clonal para dispensar o controle determinístico complexo dos detectores feito na maior parte dos trabalhos da área.

4.5.2 Seleção clonal

A principal característica da seleção clonal, que a difere dos outros métodos, é seu mecanismo de ajuste da mutação. Enquanto nos outros métodos o grau de mutação (ou o grau em que a mutação muda) é constante, na seleção clonal esse grau é determinado por uma função baseada na aptidão da solução. Soluções com pouca derivação do estado ótimo sofrem pequenos graus de mutação. Da mesma maneira, outra característica única da seleção clonal é a função que define o número de clones de acordo com a aptidão da solução.

No entanto, algoritmos baseados na seleção clonal podem ser lentos em sua execução devido ao número de clones, que aumenta conforme a média função de aptidão aumenta. A otimização é uma área onde os algoritmos baseados na seleção clonal têm mostrado bons resultados (GARRETT, 2005).

4.5.3 Redes imunológicas artificiais

Devido às comparações entre cada possível par de anticorpos, as redes imunológicas são, de maneira geral, lentas. Como cada anticorpo é comparado com cada um dos outros, a complexidade (em questão de tempo) é geralmente $O(n^2)$. Algumas pesquisas na área tentam reduzir essa complexidade, permitindo apenas um número finito de comparações em cada iteração. No entanto, isso pode comprometer os resultados, devido ao menor número de interações entre os elementos da rede, o que parece ir contra o propósito inicial do modelo.

4.5.4 Teoria do Perigo

As dificuldades na implementação de sistemas baseados na Teoria do Perigo são as mesmas encontradas na aceitação da teoria na imunologia: a natureza dos sinais de perigo não é facilmente identificável; e o sistema aparentemente deve primeiro sofrer o dano para que as medidas de proteção possam ser tomadas.

4.6 Considerações finais

Os algoritmos apresentados nesse capítulo foram aplicados em sistemas computacionais de diversas áreas, apresentado resultados satisfatórios, em especial em sistemas de detecção. O domínio dos problemas de detecção de fraude é promissor

para esses algoritmos, como a maioria dos problemas de detecção.

Os resultados dos algoritmos apresentados nas seções anteriores foram comparados com os resultados de algoritmos tradicionais da área da Inteligência Artificial. Dos algoritmos imunológicos, são usados: AIRS, Immunos e seleção clonal. Esses algoritmos foram escolhidos por serem representantes característicos da área de sistema imunológicos artificiais e por terem implementações atualizadas e disponíveis abertamente.

Para os algoritmos tradicionais, foram escolhidos aqueles que são comumente usados na área de mineração de dados. Em especial, os algoritmos genéticos, que têm características semelhantes aos imunológicos, já que ambos têm inspiração na biologia.

Tabela 4.3: Algoritmos utilizados para comparação

Algoritmos tradicionais	Algoritmos imunológicos
Redes neurais	AIRS
Máquina de vetor de suporte	Immunos
Algoritmos genéticos	CLONALG
Árvores de decisão	
Learning Vector Quantization	

Nas próximas seções, será mostrada a aplicação de três algoritmos baseados no sistema imunológico natural a conjuntos de dados de fraudes. Os resultados são então comparados com algoritmos tradicionais da inteligência artificial.

Para que essa comparação tenha utilidade prática, primeiro é necessário que se estabeleçam critérios para a avaliação do desempenho desses algoritmos. O próximo capítulo apresenta os critérios de avaliação comumente empregados na mineração de dados. Apresenta também como esses critérios devem ser adaptados para a aplicação na detecção de fraude e demonstra como estes são aplicados nesse trabalho.

5 AVALIAÇÃO DE RESULTADOS

A avaliação de algoritmos de classificação (e de aprendizagem de máquina em geral) é um tópico controverso, que é objeto de discussão de diversos trabalhos. Segundo um teorema da aprendizagem de máquina conhecido como "*no free lunch*", algoritmos de aprendizagem têm performance igual quando comparados utilizando todos os possíveis conjuntos de dados. Ou seja, um algoritmo que é melhor aplicado a um domínio o faz somente abrindo mão de outros domínios.

Assim, a comparação simples desses algoritmos não tem valor prático. BROWNLEE nota que que muitos (ou a maioria) dos trabalhos foca apenas na qualidade da solução, que é apenas um aspecto da eficácia. Segundo BARR et al., existem diversos aspectos que devem ser considerados na avaliação de um método: eficiência, eficácia, robustez, complexidade, impacto, potencial para generalização e inovação (BARR et al., 1995). Em seu trabalho, os autores apresentam os passos básicos para a experimentação, conforme a listagem abaixo:

a) Definir os objetivos do experimento

b) Selecionar uma medida de performance e os fatores a serem explorados

c) Planejar e executar o experimento

d) Analisar os dados e tirar conclusões

e) Divulgar os resultados do experimento

Além disso, sugerem, no mesmo trabalho, diretrizes de alto-nível para o formato de divulgação dos resultados:

a) O experimento deve ser facilmente reproduzido

b) Todos os fatores que influenciam os resultados dever ser listados (código, ambiente de execução, etc)

c) As medições devem ser precisas

d) Os parâmetros devem ser especificados

e) Comparações com outros métodos

f) Reduzir a variabilidade dos resultados

g) Garantir que os resultados são abrangentes

Com isso em mente, é importante considerar as características únicas do domínio onde o método é aplicado. Os dados nos problemas de detecção de fraude apresentam desafios únicos que devem ser levados em consideração durante o desenvolvimento de um sistema de detecção, conforme apresentado na seção 2.1.

5.1 Detecção de fraude

Phua e outros autores listam as diversas medidas de performance utilizadas em trabalhos recentes na área de detecção de fraude (PHUA et al., 2010). Em relação à comparação da eficácia dos métodos, destacam-se os seguintes critérios:

a) Similaridade de uma instância com os exemplos de fraude conhecidos divida pela dissimilaridade com exemplos legais.

b) Análise da curva ROC (*Receiver Operating Characteristic*, Característica de Operação do Receptor): a taxa de verdadeiros positivos para falsos positivos.

c) Análise da área sob a curva ROC, que mede a probabilidade de uma instância positiva ser classificada como "mais positiva"do que uma instância negativa.

d) Entropia cruzada: mede a diferença entre o valor atribuído na classificação de uma instância e o valor real atribuído àquela instância nos dados de teste.

e) Escore Brier: o erro quadrático médio, ou seja, a média do somatório dos quadrados da diferença entre a classificação de uma instância e a classificação real.

f) Alguns trabalhos adotam um critério peculiar: as instâncias são classificadas de acordo com o seu valor monetário. Esse valor pode ser tanto explícito quanto baseado em modelos financeiros.

Em relação à eficiência dos métodos, destacam-se os seguintes critérios:

a) Velocidade de detecção (tempo até o alarme).

b) Número de tipos ou estilos de fraude detectados.

c) Processo de de detecção em tempo real ou em lote.

Os métodos tradicionais de medição, como a taxa de positivos (número de fraudes detectadas corretamente dividido pelo número de fraudes verdadeiras) e a precisão a um determinado limite (número de instâncias classificadas corretamente, dividido pelo número total de instâncias) foram abandonados pelos trabalhos recentes, devido à natureza peculiar da detecção de fraude.

A razão para isso é que o custo das classificações errôneas, no caso da detecção de fraude, são irregulares, incertos, variam de exemplo a exemplo e podem variar conforme o tempo. Falsos negativos são geralmente mais custosos do que falsos positivos. Um falso positivo geralmente leva apenas a uma verificação desnecessária por um especialista. Um falso negativo, no entanto, acarreta em uma fraude que não é detectada pelo sistema e não será reportada, deixando o fraudador impune. Mesmo assim, muitos dos sistemas de detecção comerciais, como a maioria dos departamentos governamentais que atuam na área de detecção de fraude, utilizam valores monetários como medida de avaliação (PHUA et al., 2010).

5.2 Avaliação dos modelos

A primeira parte na avaliação de um algoritmo de classificação é a avaliação dos modelos criados por ele. Um modelo é a saída de um algoritmo de classificação ou a estrutura intermediária utilizada por ele, e define como as instâncias serão classificadas. O modelo é então aplicado a cada uma das instâncias da base de dados e gera uma classificação dessas instâncias.

Essa classificação pode ser comparada com as classificações verdadeiras das instâncias, gerando um total de instâncias classificadas corretamente. Esse valor é comumente representado como uma porcentagem, um valor entre 0 e 1, calculado como *1 - taxaDeErros* e é chamado de *precisão*.

5.2.1 *Holdout*

Quando um modelo é criado à partir de um conjunto de dados e os testes são executados sobre esse mesmo conjunto, é muito provável que eles apresentem um bom resultado. No entanto, isso não garante que o modelo pode ser aplicado com a mesma eficácia sobre novos dados, o que geralmente é o objetivo dos sistemas de detecção. Para isso, o conjunto de dados é dividido em dados de treinamento e de testes.

O resultado dos testes pode ser altamente dependente dessa separação dos dados. Separar os dados de forma diferente geralmente altera o resultado dos testes. Para evitar essa dependência, foram criados métodos como *holdout* e *cross-validation*.

Para que seja possível avaliar se um modelo pode prever novos dados após o treinamento, a base de dados deve ser dividida em duas partes: um conjunto de

treinamento e um conjunto de teste. O modelo é criado utilizando apenas os dados de treinamento e avaliado utilizando os dados de teste. Essa técnica é denominada *holdout*, porque uma parte dos dados é retirada do processo de treinamento para que possa ser utilizada nos testes. Os resultados podem ser melhorados executando várias iterações de *holdout*, dividindo um conjunto de dados aleatoriamente a cada vez e calculando a média e o desvio padrão da precisão.

Caso o resultado dos testes de um modelo quando aplicado aos mesmos dados utilizados no treinamento seja bom, mas o resultado quando aplicado aos dados de teste sejam ruins, diz-se que ocorre *overfitting*: o modelo ajusta-se tanto aos dados de treinamento que não consegue criar regras genéricas o suficiente para identificar amostras semelhantes mas não exatamente iguais.

Há ainda a possibilidade do modelo não gerar um resultado bom nem mesmo para os dados de treinamento. Nesse caso, diz-se que ocorre *underfitting*: o modelo não é capaz de criar regras que se ajustem aos dados de treinamento, ou seja, o modelo não é capaz de memorizar os dados de treinamento.

5.2.2 *Cross-validation*

Uma alternativa mais sofisticada e eficiente do *holdout* é denominada *cross-validation*. Nela, os dados são divididos em n partições, chamadas de *folds*. A cada iteração, uma partição é utilizada dados de testes, enquanto as outras n - *1* são utilizadas como treinamento.

a) Dividir aleatoriamente os dados em dois conjuntos, treinamento e testes.

b) Criar o modelo com base no conjunto de dados de treinamento.

c) Testar o modelo utilizando o conjunto de dados de testes.

d) Repetir os passos a) a c).

e) Calcular a média e o erro padrão da média dos resultados de cada iteração.

Uma alternativa à divisão aleatória dos dados, chamada de *cross-validation* estratificada, é a divisão onde a proporção das classes em cada partição é a mesma do conjunto total de dados. Isso aumenta ainda mais a fidelidade do treinamento e dos testes. Obviamente, esse método só pode ser utilizado em conjuntos de dados que contém um atributo de classe (ou seja, supervisionados).

Quando um conjunto de dados é pequeno, pode-se utilizar um tipo de *cross-validation* chamado *leave-one-out*, onde n, o número de partições, é igual ao número de instâncias no conjunto de dados (ou seja, cada partição tem tamanho 1). Assim, a cada iteração, n - *1* instâncias são usadas no treinamento e o teste é feito sobre a instância restante.

5.2.3 *Grid search*

O *grid search* é outra técnica desenvolvido para diminuir o efeito de fatores não relacionados ao algoritmo na execução dos testes. Assim como o *cross-validation* diminui o efeito da separação do conjunto de dados em dados de treinamento e de teste, o *grid search* diminui o efeito da escolha dos parâmetros para o algoritmo. Esses parâmetros são valores numéricos ou condições que alteram o funcionamento da classificação das instâncias. Os algoritmos podem ter seu comportamento alterado através de parâmetros. Um exemplo disso é um classificador que aceita apenas valores maiores do que um determinado limiar. Diminuir ou aumentar esse limiar faz com que mais ou menos valores sejam aceitos.

Consequentemente, alterar o valor de um parâmetro afeta a precisão do classificador. Modelos mais complexos podem ter diversos parâmetros, e a relação entre estes pode também afetar a precisão. O treinamento é, então, dependente do valor dos parâmetros de um modelo. No entanto, esses parâmetros não são facilmente configuráveis, devido a dependências não-lineares que existem entre eles, que formam um sistema complexo. Os parâmetros podem também ser dependentes de contexto: parâmetros que geram um bom resultado para um determinado conjunto de dados podem não gerar um resultado tão bom para outro conjunto.

O *grid search* consiste em executar e comparar a execução do algoritmo utilizando diversos valores para os parâmetros, para decidir qual é a melhor combinação para um conjunto de dados específico. *Grid search* é um processo de diversas iterações de *cross-validation* utilizando diferentes valores para os parâmetros de um modelo. No entanto, como os valores dos parâmetros podem ser dependentes entre si (e geralmente são), devem ser testadas todas as combinações de valores para cada e entre cada parâmetro. Esse é um processo computacionalmente custoso, já que o número de combinações é uma função exponencial do número de parâmetros e do número de valores distintos de cada parâmetro. Além disso, alguns parâmetros podem ter um número infinito de valores, como é o caso para parâmetros que são números reais.

Para resolver esse problema, no *grid search* são utilizados um conjunto finito de valores para cada parâmetro. Esse conjunto é geralmente uma faixa de valores organizada em uma progressão aritmética, geométrica ou exponencial. Dessa forma é possível reduzir o conjunto (potencialmente infinito) de combinações possíveis para um tamanho razoável, através de um comprometimento no número de valores testados.

O resultado de um *grid search* é a lista de resultados para a execução do algoritmo utilizando cada combinação de parâmetros e os resultados. Dessa forma, é possível utilizar a combinação de parâmetros que gera o melhor resultado.

É importante notar que o *grid search* é um processo facilmente paralelizável.

Considerando que um algoritmo aceite dois parâmetros, e cada um desses parâmetros tenha 3 valores possíveis, cada uma das 9 combinações possíveis é testada. Como cada teste é independente do outro, pode-se executar até 9 processos ao mesmo tempo. Caso a divisão aleatória dos dados nas iterações do *cross-validation* seja calculada previamente, é possível que cada uma delas seja executada também em paralelo, aumentando ainda mais o número de processos simultâneos que podem ser executados.

5.2.4 Dados baseados em tempo

Uma característica importante de algumas áreas da detecção de fraude é que os dados são baseados em tempo, ou seja, o momento em que os dados de uma instância foram coletados é um atributo importante daquela instância. Para esse tipo de base de dados, é interessante que o treinamento seja feito utilizando dados mais antigos, e os testes utilizando dados recentes. Dessa maneira, é possível detectar se existe uma evolução nos dados conforme o tempo.

A razão para isso é que, caso os dados sejam misturados aleatoriamente, a informação da evolução dos padrões é perdida durante o treinamento. Caso sejam utilizados dados recentes para os testes, essas alterações podem ser detectadas caso haja uma variação entre os resultados dos testes nos dados de treinamento (antigos) e de testes (recentes).

Caso essa diferença seja detectada, uma solução para adicionar essa informação ao modelo é aplicar um peso sobre as instâncias de acordo com o tempo. Assim, instâncias mais recentes teriam mais importância para o treinamento do que instâncias mais antigas. Deve ser utilizado um balanceamento para essa solução, já que utilizar todos os dados disponíveis (incluindo os dados recentes) torna o modelo mais completo, mas torna impossível avaliar com segurança a eficácia desse modelo.

5.3 Avaliação de classificadores

Na área de detecção de fraude, geralmente se lida com classificadores binários. Classificadores binários são aqueles que, para um dado conjunto de valores de entrada, geram como valor de saída um valor Booleano: verdadeiro ou falso, positivo ou negativo. Os sistemas de detecção de fraude se situam, na maioria dos casos, nessa categoria: os dados referentes a uma instância passam pelo sistema de detecção e são classificados como legítimos ou fraudulentos (BEWICK, CHEEK, BALL, 2007). Sistemas mais complexos podem gerar saídas com mais informações sobre a instância, como o grau de certeza da previsão ou o grau de semelhança entre a instância e os dados da base.

Como exemplo, inspirado em BEWICK, CHEEK, BALL (2007), é apresentado

nessa seção um sistema de detecção de fraude, que considera como único atributo das instâncias o número de ocorrências de um determinado valor. Nesse ambiente simplificado, a instância é considerada fraudulenta caso um limiar de ocorrências seja ultrapassado.

Esse sistema de testes utiliza como único critério de avaliação o número de ocorrências. Um *limiar de classificação* é definido: um parâmetro que o sistema de detecção utilizará para guiar a previsão da detecção. Se o limiar de classificação fosse 1, todas as instâncias que tivessem mais de uma ocorrência (nos valores de exemplo, as instâncias 0 e 2) seriam consideradas fraudulentas.

Tabela 5.1: Exemplo de resultado

		Dados reais		
		Fraude	Legítimo	Total
Saída	Fraude	300	200	500
	Legítimo	100	1000	1100
	Total	400	1200	1600

Um exemplo do resultado de um teste nesse sistema é mostrado na tabela 5.1. As colunas sob "Dados reais" mostram o número de instâncias fraudulentas e legítimas; as linhas sob "Saída" mostram a classificação gerada pelo sistema. A partir desses dados, a avaliação dos resultados é feita usando dois conceitos: sensibilidade e especificidade.

5.3.1 Matriz de confusão

A avaliação dos classificadores binários é feita com base em uma *matriz de confusão*, conforme mostrado na tabela 5.2. Nessa tabela são listados o número de valores (ou porcentagem) para cada combinação de saída do sistema e a classificação real. Instâncias corretamente classificadas ocupam as colunas Positivo e Negativo, enquanto instâncias cujo classificação difere da realidade ocupam as colunas "Falsos positivos" e "Falsos negativos".

Como o conjunto de dados possui apenas duas classes, a matriz de confusão é $2x2$, mas o número de linhas e colunas pode ser maior, sendo igual ao número de classes. Todos os principais indicadores podem ser extraídos através da matriz de confusão, fazendo dela uma forma sucinta de apresentação dos resultados.

A diagonal principal representa as instâncias corretamente classificadas (*i.e.* instâncias positivas classificadas como positivas, negativas classificadas como negati-

Tabela 5.2: Matriz de confusão

		Dados reais	
		Positivo	Negativo
Saída	Positivo	Positivo	Falso positivo
	Negativo	Falso negativo	Negativo

vas). Todas as outras células da matriz são instâncias incorretamente classificadas, onde o cruzamento do nome da coluna e da linha mostra a classificação gerada pela algoritmo e a classificação real. Através dos elementos da diagonal principal, é possível calcular:

a) *Taxa de positivos verdadeiros (TP)*: a proporção das instâncias classificadas como x entre todas as que realmente fazem parte da classe x, calculada dividindo o elemento da classe na diagonal principal pela soma dos elementos na mesma linha. Também é conhecida como *recall*.

b) *Taxa de falsos positivos (FP)*: a proporção das instâncias classificadas como x que na verdade pertencem a outra classe entre todas as que não fazem parte da classe x, calculada subtraindo o elemento da classe na diagonal principal da soma dos elementos na mesma coluna e dividindo o valor resultante pela soma dos elementos nas linhas de todas as outras classes.

c) *Precisão*: a proporção de instâncias que fazem parte da classe x entre todas as que foram classificadas como x, calculada dividindo o elemento da classe na diagonal principal pela soma dos elementos na mesma coluna.

d) F-measure: uma combinação da precisão e taxa de positivos verdadeiros, calculada através da média harmônica da precisão e *recall*, 2 * (precisão * *recall*) / (precisão + *recall*).

5.3.2 Sensibilidade e especificidade

Sensibilidade refere-se à proporção de instâncias corretamente classificadas como positivas. *Especificidade* refere-se à proporção de instâncias corretamente classificadas como negativas. Tomando como exemplo os dados da tabela 5.1, a sensibilidade é igual a 300 / 400 = 0.75; e a especificidade é igual a 1000 / 1200 = 0.8333. Ou, visto de outra maneira, 75% das fraudes foram realmente classificadas como fraudes; e 83.33% das instâncias legítimas foram classificadas como legítimas.

Apenas a consideração desses dois valores pode levar a uma correta avaliação dos resultados do teste, principalmente nos domínios da detecção de fraude. Uma alta sensibilidade não necessariamente implica em uma alta especificidade, e vice versa. Dessas duas informações são derivados os *valores preditivos positivos e negativos*.

5.3.3 Valores preditivos

O *valor preditivo positivo* é a chance de uma instância ser realmente uma fraude caso seja classificada como tal pelo sistema. O *valor preditivo negativo* é a chance de uma instância ser realmente legítima caso seja classificada como tal pelo sistema. Para os dados de exemplo, o valor preditivo positivo é 300 / 500 = 0.6; e o valor preditivo negativo é 1000 / 1100 = 0.9091. Ou, visto de outra maneira, 60% das instâncias que foram classificadas como fraude eram realmente fraudes, e 90.91% das instâncias classificadas como legítimas eram realmente legítimas.

Esses dois valores são o oposto da sensibilidade e especificidade, respectivamente. Enquanto os valores preditivos dão uma avaliação direta sobre os resultados dos testes, a sensibilidade e especificidade não são afetadas pela proporção dos valores nas instâncias, ou seja, não são alteradas quando há uma alteração na proporção de fraudes.

5.3.4 Taxas de verossimilhança

A sensibilidade e especificidade tornam-se ainda mais úteis quando combinadas, gerando as taxas de verossimilhança. A taxa de verossimilhança de um resultado positivo (LR^+) é a razão entre a probabilidade de um resultado positivo no teste caso a instância seja realmente positiva (coluna "positivo"da matriz de confusão) e a probabilidade de um resultado positivo no teste caso a instância seja na verdade negativa:

$$LR^+ = \frac{sensibilidade}{1 - especificidade} \tag{5.1}$$

No exemplo, $LR^+ = 0.75 / (1 - 0.8333) = 4.4991$. Isso significa que um resultado positivo nos testes é 4.4991 vezes mais provável para instâncias que são realmente positivas do que para aquelas que não são.

De maneira, similar, a taxa de verossimilhança negativa (LR^-) é a razão entre a probabilidade de um resultado negativo no teste caso a instância seja realmente negativa (coluna "negativo"da matriz de confusão) e a probabilidade de um resultado negativo no teste caso a instância seja na verdade positiva:

$$LR^- = \frac{1 - sensibilidade}{especificidade} \tag{5.2}$$

No exemplo, LR^{-} = (1 - 0.75) / 0.8333 = 0.3, o que significa que um resultado negativo no teste é 0.3 vezes mais provável para instâncias que são realmente negativas do que para aquelas que não são.

Desses valores, pode-se atestar a utilidade do método para classificação. Uma alta taxa de verossimilhança positiva indica que o teste é útil para verificar se uma instância é positiva, enquanto uma alta taxa de verossimilhança negativa é útil para verificar se uma instância é negativa. Assim como os valores preditivos, essas taxas são sensíveis à predisposição dos dados na base de dados.

5.3.5 Índice de Youden e ROC

Os dados de teste do exemplo mostrados nas tabelas até agora consideraram apenas um valor como limiar de classificação. Mudanças nesse limiar afetam a precisão dos testes: caso o limiar seja seja igual a -1, todas as instâncias são classificadas como fraudulentas, já que todas possuem como número de ocorrências um número maior ou igual a zero. Por outro lado, caso o limiar seja um número maior do que qualquer uma das instâncias, todas serão classificadas como legítimas. Em sistemas reais, é comum que sejam testados diversos valores para esse limiar sobre uma mesma base de dados, para que o melhor seja identificado e usado no sistema final. A tabela 5.3 é um exemplo de resultados para esse tipo de teste.

Tabela 5.3: Exemplo de resultados para testes comparativos de limiares

Limiar	Fraudes	Legítimos	Sensibilidade	Especificidade	Índice de Youden
0	400	0	1	0	0
1	395	400	0.9875	0.3333	0.3208
2	380	600	0.95	0.5	0.45
3	375	900	0.9375	0.75	0.6875
4	200	1000	0.5	0.8333	0.3333
99	0	1200	0	1	0
Total	400	1200	-	-	-

Fonte: inspirado em BEWICK, CHEEK, BALL (2007).

Uma medida para a avaliação dos diferentes limiares pode ser a sensibilidade e especificidade. O índice de Youden (J), que é uma derivação desses valores, é uma medida apropriada nesses casos (eq. 5.3). Esse índice é um valor normalizado

na faixa [0, 1], onde 1 significa um teste perfeito, classificando todas as instâncias corretamente; e 0 significa um teste sem valor nenhum. A tabela 5.3 mostra os valores para os resultados com diversos limiares. É importante notar como valores extremos maximizam a sensibilidade e a especificidade, mas apenas aqueles limiares que maximizam os dois valores recebem índices significativos.

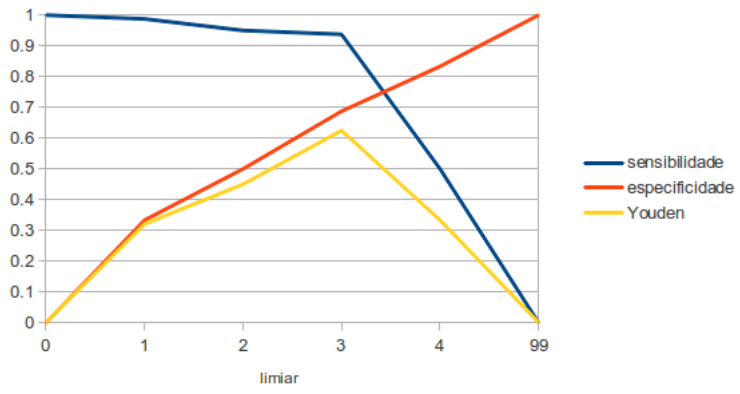
$$J = sensibilidade + especificidade - 1 \tag{5.3}$$

Uma característica importante do índice de Youden é que ele estabelece uma equivalência na relevância dos falsos positivos e negativos. Quando um é mais relevante que o outro, devido à peculiaridades nos dados ou no domínio do problema, ele não é apropriado. Nesses casos, pode-se atribuir pesos diferentes a ambos os valores. Um exemplo de adaptação do cálculo do índice de Youden é mostrado abaixo. Nessa fórmula, os valores ainda são mantidos na mesma faixa, mas falsos positivos recebem o dobro da significância.

$$J = 0.75 * sensibilidade + 0.25 * especificidade \tag{5.4}$$

Como pode-se observar na tabela 5.3, alterações no valor usado na classificação alteram o número de instâncias classificadas como positivas e negativas. Geralmente os resultados são distribuídos começando com um grande valor de sensibilidade e um valor baixo de especificidade; esses valores convergem até um máximo global; e então terminam em com um valor baixo de sensibilidade e um grande valor de especificidade. A variação desses três valores, e a interação entre eles, é mostrada no gráfico da figura 5.1.

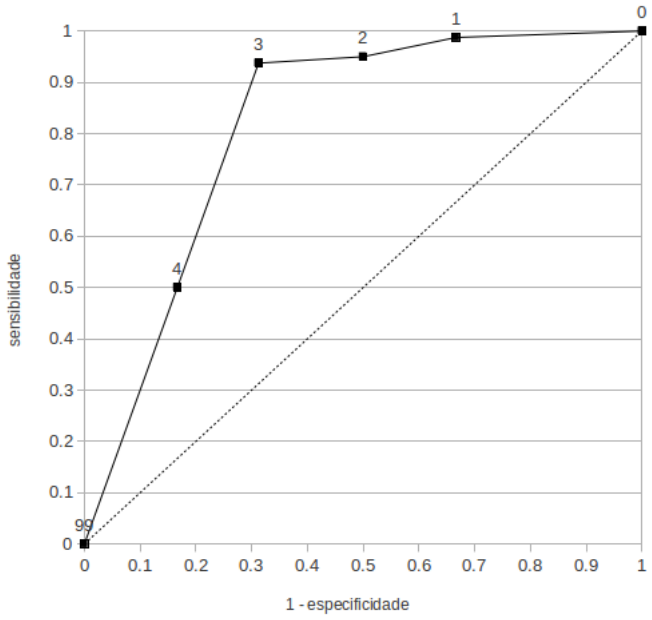
Figura 5.1: Variação dos valores de avaliação conforme o limiar de detecção



No lado esquerdo, onde o limiar é muito baixo, a sensibilidade é 1 e a especifici-

dade, zero; enquanto no lado direito, onde o limiar é muito alto, ocorre o oposto: a sensibilidade é zero e a sensibilidade, 1. Conforme o limiar aumenta (começando do lado esquerdo e caminhando para a direita), ocorreu uma diminuição na especificidade (algumas instâncias fraudulentas começam a ser classificadas como legítimas). No entanto, há um aumento muito maior na especificidade (instâncias legítimas começam a ser classificadas corretamente). É possível ver como o índice de Youden mostra a real proporção entre esses dois valores. Uma forma mais sucinta para apresentar esse tipo de informação é através de um gráfico dos valores de sensibilidade e 1 - especificidade, que é denominado ROC (característica operativa do receptor, *receiver operating characteristic*), conforme mostrado na figura 5.2.

Figura 5.2: Gráfico da curva ROC dos valores de exemplo



No gráfico é visualmente aparente que o valor 3 como limiar de classificação oferece o melhor balanceamento entre sensibilidade e especificidade. Também é possível perceber como as alterações no limiar influenciam a distribuição desses dois valores. Um teste perfeito, que classificasse todas as instâncias corretamente, teria ambos os valores iguais a 1, tendo portanto um ponto no canto superior esquerdo do gráfico. Os pontos mais próximos dessa localização são aqueles que melhor classificam as instâncias.

Caso as instâncias fossem classificadas aleatoriamente, tendo chances iguais de serem classificadas tanto positivas quanto negativas, a sensibilidade e a especificidade seria ambas 0.5, e o teste não teria qualquer valor. Essa situação é representada pela linha diagonal entre os pontos (0, 1) e (1, 0). Essa linha é outro indicador visual

útil: testes abaixo dela (na direção do canto inferior direito) têm muito pouco valor; testes próximos dela têm valor mediano; e testes acima (na direção do canto superior esquerdo) têm grande valor.

Uma das maneiras de quantificar a (dar um valor à) validade de um atributo como variável de diagnóstico é através do cálculo dá área sob a curva ROC (denominada AUROC, *area under the ROC curve*). O teste ideal citado acima teria área igual a 1, enquanto testes completamente aleatórios teriam área 0.5. Testes reais normalmente situam-se entre esses dois valores, com valores mais próximos de 1 representando testes mais precisos.

O cálculo da área pode ser feito somando-se a área dos trapézios formados pela curva. Tomando dois pontos da curva, (0.1667, 0.5) e (0.3125, 0.9375), o trapézio formado por esses pontos e os pontos (0.1667, 0.0) e (0.3125, 0.0) é igual a (0.9375 - 0.5) x (0.3125 + 0.1667)/2 = 0.1048. Aplicando essa regra aos outros pontos da curva, obtêm-se uma área total de 0.8161, ou seja, uma instância fraudulenta tem 81.61% de ter um número de ocorrências de inadimplência maior do que uma instância legítima.

5.3.6 Comparação

Uma vez que possa ser quantificada, a capacidade de diagnóstico de uma variável pode ser comparada com outras, simplesmente comparando-se as suas curvas ROC e as áreas sob essas curvas. Variáveis com uma maior área sob a curva geram diagnósticos mais confiáveis. O formato da curva também pode servir como instrumento de análise: uma variável pode ter comportamentos preferíveis caso as condições de teste sejam diferenciadas, ou para filtrar casos específicos. Por exemplo, uma variável que, para valores muito baixos de sensibilidade, apresenta uma alta taxa de especificidade pode ser usada quando quer-se favorecer a especificidade.

A área sob a curva ROC é um método útil para a medição da precisão de diagnósticos, além da comparação entre diferentes testes. No entanto, ela não deve ser tomada como verdade absoluta. A sensibilidade e a especificidade podem manter-se fixas em um ambiente de testes, mas podem variar conforme as características da população analisada.

Uma consideração importante quando analisa-se o desempenho de dois ou mais sistemas é que muito difícil (ou até mesmo impossível) excluir-se todos os fatores externos, resultando em uma análise completamente imparcial. Nesse caso específico, o resultado dos testes em cada algoritmo é fortemente influenciado pelas características dos dados na base de dados, embora os métodos de análise dos resultados procuram minimizar essa influência.

5.3.7 Sistemas imunológicos artificiais

Uma característica dos algoritmos inspirados na biologia é que eles tem um elemento estocástico, ou seja, em qualquer momento da sua execução, seu estado tem um elemento não-determinístico. Por isso, ao contrário dos métodos tradicionais, diversas execuções do treinamento podem apresentar resultados diferentes, mesmo quando os mesmos valores são usados para os parâmetros.

5.4 Considerações finais

A aplicação de critérios é vital para que um experimento possa ser analisado corretamente. Critérios não apropriados podem levar a informações erradas e conclusões que não seguem a realidade. Um exemplo clássico disso é um modelo que apresenta *overfitting* (seção 5.2.2). Caso o modelo fosse testado apenas nos dados de treinamento, o resultado pareceria extremamente satisfatório. No entanto, o modelo apresentaria um desempenho péssimo em outros conjuntos de dados, o que tornaria-o inútil para qualquer aplicação.

Nos testes, a execução de cada algoritmos é feita utilizando *grid search* e *cross-validation*. Utilizar *cross-validation* diminui a influência do conjunto de dados no resultado dos testes. Como o algoritmo é treinado e testado várias vezes, cada vez com uma divisão diferente do conjunto de dados para treinamento e testes, quaisquer tendências nos dados são eliminadas, e o resultado se torna mais confiável.

Geralmente, o *grid search* é utilizado para selecionar os melhores parâmetros para um modelo, para que esses parâmetros sejam incorporados no sistema final, obtendo a melhor performance possível. Nesse trabalho, ele é utilizado para que o modelo utilizado nos testes seja o melhor possível para os conjuntos de dados dos testes.

Para a avaliação dos resultados, para cada algoritmo, são utilizados como critérios o erro médio relativo e o índice de Youden. O erro médio relativo é uma medida padrão simples, que mostra o quanto o resultado de um teste difere dos valores corretos. Ele é utilizado como uma métrica simples para o desempenho de um algoritmo, e pode ser utilizado como análise superficial na comparação de algoritmos. No entanto, o erro médio relativo não pode ser utilizado como métrica única: devido a simplicidade do seu cálculo, não é suficiente para a análise de áreas mais complexas como a detecção de fraude.

O índice de Youden é uma medida de performance que leva em consideração não só o número de instâncias classificadas corretamente. Para que o índice de Youden seja alto, tanto a taxa de falsos positivos quanto a de falsos negativos deve ser baixa. Conforme apresentado na seção 5.1, na detecção de fraude os falsos negativos são muito mais críticos do que os falsos positivos. Assim, serão calculados dois índices:

o índice de Youden padrão e o índice de Youden com maior significância para os falsos negativos. Esse é calculado através da fórmula:

$$J = 0.25 * sensibilidade + 0.75 * especificidade \tag{5.5}$$

A partir do próximo capítulo é apresentado o desenvolvimento dos experimentos, tendo como base os conceitos apresentados nesse e nos capítulos anteriores.

6 PLANEJAMENTO DO EXPERIMENTO

Conforme apresentado nos capítulos anteriores, a modelagem de um sistema de detecção de fraude com a utilização de técnicas inspiradas no sistema imunológico natural busca trazer diversas vantagens tanto no processo de planejamento quanto na sua arquitetura final. Esse trabalho busca identificar exatamente *quanto* esses modelos podem aperfeiçoar os métodos existentes. O método utilizado é a comparação dos resultados da classificação de algoritmos tradicionais e Sistemas Imunológicos Artificiais sobre um mesmo conjunto de dados. Esta seção apresenta os elementos principais para o desenvolvimento da proposta.

6.1 Etapas do experimento

A comparação dos resultados da execução de diferentes algoritmos é baseada em critérios de avaliação. A definição desses critérios é uma parte vital da preparação do experimento. Deve-se garantir que esses critérios sejam apropriados para a comparação dos métodos avaliados e que eles reflitam as necessidades e desafios dos ambientes em que esses métodos serão implementados. Sem critérios bem definidos, corre-se o risco de gerar uma análise que não tem nenhum valor prático ou que leve a conclusões erradas, por examinar dados irrelevantes sobre o resultado dos experimentos. Nesse trabalho, a comparação dos resultados foi divida em duas etapas: execução e análise. A etapa de execução consistiu em:

1. Selecionar os algoritmos imunológicos.
2. Selecionar os algoritmos tradicionais para comparação.
3. Preparar os conjuntos de dados para importação no WEKA.
4. Para cada algoritmo:
   (a) Utilizar *grid search* para otimizar os parâmetros para os dados.
   (b) Utilizando esses parâmetros, executar os testes usando *cross-validation*.

5. Calcular, para o resultado de cada algoritmo:
   (a) O erro médio relativo.
   (b) O índice de Youden.
   (c) O índice de Youden com significância maior para os falsos negativos.

A etapa de análise consistiu em:

1. Tabular os resultados da execução dos algoritmos.
2. Ordenar e classificar os resultados de acordo com os critérios definidos no capítulo 5.
3. Apresentar os resultados.

Conforme o padrão nesse tipo de experimento, os dados são apresentados em formato de tabela e gráfico. Os critérios escolhidos já são parte integrante dos resultados exibidos na execução do WEKA. Por um lado, isso mostra a aderência dessa plataforma aos padrões da área, por outro, demonstra a facilidade que uma ferramenta que apresenta os resultados em um formato padrão oferece para experimentos como esse.

6.2 Descrição dos dados de teste

Para os testes foram utilizados dois conjuntos de dados contendo informações de contas de cartões de crédito. Esses dados estão disponíveis publicamente e fazem parte de um projeto chamado StatLog (MICHIE, SPIEGELHALTER, TAYLOR, 1994). Esse projeto foi concebido para testar diversos métodos de classificação em problemas grandes e comercialmente relevantes, comparando os seus resultados e determinando o quanto eles atendiam as necessidades da indústria. Conforme a sua própria descrição, os objetivos do projeto eram três:

a) Possibilitar medidas críticas de desempenho para procedimentos de classificação disponíveis,

b) Indicar a natureza e escopo dos desenvolvimentos futuros necessários para que os métodos atendam as necessidades e expectativas dos usuários e

c) Indicar as direções mais promissoras de desenvolvimento para abordagens comercialmente imaturas.

Os conjuntos de dados são denominados German Credit (Cr.Ger) e Australian Credit (Cr.Aust). Ambos contém informações sobre contas de crédito, conforme apresentado nas próximas seções.

6.2.1 Cr.Ger

Esse conjunto de dados foi cedido pelo professor Hans Hoffman, da Universidade de Hamburgo. Ele contém 1000 instâncias. Os atributos e valores que esses atributos podem assumir são descritos na listagem 6.1 (traduzido do original)[1].

Listagem 6.1: Atributos do conjunto de dados alemão

```
1       Atributo 1: (qualitativo)
2       Situação da conta corrente existente
3       A11 : ... < 0 DM
4       A12 : 0 <= ... < 200 DM
5       A13 : ... >= 200 DM
6       A14 : sem conta corrente
7
8       Atributo 2: (numérico)
9       Duração em meses
10
11      Atributo 3: (qualitativo)
12      Histórico de crédito
13      A30 : nenhum crédito retirado / todos os créditos pagos apropriadamente
14      A31 : todos os créditos nesse banco pagos apropriadamente
15      A32 : créditos existentes pagos apropriadamente até agora
16      A33 : atraso no pagamento no passado
17      A34 : conta crítica / outros créditos existentes (não nesse banco)
18
19      Atributo 4: (qualitativo)
20      Propósito
21      A40 : carro (novo)
22      A41 : carro (usado)
23      A42 : móveis/equipamento
24      A43 : rádio/televisão
25      A44 : eletrodomésico
26      A45 : reparos
27      A46 : educação
28      A47 : (férias - não existe no conjunto de dados)
29      A48 : reciclagem profissional
30      A49 : negócios
31      A410 : outros
32
33      Atributo 5: (numérico)
34      Quantidade de crédito
35
36      Atributo 6: (qualitativo)
37      Poupança
38      A61 : ... < 100 DM
39      A62 : 100 <= ... < 500 DM
40      A63 : 500 <= ... < 1000 DM
41      A64 : .. >= 1000 DM
42      A65 : desconhecido / sem poupança
43
```

[1]Na descrição dos atributos, DM significa *deutsche mark* (marco alemão), a moeda corrente na Alemanha na época da coleta dos dados. Para efeito de comparação, o Banco Central Europeu estipulou a conversão irrevogável do marco alemão, a partir de 1º de janeiro de 1999, em DM 1.95583 = €1 (http://www.ecb.int/press/pr/date/1998/html/pr981231_2.en.html).

```
44      Atributo 7: (qualitativo)
45      Emprego atual desde
46      A71 : desempregado
47      A72 : ... < 1 ano
48      A73 : 1 <= ... < 4 anos
49      A74 : 4 <= ... < 7 anos
50      A75 : .. >= 7 anos
51
52      Atributo 8: (numérico)
53      Taxa de parcelamento em porcentagem do rendimento disponível
54
55      Atributo 9: (qualitativo)
56      Estado civil e sexo
57      A91 : masculino : divorciado/separado
58      A92 : feminino : divorciada/separada/casada
59      A93 : masculino : solteiro
60      A94 : masculino : casado/viúvo
61      A95 : feminino : solteira
62
63      Atributo 10: (qualitativo)
64      Outros devedores / fiadores
65      A101 : nenhum
66      A102 : devedor solidário
67      A103 : fiador
68
69      Atributo 11: (numérico)
70      Residência atual desde
71
72      Atributo 12: (qualitativo)
73      Propriedade
74      A121 : imóvel
75      A122 : se não A121 : financiamento / seguro de vida
76      A123 : se não A121/A122 : carro ou outro, não incluso no atributo 6
77      A124 : desconhecido / sem propriedade
78
79      Atributo 13: (numérico)
80      Idade em anos
81
82      Atributo 14: (qualitativo)
83      Outros planos de parcelamento
84      A141 : banco
85      A142 : lojas
86      A143 : nenhum
87
88      Atributo 15: (qualitativo)
89      Residência
90      A151 : alugada
91      A152 : própria
92      A153 : gratuita
93
94      Atributo 16: (numérico)
95      Número de créditos existentes nesse banco
96
97      Atributo 17: (qualitativo)
98      Emprego
99      A171 : desempregado / sem proficiência / não-doméstico
100     A172 : sem proficiência / doméstico
101     A173 : proficiente / funcionário público
102     A174 : administrador / auto-empregado /
```

```
103             empregado altamente qualificado / oficial
104
105       Atributo 18: (numérico)
106       Número de dependentes
107
108       Atributo 19: (qualitativo)
109       Telefone
110       A191 : nenhum
111       A192 : sim, registrado sob o nom do consumidor
112
113       Atributo 20: (qualitativo)
114       Trabalhador estrangeiro
115       A201 : sim
116       A202 : não
```

6.2.2 Cr.Aust

Os atributos do conjunto de dados australiano são descritos na listagem 6.2 (traduzido do original). Ele contém 690 instâncias. Uma grande desvantagem, muito comum nesse tipo de conjunto de dados (seção 2.1), é a de o nome dos campos ter sido alterado, perdendo o significado original. Para garantir a privacidade das informações contidas no conjunto de dados, provavelmente por questões competitivas, a empresa que cedeu os dados utilizou um processo de anonimização, garantindo que as informações confidenciais não possam ser recuperadas.

Os dados ainda podem ser utilizados para a validação de sistemas de detecção, mas não é possível saber de que forma o sistema chegou ao diagnóstico, tornando o resultado bem menos útil.

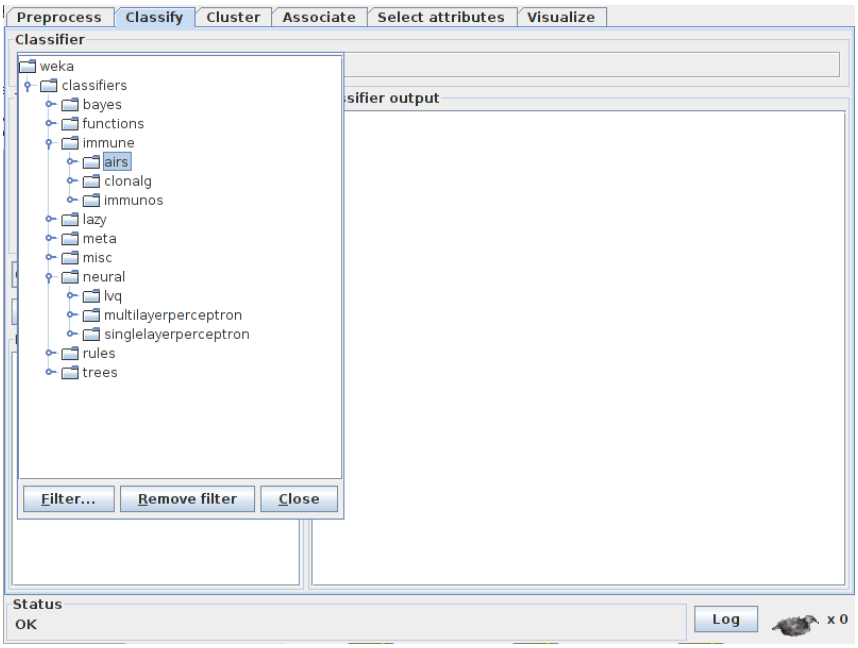
Listagem 6.2: Atributos do conjunto de dados Cr.Aust

```
 1  A1: b, a
 2  A2: contínuo
 3  A3: contínuo
 4  A4: u, y, l, t
 5  A5: g, p, gg
 6  A6: c, d, cc, i, j, k, m, r, q, w, x, e, aa, ff
 7  A7: v, h, bb, j, n, z, dd, ff, o
 8  A8: contínuo
 9  A9: t, f
10  A10: t, f
11  A11: contínuo
12  A12: t, f
13  A13: g, p, s
14  A14: contínuo
15  A15: contínuo
16  A16: +,- (atributo de classe)
```

6.3 Algoritmos imunológicos

Os algoritmos imunológicos utilizados no experimento são parte de um pacote de algoritmos desenvolvido por Jason Brownlee (BROWNLEE, 2011), na versão mais atual (1.8, maio de 2011). Esse pacote foi especialmente desenvolvido para a plataforma de aprendizagem de máquina WEKA (descrita na seção 6.4) e são disponibilizados através de uma licença aberta (GNU GPL). Nele, são implementadas diversas categorias de algoritmos, como Redes Neurais e Sistemas Imunológicos Artificiais. A figura 6.1 mostra esses algoritmos conforme apresentados na interface do WEKA, nos módulos *airs*, *clonalg* e *immunos* em *weka.classifiers.immune* e no módulo *lvq* em *weka.classifiers.neural*.

Figura 6.1: Algoritmos no WEKA



Dentre os algoritmos imunológicos, serão utilizados:

a) **Artificial Immune Recognition System (AIRS)**: algoritmo imunológico supervisionado descrito na seção 4.1.6.

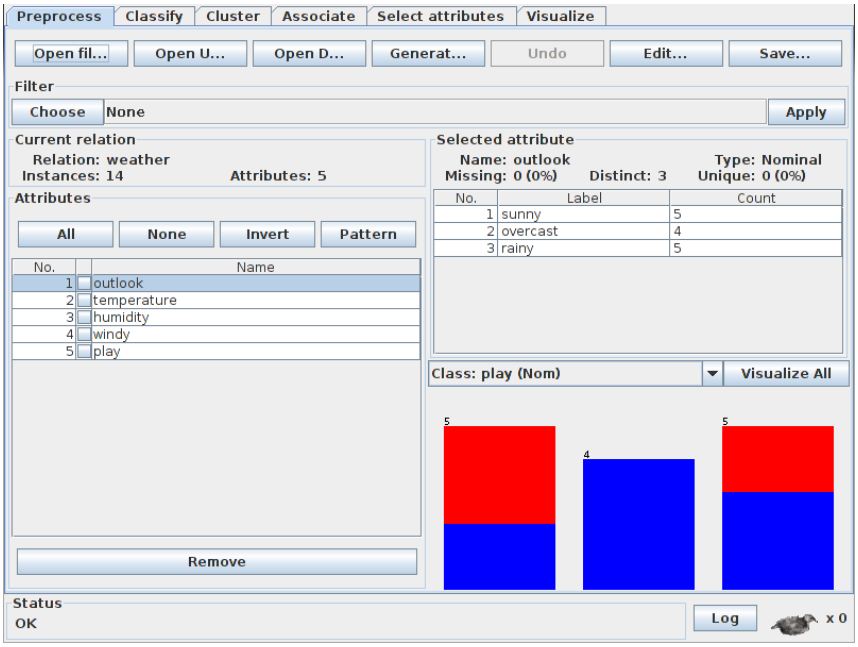
b) **Clonal Selection Algorithm (CLONALG)**: um dos principais algoritmos imunológicos, o algoritmo da seleção clonal, descrito na seção 4.1.3.

c) **Immunos-81**: algoritmo imunológico descrito na seção 4.1.7.

6.4 WEKA

A plataforma de testes utilizada é o *software* WEKA (*Waikato Environment for Knowledge Analysis*, Ambiente para Aprendizagem de Máquina de Waikato), uma suite de aplicações de aprendizagem de máquina desenvolvida na Universidade de Waikato na Nova Zelândia. Essa ferramenta é largamente utilizada em projetos nessa área, devido a sua licença aberta (GNU GPL), que permite que seja utilizada quase sem restrições. A figura 6.2 mostra uma captura de tela do programa em execução, mostrando a visualização de um conjunto de dados de exemplo.

Figura 6.2: Janela do módulo Explorer - WEKA 3.6.4



6.4.1 Arquivos ARFF

Para que um conjunto de dados possa ser utilizado no WEKA, ele deve estar em um formato específico, denominado ARFF (*Attribute Relationship File Format*, Formato de Arquivo de Relação de Atributos). Um arquivo nesse formato é apresentado na listagem 6.3. Esse é o mesmo arquivo apresentado na figura 6.2, onde é mostrada a visualização dos dados pela interface gráfica.

Listagem 6.3: Exemplo de arquivo no formato ARFF

```
1       @relation weather
2
3       @attribute outlook {sunny, overcast, rainy}
```

```
4       @attribute temperature real
5       @attribute humidity real
6       @attribute windy {TRUE, FALSE}
7       @attribute play {yes, no}
8
9       @data
10      sunny,85,85,FALSE,no
11      sunny,80,90,TRUE,no
12      overcast,83,86,FALSE,yes
13      rainy,70,96,FALSE,yes
14      rainy,68,80,FALSE,yes
15      rainy,65,70,TRUE,no
16      overcast,64,65,TRUE,yes
17      sunny,72,95,FALSE,no
18      sunny,69,70,FALSE,yes
19      rainy,75,80,FALSE,yes
20      sunny,75,70,TRUE,yes
21      overcast,72,90,TRUE,yes
22      overcast,81,75,FALSE,yes
23      rainy,71,91,TRUE,no
```

Fonte: (HALL et al., 2009)

Um arquivo ARFF é um arquivo de texto simples, em um formato padronizado, que combina a descrição dos dados e os dados em si (HALL et al., 2009). Ele é dividido em duas seções. A primeira, chamada de cabeçalho (*header*) contém o nome do conjunto de dados (*@relation*) e a lista de atributos (colunas) e seus tipos (*@attribute*). Esses arquivos também podem conter comentários, que são ignorados quando o arquivo é analisado, e podem ser utilizados para adicionar informações ao arquivo como fonte, licença, etc. Um comentário é inserido através do caractere '%', e tem efeito até o caractere de nova linha. Após o cabeçalho, vem a seção que contém os dados em si (*@data*).

O nome do conjunto de dados é uma *string* arbitrária, e serve apenas para a identificação na interface (figura 6.2). A lista de atributos é uma sequência de entradas *@attribute*. Cada atributo no conjunto de dados tem uma entrada *@attribute* correspondente, que identifica o atributo. A ordem da declaração deve ser exatamente a ordem em que os atributos são listados na seção de dados. Isso facilita a listagem de dados, onde não é necessário indicar, para cada valor, o atributo ao qual ele se refere. O formato de cada *@attribute* é ilustrado na listagem 6.4.

Listagem 6.4: Formato de uma entrada *@attribute*

```
1  @attribute nome tipo
```

Fonte: (HALL et al., 2009)

O WEKA suporta quatro tipos de atributos. O formato ARFF, no entanto, suporta seis, e a conversão é feita automaticamente:

a) *numeric*: um atributo que pode ser tanto um número inteiro como real.

b) *inteiro*: tratado pelo WEKA como *numeric*.

c) *real*: tratado pelo WEKA como *numeric*.

d) *nominal*: um atributo com valores pré-definidos. Esses valores são especificados através de uma lista no formato: { valor1, valor2, valor3, ... }.

e) *string*: uma *string* de caracteres.

f) *date*: uma data, com um campo opcional que especifica o formato da data, como "yyyy-MM-dd HH:mm:ss".

A seção de dados inicia com a entrada *@data* e segue até o final do arquivo. Nela são listados os valores do conjunto de dados. Cada entrada é representada em uma linha, ou seja, as entradas são separadas pelo caractere de nova linha. Em cada entrada, seus atributos são listados em uma lista separada por vírgulas. Como dito anteriormente, os atributos são identificados pela ordem ordem de declaração, logo ela deve ser a mesma no cabeçalho e nessa seção. Valores ausentes são representados por um caractere de ponto de interrogação (*?*). Os atributos do tipo nominal e *string* são *case sensitive*, e devem ser envolvidos em aspas se contém espaços.

Por padrão, o último atributo é considerado o objetivo ou classe da instância (o atributo que é considerado uma função dos outros atributos). Isso pode ser alterado através da interface de configuração do experimento.

6.4.2 Resultados

A listagem 6.5 mostra um exemplo dos dados de saída após a execução de um teste de um dos algoritmos (LVQ) em um conjunto de dados de teste. Essa saída é dividia em seções. Na primeira seção, "*Run information*", o atributo "*Scheme*" mostra o classificador e todos os parâmetros correspondentes utilizados no treinamento. Os atributos "*Relation*", "*Instances*" e "*Atttributes*" mostram informações do conjunto de dados: o nome, número de instâncias e atributos. Por fim, "*Test mode*" mostra o método utilizado para a execução dos testes. As quatro opções disponíveis são:

1. *Use training set*: Executa os testes utilizando os mesmos dados utilizados no treinamento.

2. *Supplied test set*: Executa os testes utilizando um arquivo de testes fornecido.

3. *Cross-validation*: Executa os testes utilizando k-*fold cross-validation* (seção 5.2.2). O número de *folds* pode ser definido.

4. *Percentage split*: Executa os testes utilizando uma parte dos dados para testes e outra para o treinamento. A porcentagem pode ser definida.

A seção seguinte da saída, "*Classifier model*", mostra dados sobre o processo de treinamento: tempos de execução e informações sobre o modelo gerado pelo classificador. A seção "*Summary*" mostra os resultados dos testes. Os dados dessa seção são: número de instâncias corretamente classificadas, número de instâncias incorretamente classificadas, o coeficiente *kappa*, o erro de aproximação e a raiz quadrada do erro de aproximação, o erro relativo e a raiz quadrada do erro relativo e o número total de instâncias.

A seção "*Detailed class by accuracy*" agrupa os resultados de cada classe presente nos dados: taxa de verdadeiros e falsos positivos, *precision*, *recall* e *f-measure* e a área sob a curva ROC.

A última seção, "*Confusion matrix*", é uma matriz de confusão. As colunas dessa matriz mostram a classificação das instâncias e as linhas, a classe verdadeira. Dessa forma, é possível identificar o número de instâncias correta e incorretamente classificadas. Para um conjunto de dados que contém apenas duas classes, essa tabela também mostra o número de falsos positivos e negativos.

Listagem 6.5: Exemplo de saída de uma execução do WEKA

```
1  === Run information ===
2
3  Scheme:       weka.classifiers.neural.lvq.Lvq1 -M 1 -C 20 -I 1000 -L 1 -R 0.3 \
4      -S 1 -G false
5  Relation:     weather
6  Instances:    14
7  Attributes:   5
8                outlook
9                temperature
10               humidity
11               windy
12               play
13 Test mode:    10-fold cross-validation
14
15 === Classifier model (full training set) ===
16
17 -- Training Time Breakdown --
18 Model Initialisation Time   : 4ms
19 Model Training Time         : 43ms
20 Total Model Preparation Time: 47ms
21
22 -- Cass Distribution --
23 yes :  15 (75%)
24 no :  5 (25%)
```

```
25
26
27
28  Time taken to build model: 0.05 seconds
29
30  === Stratified cross-validation ===
31  === Summary ===
32
33  Correctly Classified Instances           5               35.7143 %
34  Incorrectly Classified Instances         9               64.2857 %
35  Kappa statistic                         -0.4651
36  Mean absolute error                      0.6429
37  Root mean squared error                  0.8018
38  Relative absolute error                135      %
39  Root relative squared error            162.5137 %
40  Total Number of Instances               14
41
42  === Detailed Accuracy By Class ===
43
44                 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
45                   0.556     1          0.5       0.556     0.526       0.278    yes
46                   0         0.444      0         0         0           0.278    no
47  Weighted Avg.    0.357     0.802      0.321     0.357     0.338       0.278
48
49  === Confusion Matrix ===
50
51   a b   <-- classified as
52   5 4 | a = yes
53   5 0 | b = no
```

Fonte: (HALL et al., 2009)

7 DESENVOLVIMENTO DO EXPERIMENTO

Nesse capítulo é apresentado o desenvolvimento do experimento, baseado nos conceitos apresentados nos capítulos anteriores. Em especial, esse capítulo mostra como as ferramentas do capítulo 6 (Planejamento do experimento) foram utilizadas para a comparação dos resultados dos algoritmos apresentados no capítulo 4 (Sistemas Imunológicos Artificiais).

7.1 Preparação dos dados

Os dois conjuntos de dados utilizados (*cr.ger* e *cr.aust*) se encontravam em formato de texto, com os atributos separados por espaços e as instâncias separadas pelo caractere nova linha. As listagens 7.1 e 7.2 mostram as instâncias dos conjuntos de dados no formato do arquivo original (a primeira coluna representa o número da linha, e não faz parte dos dados)[1].

Listagem 7.1: Formato original dos dados (*Cr.Ger*)

```
1  A11 6 A34 A43 1169 A65 A75 4 A93 A101 4 A121 67 A143 A152 2 A173 1 A192 A201 1
2  A12 48 A32 A43 5951 A61 A73 2 A92 A101 2 A121 22 A143 A152 1 A173 1 A191 A201 2
3  A14 12 A34 A46 2096 A61 A74 2 A93 A101 3 A121 49 A143 A152 1 A172 2 A191 A201 1
4  A11 42 A32 A42 7882 A61 A74 2 A93 A103 4 A122 45 A143 A153 1 A173 2 A191 A201 1
5  A11 24 A33 A40 4870 A61 A73 3 A93 A101 4 A124 53 A143 A153 2 A173 2 A191 A201 2
```

Listagem 7.2: Formato original dos dados (*Cr.Aust*)

```
1  1 22.08 11.46 2 4 4 1.585 0 0 0 1 2 100 1213 0
2  0 22.67 7 2 8 4 0.165 0 0 0 0 2 160 1 0
3  0 29.58 1.75 1 4 4 1.25 0 0 0 1 2 280 1 0
4  0 21.67 11.5 1 5 3 0 1 1 11 1 2 0 1 1
5  1 20.17 8.17 2 6 4 1.96 1 1 14 0 2 60 159 1
```

[1]Aqui são mostradas apenas as primeiras cinco instâncias dos conjuntos de dados *Cr.Ger* e *Cr.Aust*. Eles possuem 1000 e 650 instâncias, respectivamente

A preparação dos dados para a importação no WEKA consistiu na adição de uma seção de cabeçalho e da formatação dos dados para valores separados por vírgula (seção 6.4.1) e o resultado é mostrado nas listagens 7.3 e 7.4 (essas listagens mostram apenas as primeiras cinco instâncias na seção de dados).

Listagem 7.3: Arquivo ARFF do *Cr.Ger*

```
1  @relation cr.ger
2
3  @attribute A1  {A11,A12,A13,A14}
4  @attribute A2  numeric
5  @attribute A3  {A30,A31,A32,A33,A34}
6  @attribute A4  {A40,A41,A42,A43,A44,A45,A46,A47,A48,A49,A410}
7  @attribute A5  numeric
8  @attribute A6  {A61,A62,A63,A64,A65}
9  @attribute A7  {A71,A72,A73,A74,A75}
10 @attribute A8  numeric
11 @attribute A9  {A91,A92,A93,A94,A95}
12 @attribute A10 {A101,A102,A103}
13 @attribute A11 numeric
14 @attribute A12 {A121,A122,A123,A124}
15 @attribute A13 numeric
16 @attribute A14 {A141,A142,A143}
17 @attribute A15 {A151,A152,A153}
18 @attribute A16 numeric
19 @attribute A17 {A171,A172,A173,A174}
20 @attribute A18 numeric
21 @attribute A19 {A191,A192}
22 @attribute A20 {A201,A202}
23 @attribute A21 {1,2}
24
25 @data
26 A11,6,A34,A43,1169,A65,A75,4,A93,A101,4,A121,67,A143,A152,2,A173,1,A192,A201,1
27 A12,48,A32,A43,5951,A61,A73,2,A92,A101,2,A121,22,A143,A152,1,A173,1,A191,A201,2
28 A14,12,A34,A46,2096,A61,A74,2,A93,A101,3,A121,49,A143,A152,1,A172,2,A191,A201,1
29 A11,42,A32,A42,7882,A61,A74,2,A93,A103,4,A122,45,A143,A153,1,A173,2,A191,A201,1
30 A11,24,A33,A40,4870,A61,A73,3,A93,A101,4,A124,53,A143,A153,2,A173,2,A191,A201,2
```

Listagem 7.4: Arquivo ARFF do *Cr.Aust*

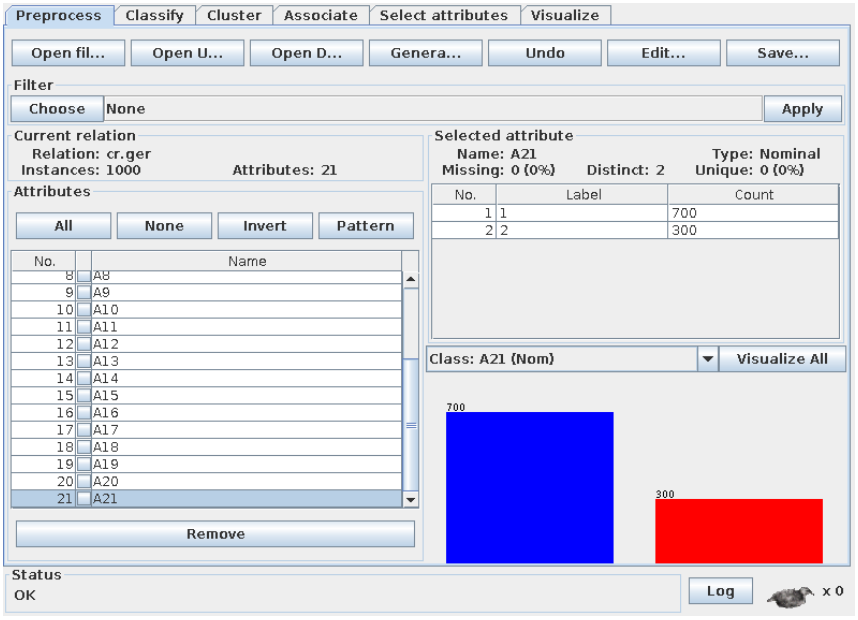
```
1  @relation cr.aust
2
3  @attribute A1  {0,1}
4  @attribute A2  numeric
5  @attribute A3  numeric
6  @attribute A4  {1,2,3}
7  @attribute A5  {1,2,3,4,5,6,7,8,9,10,11,12,13,14}
8  @attribute A6  {1,2,3,4,5,6,7,8,9}
9  @attribute A7  numeric
10 @attribute A8  {1,0}
11 @attribute A9  {1,0}
```

```
@attribute A10 numeric
@attribute A11 {1,0}
@attribute A12 {1,2,3}
@attribute A13 numeric
@attribute A14 numeric
@attribute A15 {0,1}

@data
1,22.08,11.46,2,4,4,1.585,0,0,0,1,2,100,1213,0
0,22.67,7,2,8,4,0.165,0,0,0,0,2,160,1,0
0,29.58,1.75,1,4,4,1.25,0,0,0,1,2,280,1,0
0,21.67,11.5,1,5,3,0,1,1,11,1,2,0,1,1
1,20.17,8.17,2,6,4,1.96,1,1,14,0,2,60,159,1
```

Com os arquivos no formato ARFF, os conjuntos de dados podem ser importados no WEKA. A figura 7.1 mostra o conjunto de dados *Cr.Ger* quando importado. Um conjunto de dados importado pode ser utilizado para qualquer algoritmo que suporte os tipos de atributos contidos nele.

Figura 7.1: Conjunto de dados *Cr.Ger importado no WEKA*



7.2 Utilizando o WEKA

O WEKA é dividido em dois módulos principais: Explorer e Experimenter. Além disso, o WEKA apresenta duas interfaces principais: linha de comando (*command-line interface*, CLI) e gráfica (*graphical user interface*, GUI). A interface gráfica é

mais apropriada para exploração e experimentação, e para a apresentação dos dados, algoritmos e resultados. A interface de linha de comando é mais apropriada para automatização de tarefas e integração com outros sistemas, além de consumir menos recursos. Os dois módulos podem ser usados em qualquer uma das duas interfaces, com as mesmas funcionalidades.

Nesse trabalho foi utilizado o módulo *Experimenter* na interface por linha de comando. O módulo *Experimenter* tem um formato de arquivo de configuração que facilita a definição de experimentos e a repetição e automação dos testes para cada algoritmo usando a interface por linha de comando é muito mais fácil do que se fosse utilizada a interface gráfica. Os exemplos de execução são apresentados conforme devem ser digitados em uma interface de linha de comando, em um emulador de terminal, utilizando um *shell* como *sh* ou *bash*.

7.2.1 Filtros

No WEKA, um filtro é um objeto que recebe um conjunto de dados como entrada e produz um conjunto de dados modificado. Esse é um processo comum da Mineração de Dados, chamado de pré-processamento dos dados: adicionar, remover ou alterar atributos, etc.

Um filtro comum, que é utilizado nesse trabalho, é o de criação de partições para o *cross-validation*. Para esse filtro, são passados três argumentos. O argumento c indica qual dos atributos é o atributo correspondente à classe, e é representado por um índice, iniciado em 1, conforme a declaração na seção de atributos do arquivo de dados (caso o padrão do WEKA seja usado, ou seja, o atributo de classe seja o último da listagem, pode ser utilizado o valor "*last*" como argumento). O argumento N indica o número de partições, e o argumento F indica o índice da partição selecionada.

Além desses, o argumento V pode ser utilizado para gerar o conjunto inverso de seleções, útil para dividir o conjunto em duas partes complementares. Dessa forma, para gerar um conjunto de dados para testes e outro para treinamento, podem ser usados comandos como os da listagem 7.5[2]. Nesse exemplo, são o conjunto de dados é dividido em quatro partições, conforme indicado pelo argumento N. Na primeira linha, a primeira partição é escolhida, conforme indicado pelo argumento F. Na segunda, todas as partições com exceção da primeira são selecionadas, conforme indicado pelo parâmetro V.

[2]Nesses exemplos, é usado o redirecionamento de entrada e saída presentes na maioria dos *shells* UNIX. O caractere "<" seguido de um nome de arquivo indica que aquele arquivo será usado como entrada para o comando. De maneira semelhante, o caractere ">" seguido de um nome de arquivo indica que ele será usado como saída. O WEKA também permite que sejam utilizadas as opções *i* e *o*, respectivamente, para obter os mesmos resultados. No primeiro exemplo, a forma equivalente seria "*-i dataset.arff -o dataset_ test.arff*".

Listagem 7.5: Filtro para geração de partições para *cross-validation*

```
1  java weka.filters.supervised.instance.StratifiedRemoveFolds -c last -N 4 -F 1 \
2      < dataset.arff > dataset_test.arff
3  java weka.filters.supervised.instance.StratifiedRemoveFolds -c last -N 4 -F 1 -V \
4      < dataset.arff > dataset_train.arff
```

7.2.2 Execução de um classificador

A execução de um algoritmo é feita através da classe que implementa o algoritmo no WEKA. Diversas opções podem ser passadas na linha de comando para mudar os parâmetros do algoritmo. Existem algumas opções adicionais para especificar dados adicionais, como os arquivos de dados para treinamento e testes. Além das opções gerais, cada algoritmo pode aceitar diferentes tipo de opções específicas. Como exemplo, para gerar a saída da listagem 6.5, foi utilizado o comando da listagem 7.6.

Listagem 7.6: Execução de um classificador

```
1  java weka.classifiers.neural.lvq.Lvq1 -t data/weather.numeric.arff -i
```

Como é possível ver, a linha de comando é reproduzida no atributo "*Scheme*" na saída. Esse atributo pode ser consultado para executar exatamente o mesmo teste novamente, tornando a reprodução do experimento muito mais fácil. As diversas opções do atributo que não estão presentes na linha de comando são os parâmetros do algoritmo. Como não foram especificados na linha de comando, foram assumidos os valores padrão, que são mostrados na saída. As duas opções que não estão presentes na saída são a opção "*i*", que mostra uma saída mais completa e a opção "*t*", que indica o arquivo que será utilizado como entrada.

Para executar os algoritmos do pacote de algoritmos imunológicos, é necessário informar à máquina virtual a localização do arquivo que contém o código executável. Esse código é disponibilizado em um arquivo *jar*, um tipo de arquivo específico da linguagem java que é semelhante a um arquivo compactado utilizando os programas *tar* ou *zip*.

Para indicar o arquivo, é utilizada a opção *classpath* da máquina virtual. Essa opção pode ser utilizada tanto para indicar um diretório contendo os arquivos compilados (arquivos *.class*) quanto um arquivo *jar*. Alternativamente, o arquivo *jar* pode ser descompactado e o diretório gerado utilizado. Essa opção pode ser passada na linha de comando ou como uma variável de ambiente. Para executar o algoritmo AIRS, por exemplo, é utilizado qualquer um dos comandos da listagem 7.7. A sin-

taxe da opção *classpath* é semelhante ao *path* da maioria dos sistemas operacionais, ou seja, uma lista dos caminhos e arquivos separados pelo caractere ":" (dois-pontos) em sistemas Unix ou ";" (ponto-e-vírgula) em sistemas Windows.

Listagem 7.7: Execução de um algoritmo do pacote de algoritmos imunológicos

```
1  # Opção na linha de comando.
2  java -classpath wekaclassalgos.jar weka.classifiers.immune.airs.AIRS1 # parâmetros
3  # Variável de ambiente.
4  CLASSPATH=wekaclassalgos.jar
5  java weka.classifiers.immune.airs.AIRS1 # parâmetros
```

Combinando os conceitos apresentados nessa seção, a execução de um teste para um algoritmo imunológico utilizando um dos conjuntos de dados é feita de acordo com a listagem 7.8. Aqui, é usado a opção "*t*" para especificar o conjunto de dados *Cr.Ger*. Todas as outras opções são parâmetros do filtro (os valores utilizados são os valores padrão para esse algoritmo).

Listagem 7.8: Execução de um algoritmo do pacote de algoritmos imunológicos utilizando um dos conjuntos de dados

```
1  # Variável de ambiente.
2  CLASSPATH=wekaclassalgos.jar
3  java weka.classifiers.immune.airs.AIRS1 \
4       -S 1 -F 0.2 -C 10.0 -H 2.0 -M 0.1 -R 150.0 -V 0.9 -A -1 -B 1 -E 1 -K 3 \
5       -t german.arff
```

7.2.3 Execução dos classificadores

Para a execução de um teste, pode ser criado um executor genérico de classificadores, com base na listagem 7.9:

Listagem 7.9: Execução genérica de um classificador

```
1  # Variável de ambiente.
2  java -classpath "$classpath" \
3       "$classificador" "$parametros" -t "$conjunto_de_dados" \
4       > "$arquivo_resultados"
```

Dessa forma, é possível utilizar o mesmo comando para execução, variando apenas os valores das variáveis. Os valores utilizados para cada variável foram:

a) *classpath*: conforme explicado na sessão anterior. Esse valor não muda entre as execuções dos testes, mas é mantido como uma variável para que ele possa ser facilmente alterado caso necessário. Esse é o arquivo *jar* do WEKA que contém a implementação dos algoritmos imunológicos (*wekaclassalgos.jar*). Esse arquivo também contém os algoritmos do WEKA, então ele é a única dependência necessária para utilizar os classificadores imunológicos e os que já são incluídos no WEKA.

b) *classificador*: todos os classificadores testados no experimento:

   a) weka.classifiers.immune.airs.AIRS2
   b) weka.classifiers.immune.immunos.Immunos99
   c) weka.classifiers.immune.clonalg.CLONALG
   d) weka.classifiers.functions.MultilayerPerceptron
   e) weka.classifier.functions.SMO
   f) weka.classifiers.meta.AttributeSelectedClassifier
   g) weka.classifiers.trees.J48
   h) weka.classifiers.neural.lvq.Lvq2_1

c) *parâmetros*: os parâmetros são listados junto com cada classificador no arquivo de configuração.

d) *conjunto_de_dados*: o caminho para o arquivo ARFF dos dois conjuntos de dados utilizados.

e) *arquivo_resultados*: arquivo onde os resultados são gravados.

7.2.4 Seleção de parâmetros no WEKA

Conforme descrito no capítulo 5, o desempenho de um algoritmo depende da escolha dos parâmetros utilizados. Como o processo de escolha de parâmetros é repetitivo e fácil de ser automatizado, existem métodos para escolher os parâmetros que geram o melhor valor para os parâmetros de um modelo para um conjunto de dados específico. Um desses métodos, muito utilizado por sua simplicidade e eficácia (embora não seja tão eficiente), é o *grid search*.

No WEKA, existem duas implementações desse método. Ambos fazem parte de uma categoria denominada *meta-classificadores*[3], ou seja, classificadores que atuam sobre a execução de outros classificadores. Esses dois componentes são *GridSearch* e *CVParameterSelection*, e seu funcionamento é semelhante. O processo para utilizá-los para a escolha de parâmetros é o seguinte:

[3]Esses e outros meta-classificadores que o WEKA disponibiliza podem ser encontrados no pacote *weka.classifiers.meta*.

a) Indicar o meta-classificador utilizado.

b) Indicar o classificador real que será utilizado.

c) Listar os parâmetros que serão testados.

d) Listar a faixa de valores para os parâmetros.

e) Indicar o conjunto de dados que será utilizado.

f) Executar o teste.

Com exceção dos itens *a)* e *d)*, o processo é semelhante à execução de um classificador comum. De fato, o meta-classificador executará o classificador da mesma forma que o WEKA faria. No entanto, ao invés de um única execução com parâmetros fixos, esses dois meta-classificadores fazem diversos testes, utilizando todas as combinações possíveis dos valores listados no item *d)* para os parâmetros.

É importante levar em consideração que, conforme discutido anteriormente, esse processo é de grande utilidade para determinar os melhores valores para os parâmetros do classificador, no entanto, uma óbvia desvantagem é o grande aumento no tempo de execução. O número de combinações possíveis cresce exponencialmente com o número de parâmetros e número de elementos na faixa de valores para esses parâmetros.

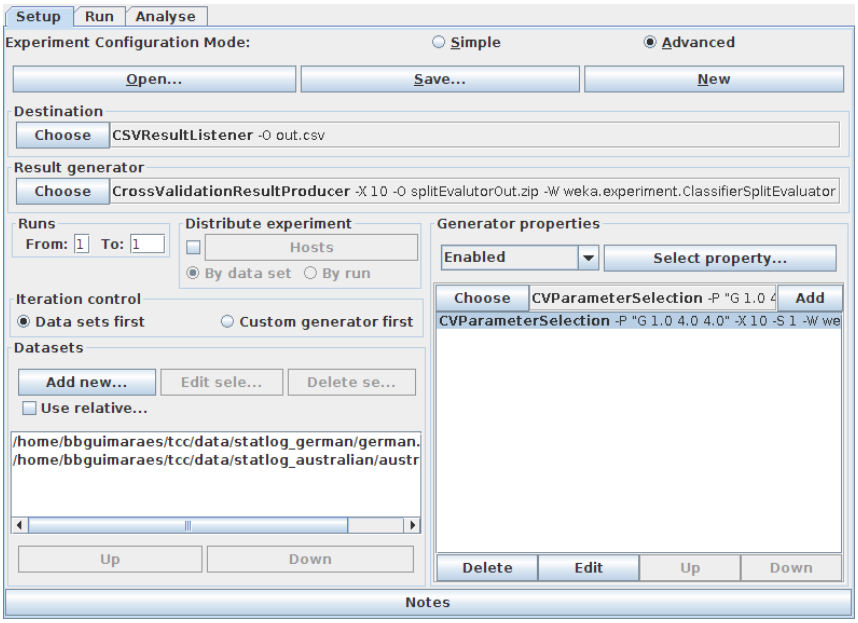
Existem formas de amenizar o efeito da explosão combinatória desses métodos. A opção mais popular aproveita o grande potencial de paralelização do processo de execução do classificador com diferentes parâmetros. Como cada execução é completamente independente das outras, e podem ser executadas em paralelo. Na prática, grandes quantidades de unidades de processamento (denominados *clusters*) são utilizados, e conjuntos de combinações de valores são distribuídas para cada um. Dessa forma, é possível aproveitar ao máximo a quantidade de recursos disponíveis. Os resultados dos testes podem então ser combinados em uma única unidade para a análise final.

7.3 *Experimenter*

Além do módulo *Explorer*, para execução de testes, o WEKA apresenta um módulo para configuração e execução de experimentos, chamado *Experimenter* (figura 7.2). Esse módulo facilita a criação de experimentos, criando um arquivo de configuração onde são descritas as etapas, que pode ser utilizado para reproduzir o experimento usando as mesmas configurações.

A configuração do *Experimenter* pode ser dividida nas seções:

Figura 7.2: Experimenter



1. **Entrada**: a fonte do conjunto de dados. O padrão do WEKA é utilizar arquivos ARFF, mas todos os tipos de arquivos suportados podem ser usados, como arquivos CSV. Também é possível utilizar mais de um conjunto de dados para o mesmo experimento.

2. **Tipo de experimento**: *cross-validation* ou divisão fixa em dados de treino e teste.

3. **Iterações**: número de execuções do experimento.

4. **Algoritmos**: os algoritmos que serão testados. É possível configurar o parâmetros de cada algoritmo exatamente como é feito no *Explorer*.

5. **Saída**: o local onde os resultados do experimento serão gravados. O padrão é um arquivo CSV, mas outros tipos podem ser usados, com arquivos ARFF ou bancos de dados.

Essas configurações podem ser gravadas em um arquivo para que possam ser reutilizadas. O formato desse arquivo pode ser tanto uma representação em formato binário exclusiva do WEKA ou um arquivo CSV.

A partir de um arquivo de configuração é possível executar um experimento usando a interface gráfica. Também é possível utilizar a interface de linha de comando, conforme a listagem 7.10.

Listagem 7.10: Execução de um experimento

```
1  # Flags:
2  #     -l: indica o arquivo de configuração (load).
3  #     -r: executar o experimento (run).
4  java weka.experiment.Experiment -l exp.xml -r
```

7.3.1 Resultados

Conforme mencionado acima, o formato padrão de gravação dos resultados de um experimento pelo *Experimenter* é em arquivos CSV. É gerada uma linha no arquivo com as informações e resultados a cada execução. O número de linha gerado é igual ao número de execuções na configuração do experimento, multiplicado pelo número de *folds* usado no *cross-validation* (caso essa opção tenha sido selecionada), multiplicado pelo número de conjuntos de dados (que pode ser apenas um). Dessa forma, caso o número de execuções seja configurado como "10", o número de *folds* como "10"e sejam utilizados dois conjuntos de dados, serão geradas 200 linhas de resultados no arquivo de saída.

Para cada execução são gravados dados sobre os resultados do treinamento e da classificação das instâncias. Esses dados são os mesmos exibidos quando uma classificação é executada usando o módulo *Explorer* (listagem 6.5). Uma lista completa das variáveis é mostrada na listagem 7.11.

Listagem 7.11: Variáveis gravados pelo *Experimenter*

```
1  Key_Dataset
2  Key_Run
3  Key_Fold
4  Key_Scheme
5  Key_Scheme_options
6  Key_Scheme_version_ID
7  Date_time
8  Number_of_training_instances
9  Number_of_testing_instances
10 Number_correct
11 Number_incorrect
12 Number_unclassified
13 Percent_correct
14 Percent_incorrect
15 Percent_unclassified
16 Kappa_statistic
17 Mean_absolute_error
18 Root_mean_squared_error
19 Relative_absolute_error
20 Root_relative_squared_error
21 SF_prior_entropy
22 SF_scheme_entropy
```

```
23  SF_entropy_gain
24  SF_mean_prior_entropy
25  SF_mean_scheme_entropy
26  SF_mean_entropy_gain
27  KB_information
28  KB_mean_information
29  KB_relative_information
30  True_positive_rate
31  Num_true_positives
32  False_positive_rate
33  Num_false_positives
34  True_negative_rate
35  Num_true_negatives
36  False_negative_rate
37  Num_false_negatives
38  IR_precision
39  IR_recall
40  F_measure
41  Area_under_ROC
42  Weighted_avg_true_positive_rate
43  Weighted_avg_false_positive_rate
44  Weighted_avg_true_negative_rate
45  Weighted_avg_false_negative_rate
46  Weighted_avg_IR_precision
47  Weighted_avg_IR_recall
48  Weighted_avg_F_measure
49  Weighted_avg_area_under_ROC
50  Elapsed_Time_training
51  Elapsed_Time_testing
52  UserCPU_Time_training
53  UserCPU_Time_testing
54  Serialized_Model_Size
55  Serialized_Train_Set_Size
56  Serialized_Test_Set_Size
57  Summary
```

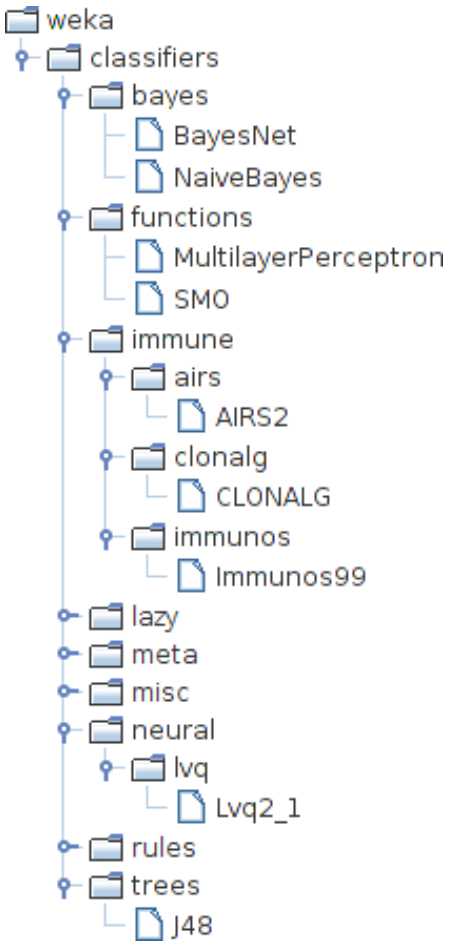
Essas variáveis contém informações sobre o conjunto de dados (nome e número de instâncias de treinamento e testes), classificador (nome e parâmetros), sobre o processo de treinamento e classificação (data, tempo de execução e consumo de memória) e sobre os resultados (diversas medidas de eficácia).

7.4 Seleção de algoritmos

Todos os algoritmos utilizados para comparação com os algoritmos imunológicos fazem parte da distribuição padrão do WEKA (a versão utilizada é a 3.6.9 de 25 de janeiro de 2013). A figura 7.3 mostra os algoritmos na listagem de classificadores do WEKA. Os algoritmos serão descritos brevemente nas próximas seções.

Esses algoritmos podem ser divididos em 5 categorias de algoritmos: árvores de decisão, redes neurais artificiais, redes Bayesianas, *support vector machine* e imunológicos. Na tabela 7.1 são apresentados os mesmo algoritmos, organizados

Figura 7.3: Algoritmos no WEKA



pela categoria a que pertencem.

7.4.1 Redes neurais artificiais

Um algoritmo que utiliza uma rede de neurônios artificiais inspirados no modelo dos neurônios biológicos do sistema nervoso. Simula uma rede de neurônios usando um modelo computacional baseado nas conexões entre os neurônios.

7.4.1.1 MultilayerPerceptron

Uma implementação de uma rede neural multicamadas que usa o algoritmo *back-propagation*, proposto por Paul Werbos (WERBOS, 1974). A aprendizagem acontece através de repetidas iterações e ajustes dos pesos das conexões entre neurônios. Pode ser treinada de forma automática, mas o WEKA oferece uma interface onde os pesos da rede podem ser alterados manualmente.

a) **Parâmetros**:

Tabela 7.1: Algoritmos utilizados para comparação

Categoria	**Algoritmos**
Árvores de decisão	ID3
	C4.5 (J48)
Redes neurais artificiais	Multilayer Perceptron
	Learning Vector Quantization (Lvq2_1)
Redes Bayesianas	Naive Bayes
	Bayes Net
Support Vector Machine	Sequential Minimal Optimization
Imunológicos	AIRS
	Immunos
	Seleção Clonal (CLONALG)

- **L**: taxa de aprendizagem.
- **M**: taxa de *momentum*.
- **N**: número de *epochs* (uma iteração do processo de treinamento e teste).
- **V**: tamanho do conjunto de validação do treinamento.
- **S**: semente do gerador de números aleatórios.
- **E**: número máximo de erros consecutivos para término da validação da rede.
- **G**: abre a interface de criação da rede.
- **A**: não cria as conexões da rede.
- **B**: não aplica o filtro NominalToBinary.
- **H**: parâmetro para configuração das camadas escondidas da rede.
- **C**: desabilita a aplicação de normalização no atributo de classe.
- **I**: desabilita a aplicação de normalização em todos os atributos.
- **R**: desabilita a reinicialização da rede durante o treinamento.
- **D**: habilita a diminuição gradual da taxa de aprendizagem.

b) **Parâmetros testados**:

- **L**: 0.1, 0.2, 0.3, 0.4, 0.5.

7.4.2 Árvores de decisão

Algoritmo supervisionado que monta uma árvore ligando observações sobre os dados à conclusões sobre esses dados. É um tipo de algoritmo preditivo muito usado em sistemas especialistas e aprendizagem de máquina por gerar um modelo que pode ser facilmente analisado e ajustado.

7.4.2.1 ID3

Algoritmo de árvores de decisão usado em aprendizagem de máquina, desenvolvido por Ross Quinlan (QUINLAN, 1986). Divide conjunto de atributos dos dados em subconjuntos iterativamente, separando sempre pelo atributo com menor entropia (maior ganho de informação).

a) **Parâmetros**: esse algoritmo não aceita parâmetros.

b) **Filtros**: como o ID3 não suporta atributos numéricos, foi necessário discretizar os valores numéricos (seção 7.2.1).

7.4.2.2 C4.5

Uma extensão do algoritmo ID3, desenvolvido pelo mesmo autor (QUINLAN, 1993). Entre as melhorias do algoritmo estão o tratamento mais eficiente de dados contínuos, o tratamento a dados incompletos e a poda da árvore após o treinamento. O WEKA tem uma implementação *open source* desse algoritmo chamado J48.

a) **Parâmetros**:

- **U**: usa uma árvore sem poda.
- **C**: limiar de confiança para poda.
- **M**: número mínimo de instâncias por folha.
- **R**: reduz o número de instâncias usadas na poda pós-execução.
- **N**: número de partições para a poda pós-execução.
- **B**: realiza apenas divisões binárias em atributos numéricos, ao invés de divisões de tamanho arbitrário.
- **S**: desativa a poda baseada em elevação de ramos da árvore.
- **L**: desativa o processo de otimização de memória.
- **A**: aplica o algoritmo de suavização de Laplace às probabilidades da árvore.
- **Q**: semente do gerador de números aleatórios.

b) **Parâmetros testados**:

- **C**: 0.1, 0.2, 0.3, 0.4, 0.5.
- **M**: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.

7.4.3 Learning Vector Quantization (LVQ)

Algoritmo de classificação baseado no *Vector Quantization* (*VQ*), um algoritmo que distribui as instâncias no espaço de estados e iterativamente aproxima as instâncias semelhantes, até que sejam formados *clusters* com as instâncias classificadas (KOHONEN, 1997).

7.4.3.1 Lvq2_1

Algoritmo presente no pacote de algoritmos de Jason Bronwlee. Uma implementação do algoritmo LVQ onde duas instâncias são analisadas a cada iteração, sendo atualizadas quando uma pertence à classe desejada e a outra não e a distância se encontra dentro de uma faixa pré-definida (BRONWLEE, 2011).

a) **Parâmetros**:

- **M**: modo de inicialização dos vetores.
- **L**: tipo de função para a taxa de aprendizagem.
- **R**: valor inicial da taxa de aprendizagem.
- **C**: número total de vetores no modelo.
- **I**: número total de iterações.
- **G**: seleção dinâmica da classe de cada instância.
- **W**: limiar de semelhança entre as instâncias comparadas.

b) **Parâmetros testados**:

- **R**: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0.
- **W**: 0.1, 0.2, 0.3, 0.4, 0.5.

7.4.4 Redes Bayesianas

Um modelo probabilístico desenvolvido usando os modelos derivados do teorema de Bayes (BAYES, 1763). O modelo gerado tem a forma de um grafo direcionado acíclico, onde os nodos são variáveis e as arestas as dependências condicionais entre elas (PEARL, 1988).

7.4.4.1 Naive Bayes

Um classificador que cria redes bayesianas simples, onde cada atributo do conjunto de dados é analisado independentemente do outro. O resultado é um modelo onde os atributos não influenciam uns aos outros. Por causa dessa limitação, o treinamento desse algoritmo é muito eficiente.

a) **Parâmetros**:

- **K**: usa Kernel Density Estimator ao invés de distribuição normal para atributos numéricos.
- **D**: usa discretização supervisionada para processar atributos numéricos.

7.4.4.2 Bayes Net

Implementação do algoritmo padrão de redes bayesianas. Permite a escolha da medida de avaliação e do método de busca no espaço de estados utilizados pelo classificador.

a) **Parâmetros**:

- **D**: utiliza ADTree, aumentado a velocidade de treinamento mas consumindo mais recursos.
- **Q**: método para busca na rede.
- **E**: método para encontrar as tabelas de probabilidade condicional.

b) **Filtros**: *weka.filter.supervised.attribute.Discretize*.

7.4.5 Support Vector Machine (SVM)

Esse classificador considera as instâncias como pontos em um espaço de estados e tenta traçar uma linha que separe esses pontos em duas regiões do espaço. Por isso, é considerado um classificador binário linear. Classifica uma nova instância posicionando-a no espaço de estados e identificando em que lado da linha separadora essa instância se encontra (CORTES, VAPNIK, 1995).

7.4.5.1 Sequential Minimal Optimization (SMO)

Algoritmo iterativo para o problema da otimização de uma máquina de vetor de suporte. A implementação do WEKA é baseada no algoritmo por John Platt (PLATT, 1998).

a) **Parâmetros**:

- **C**: a constante de complexidade das regras de decisão.
- **N**: opção para normalizar, uniformizar ou usar os dados originais.
- **L**: o parâmetro de tolerância.
- **P**: *epsilon* para erro de arredondamento.
- **M**: aplica regressão logística aos pesos.
- **W**: número de partições para o *cross-validation* interno.

- **K**: o *kernel* que será utilizado.

b) **Parâmetros testados**:

- **C**: 1, 2, 3, 4, 5.

7.4.6 Algoritmos imunológicos

7.4.6.1 AIRS

a) **Parâmetros**:

- **F**: controla o limiar de afinidade para substituição de células de memória.
- **C**: taxa de clonagem.
- **H**: taxa de hiper-mutação.
- **K**: número de células de memória usadas durante a classificação de novas instâncias.
- **E**: número inicial de células de memória.
- **A**: número de células usadas no cálculo do limiar de afinidade.
- **V**: limite para o refinamento das células de memória.
- **R**: número máximo de células.

7.4.6.2 Immunos

a) **Parâmetros**:

- **G**: número de gerações.
- **E**: limiar mínimo de afinidade.
- **S**: porcentagem de cada grupo de antígenos usada na inicialização da população de células B.

b) **Parâmetros testados**:

- **G**: 1, 2, 3, 4.

7.4.6.3 CLONALG

a) **Parâmetros**:

- **N**: tamanho do *pool* de anticorpos.
- **B**: taxa de clonagem.
- **G**: número de gerações.
- **R**: número de anticorpos mantidos entre cada geração.

- **n**: número de anticorpos selecionado para clonagem e mutação a cada geração.
- **d**: número de anticorpos descartado e substituído a cada geração.

7.5 Problemas encontrados

Durante a preparação e execução dos testes, dois problemas principais foram encontrados, ambos no pacote de algoritmos imunológicos usado. Apesar de nenhum impedir a conclusão do trabalho, ambos são dignos de nota.

O primeiro foi um problema na definição dos parâmetros do CLONALG. Como foi mostrado no capítulo 6, o WEKA permite a passagem de diversas opções na linha de comando para alterar o seu modo de funcionamento. Essas opções são na forma de *flags*, como é padrão em interfaces de linha de comando. Um desses parâmetros é usado para indicar que o modelo previamente gerado deve ser lido de um arquivo, ao invés de criado novamente (a listagem 7.12 mostra a definição do parâmetro pelo próprio WEKA).

Listagem 7.12: Definição do parâmetro “d” no wEKA

```
1  -d <name of output file>
2       Sets model output file. In case the filename ends with '.xml',
3       only the options are saved to the XML file, not the model.
```

No entanto, um dos parâmetros do CLONALG também usa a letra “d” como representação (conforme apresentado na seção anterior, o parâmetro é o número de anticorpos substituído a cada geração). Isso gerava um conflito na execução do algoritmo, já que as opções do WEKA são analisadas antes das opções do algoritmo. O resultado era uma mensagem de erro apontando que o parâmetro “d” não tinha sido passado.

Para solucionar esse problema, foi feita uma alteração diretamente no código do algoritmo. As opções do algoritmo são definidas em um vetor na linha 37 do arquivo $src/weka/classifiers/immune/clonalg/CLONALG.java$ (listagem 7.13). A alteração para a solução foi alterar o nome do parâmetro "d"para uma letra que não conflitasse com outros parâmetros do WEKA ou do algoritmo. Após isso, o código do WEKA foi compilado e essa versão alterada foi utilizada na execução dos testes.

Listagem 7.13: Código fonte original do CLONALG

```
1  private final static String [] PARAMETERS = {
2      "B", "N", "n", "d", "G", "S", "R"
3  };
```

O segundo problema encontrado foi no algoritmo AIRS. Conforme o trabalho de Jason Brownlee (BROWNLEE, 2005), a condição de parada do algoritmo é:

> The stop condition for this process of ARB refinement occurs when the mean normalised stimulation is more than the user defined stimulation threshold.
>
> [...]
>
> Stimulation Threshold – As mentioned, the stopping criterion to the ARB refinement process is when the mean normalised stimulation value is above the stimulation threshold. This parameter controls the amount of refinement performed on ARBs for an antigen, and thus how closely the ARBs will be to the antigen in question. Stimulation values are commonly high, around 0.9. This means that the mean stimulation value must be quite high, that is the vast majority of the ARBs in the pool must be similar to the antigen. The range for the stimulation threshold must obviously be in the range of [0,1], given the mean also will have the same range.

Como pode ser visto, o algoritmo não tem um número de iterações definido. Como a maioria dos algoritmos inspirados no sistema imunológico, o algoritmo itera até que o sistema entre em equilíbrio. O parâmetro *stimulation threshold* (limiar de similaridade) pode ser usado para tornar esse teste mais ou menos rigoroso, conforme a necessidade. Valores mais altos geram um modelo mais sensível, mas podem aumentar o tempo de treinamento.

Utilizando o valor padrão do parâmetro (0.9), os testes demoravam um tempo proibitivo para serem executados. Foram feitos testes utilizando vários valores, para determinar um valor que fosse aceitável dado o tempo disponível para a execução dos experimentos. Os resultados desses testes são mostrados na tabela **??**. Cada linha representa uma execução, onde o parâmetro *stimulation threshold* assumiu o valor indicado e todos os outros tinham o valor padrão do algoritmo. Ao lado de cada linha, é mostrado o tempo de execução do teste.

O tempo de execução aumentava exponencialmente conforme o valor do parâmetro alcançava o valor padrão de 0.9. Por isso, para os testes, foi usado o valor 0.8 para esse parâmetro, que é baixo o suficiente para manter o tempo de execução em uma faixa aceitável mas alto o suficiente para não comprometer os resultados.

Tabela 7.2: Tempo de execução dos testes do AIRS

Valor	**Tempo de execução**
0.5	0m1.158s
0.75	0m2.148s
0.8	0m4.054s
0.85	0m14.263s
0.875	0m27.451s
0.8875	0m49.042s
0.89375	2m52.582s
0.896875	92m5.210s

7.6 Resultados

Os testes foram executados para todos os algoritmos, usando as configurações descritas nas seções anteriores. A tabela 7.3 mostra um resumo das execuções. As colunas dessa tabela representam, da esquerda para a direita, a categoria do algoritmo, o algoritmo, o número de execuções do experimento, o número de *folds* utilizado no *cross-validation*, o número total de execuções do algoritmo e os parâmetros testados. Essa última coluna utiliza a sintaxe do *CVParameterSelection* para descrição dos parâmetros, onde são indicados o nome do parâmetro, os valores mínimos e máximos que serão testados e o número de valores testados.

Tabela 7.3: Resumo das execuções

Método	**Algoritmo**	**Runs**	**Folds**	**Execuções**	**Parâmetros testados**			
					Nome	**Min.**	**Max.**	**n**
Redes neurais	MultilayerPerceptron	10	10	200	-			
Árvores de decisão	ID3	10	10	200	-			
	J48	10	10	200	M	1	10	10
					C	0.1	0.5	5
LVQ	Lvq2_1	10	10	200	R	0.1	1	10
					W	0.1	0.5	5
Redes bayesianas	NaiveBayes	10	10	200	-			
	BayesNet	10	10	200	-			
SVM	SMO	10	10	200	C	1	5	5
Imunológicos	AIRS	10	10	200	-			
	Immunos	1	10	20	G	1	4	4
	CLONALG	10	10	200	G	5	20	4
					B	0.1	0.5	5

Todos os testes foram executados em uma mesma máquina, serialmente, e a máquina não foi utilizada para nenhum outro processamento durante os testes. Isso foi feito para que não houvesse interferência externa ou interna nos resultados. As configurações da máquina relevantes para o experimento são mostradas na tabela 7.4. É importante notar nesse tipo de experimento que é impossível isolar um sistema computacional completamente.

Uma técnica comumente usada é executar os testes repetidas vezes, escolhendo para comparação aquele que tiver o melhor resultado. Isso foi feito nesse trabalho através de uma funcionalidade do próprio módulo *Experimenter* do WEKA, que permite configurar o número de execuções do experimento. Os resultados de todas as execuções são gravados no arquivo de saída e o melhor foi escolhido na coleta dos resultados.

Foi utilizado o código padrão presente na versão 3.6.1 do WEKA. As únicas exceções são as descritas na seção 7.5, mas essas alterações não têm impacto sobre a performance dos algoritmos.

Tabela 7.4: Configurações da máquina onde os testes foram executados

Máquina de testes	
Tipo	Máquina virtual (virtualbox)
Sistema operacional	Ubuntu 12.04 LTS 32-bit
Processador	Intel Pentium E2200 @2.20GHz
Memória	DDR2 800MHz 1024Mb

7.6.1 Resultados por conjunto de dados

Nessa seção são apresentadas as tabelas com os resultados dos testes agrupados pelos conjuntos de dados (tabelas 7.5 à 7.10). A separação por conjunto de dados facilita a comparação dos valores, já que há uma clara separação entre os resultados de cada um. A próxima seção faz uma análise dos resultados para cada algoritmo.

As tabelas 7.5 e 7.6 mostram o percentual de instâncias corretamente classificadas e o erro médio, ordenadas pelo primeiro[4]. Geralmente, um valor maior de instâncias classificadas corretamente implica um erro menor, mas esse nem sempre é o caso. Para classificadores que associam um percentual de certeza à classificação, o erro médio indica *quão* errada a classificação foi, e não apenas se ele foi errada ou

[4]Os valores para o conjunto de dados *cr.ger* podem parecer estranhos, mas isso se deve à maneira como os dados foram divididos. O número total de instâncias é 1000, e foi utilizado *cross-validation* de 10 *folds*, com 100 instâncias cada. Dessa forma, a porcentagem de instâncias corretamente classificadas é sempre um número inteiro.

não. Essa medida pode ser mais ou menos importante, dependendo do objetivo da classificação.

Tabela 7.5: Porcentagem correta e error médio (cr.aust)

Algoritmo	**Porcentagem correta**	**Erro quadrático médio**
J48	95.65%	0.2364105060
MultilayerPerceptron	94.20%	0.2497022089
BayesNet	94.20%	0.2264276254
SMO	92.75%	0.2691909510
AIRS	92.75%	0.2691909510
Immunos	91.30%	0.2948839123
NaiveBayes	88.41%	0.3021207800
ID3	86.96%	0.3096584495
Lvq2_1	79.71%	0.4504426165
CLONALG	76.81%	0.4815434123

Tabela 7.6: Porcentagem correta e erro médio (cr.ger)

Algoritmo	**Porcentagem correta**	**Erro quadrático médio**
NaiveBayes	84.00%	0.3655059498
BayesNet	84.00%	0.3553319820
SMO	83.00%	0.4123105626
MultilayerPerceptron	81.00%	0.4251917863
J48	80.00%	0.3856184540
Lvq2_1	76.00%	0.4898979486
Immunos	75.00%	0.5000000000
CLONALG	75.00%	0.5000000000
AIRS	74.00%	0.5099019514
ID3	71.00%	0.5248906592

As tabelas 7.7 e 7.8 mostram o índice de Youden e as taxas de falsos positivos e falsos negativos, ordenadas pelo primeiro. O índice de Youden é um cálculo que combina a taxa de falsos positivos e negativos, dando maior relevância onde os dois valores são menores. Nos resultados do WEKA, a sensibilidade é chamada de taxa de verdadeiros positivos. A especificidade não faz parte dos valores, mas pode ser calculada como $1 - taxa\ de\ falsos\ positivos$.

Tabela 7.7: Falsos positivos e falsos negativos (cr.aust)

Algoritmo	**Youden**	**Falsos positivos**	**Falsos negativos**
J48	0.9091680815	0.0645161290	0.0263157895
MultilayerPerceptron	0.8820512821	0.0666666667	0.0512820513
BayesNet	0.8820512821	0.0666666667	0.0512820513
SMO	0.8717948718	0.0000000000	0.1282051282
AIRS2	0.8641025641	0.0333333333	0.1025641026
Immunos99	0.8307692308	0.0666666667	0.1025641026
Id3	0.7595048630	0.1379310345	0.1025641026
NaiveBayes	0.7410256410	0.2333333333	0.0256410256
Lvq2_1	0.5487179487	0.4000000000	0.0512820513
CLONALG	0.5314091681	0.2580645161	0.2105263158

As tabelas 7.9 e 7.10 mostram os tempos de treinamento e teste, ordenados pelo primeiro. Em todos os casos, o tempo de treinamento é muito superior ao tempo de teste. Isso é comum na maioria dos algoritmos. É importante notar que o tempo de treinamento é sempre constante, enquanto outros têm um tempo alto de treinamento inicial, mas os treinamentos subsequentes têm um custo muito menor. Esse fator não foi avaliado nesse experimento.

Uma tendência que já pode ser vista nas tabelas e que é comum em todos os dados do experimento é a de que os resultados para o conjunto de dados *cr.aust* são sempre melhores em relação ao *cr.ger*. O desempenho é sempre cerca de 1.5 vezes "melhor" no primeiro do que no segundo. Apenas com os resultados desse experimento não é possível indicar com certeza o motivo desse comportamento, mas imagina-se que isso se deve ao fato do conjunto de dados *cr.ger* ser maior (600 contra 1000 instâncias) e possuir mais atributos (14 contra 20).

Isso também ajuda a mostrar a importância da utilização de mais de um con-

Tabela 7.8: Falsos positivos e falsos negativos (cr.ger)

Algoritmo	**Youden**	**Falsos positivos**	**Falsos negativos**
BayesNet	0.5809523810	0.3333333333	0.0857142857
NaiveBayes	0.5619047619	0.3666666667	0.0714285714
SMO	0.5476190476	0.3666666667	0.0857142857
MultilayerPerceptron	0.5000000000	0.4000000000	0.1000000000
J48	0.4095238095	0.5333333333	0.0571428571
AIRS2	0.3238095238	0.5333333333	0.1428571429
Immunos99	0.3000000000	0.6000000000	0.1000000000
Lvq2_1	0.2952380952	0.6333333333	0.0714285714
Id3	0.2862745098	0.5666666667	0.1470588235
CLONALG	0.1666666667	0.8333333333	0.0000000000

Tabela 7.9: Tempos de treinamento e teste em segundos (cr.aust)

Algoritmo	**Tempo de treinamento**	**Tempo de teste**
NaiveBayes	0.002	0.001
BayesNet	0.029	0.000
ID3	0.032	0.005
J48	2.715	0.000
Lvq2_1	5.383	0.001
MultilayerPerceptron	17.993	0.003
AIRS	23.238	0.012
SMO	57.207	0.001
Immunos	69.876	0.007
CLONALG	175.078	0.002

junto de dados na execução de qualquer experimento. Utilizando apenas um, não é possível saber se o modelo gerado terá o mesmo desempenho em conjuntos de dados diferentes, com características diferentes, como número de instâncias e atributos,

Tabela 7.10: Tempos de treinamento e teste em segundos (cr.ger)

Algoritmo	**Tempo de treinamento**	**Tempo de teste**
NaiveBayes	0.004	0.001
ID3	0.089	0.000
BayesNet	0.176	0.007
Lvq2_1	4.214	0.001
J48	5.480	0.000
AIRS	15.037	0.034
MultilayerPerceptron	58.003	0.012
SMO	118.713	0.001
Immunos	199.922	0.017
CLONALG	331.366	0.003

tipo de dados, distribuição dos valores, etc.

Com mais de um conjunto de dados é possível comparar o desempenho do algoritmo em diferentes situações, evitando análises incompletas ou tendenciosas. Como pode ser observado nos gráficos, o desempenho de alguns algoritmos foi consideravelmente melhor em um conjunto de dados que no outro. Caso apenas um dos conjuntos de dados tivesse sido usado, isso poderia favorecer alguns algoritmos ou penalizar outros.

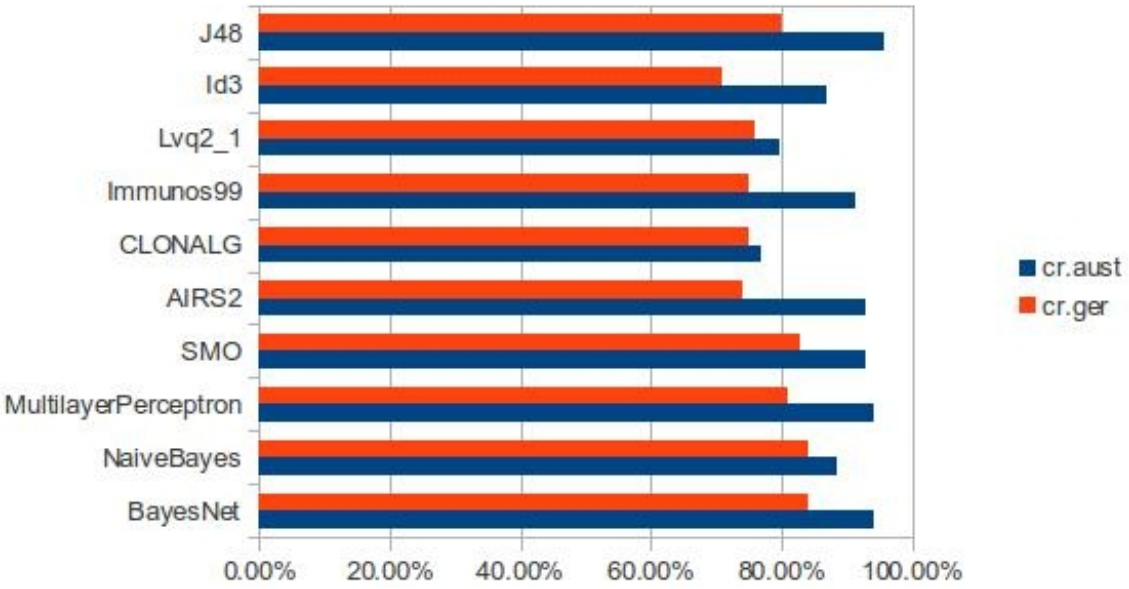
7.6.2 Resultados por algoritmo

Nessa seção são apresentados gráficos com os resultados dos testes agrupados pelos algoritmos (figuras 7.4 à 7.10). Após a análise de cada conjunto de dados, essa análise permite comparar os resultados entre os algoritmos. Os valores comparados são os mesmos (porcentagem de instâncias corretamente classificadas, erro quadrático médio, taxa de falsos positivos e falsos negativos e tempos de treinamento e testes). Dessa forma, além da comparação entre os gráficos dessa seção, é possível fazer a comparação com as tabelas da seção anterior.

O gráfico 7.4 mostra a porcentagem de instâncias corretamente classificadas. Esse é o primeiro valor a ser considerado nas análises de desempenho, por ser uma medida simples. Com relação a esse atributo, a maioria dos classificadores teve resultado semelhante. A diferença entre o melhor e o pior dos classificadores foi

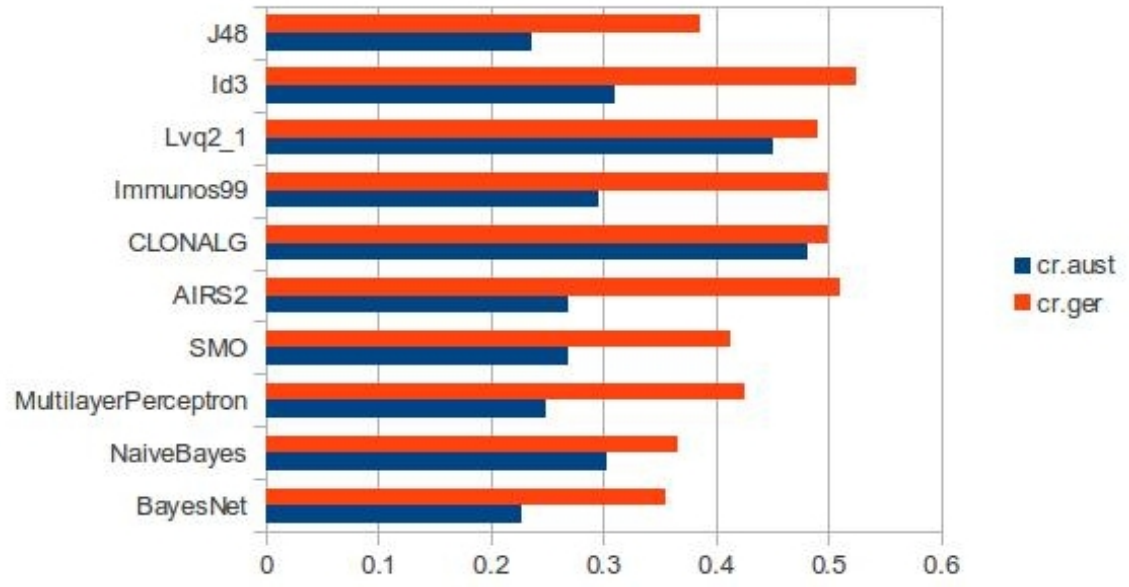
de 19% e 14% por cento para o *cr.aust* e *cr.ger*, respectivamente. Ainda, se forem desconsiderados os dois piores resultados, a diferença cai para 9% em ambos.

Figura 7.4: Porcentagem correta



Em destaque na porcentagem de instâncias corretamente classificadas ficaram os algoritmos BayesNet, MultilayerPerceptron e SMO. Os algoritmos imunológicos tiveram desempenho mediano no *cr.aust* e menor que a média no *cr.ger*.

Figura 7.5: Erro quadrático médio

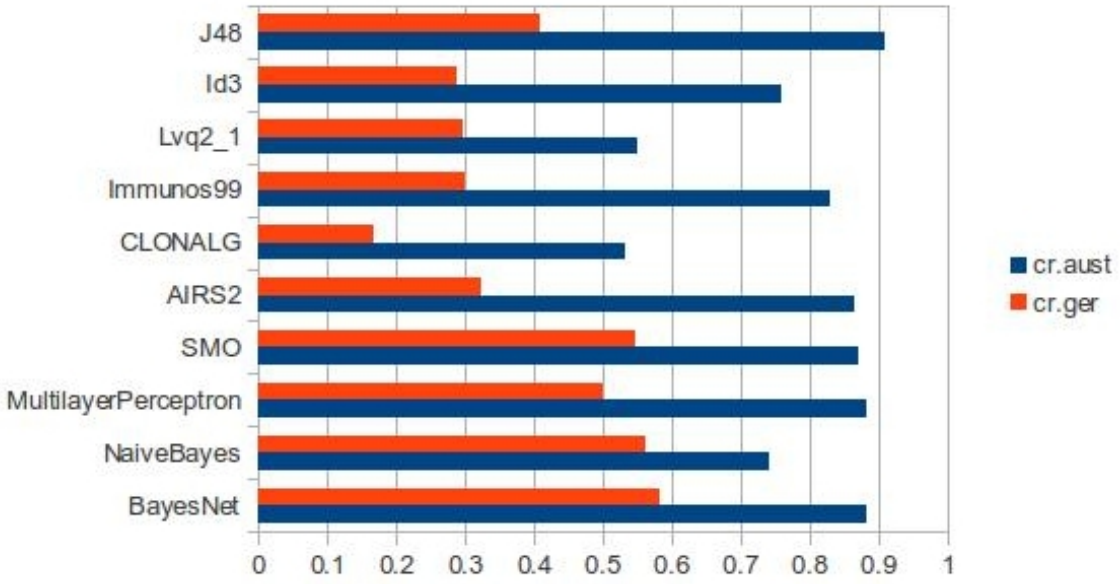


O gráfico 7.5 mostra o erro quadrático médio. Não houveram diferenças significativas entre os resultados do gráfico anterior e desse. Um classificador que classificou

mais instâncias corretamente teve um erro médio menor. Destaca-se nesse gráfico a diferença maior do erro médio entre os dois conjuntos de dados, que foi maior que a diferença entre o número de instâncias classificadas corretamente.

É importante lembrar que o erro médio é uma medida útil para comparação entre os valores de uma mesma variável, mas não pode ser usado para comparação entre variáveis (como entre os resultados dos conjuntos de dados), porque a sua escala varia conforme a escala dos valores da variável.

Figura 7.6: Índice de Youden



Os gráficos do índice de Youden, falsos positivos e falsos negativos (7.6, 7.7 e 7.8) e as matrizes de confusão de cada algoritmo (tabelas **??** à **??**) são os mais significativos para essa análise. Como foi comentado no capítulo 5, apenas a porcentagem de instâncias classificadas corretamente não é uma medida de suficiente para a avaliação do desempenho de um algoritmo.

Para que a análise seja completa, é necessário analisar o que os falsos positivos e falsos negativos significam no contexto da detecção de fraude. Um falso positivo indica que a instância foi classificada como uma fraude quando na verdade ela não era. Por outro lado, um falso negativo indica que a instância foi classificada como normal quando na verdade era uma fraude[5]. No contexto da detecção de fraude, um falso negativo é muito mais crítico que um falso positivo, ou seja, quanto mais fraudes forem identificadas, melhor. O número de instâncias normais identificadas como fraude têm bem menos importância nesse contexto.

[5]A terminologia pode ser confusa nesse contexto. Em um sistema de detecção de fraude, "positivo"significa "positivo para fraude", e não que a instância é "boa"

Figura 7.7: Taxa de falsos positivos

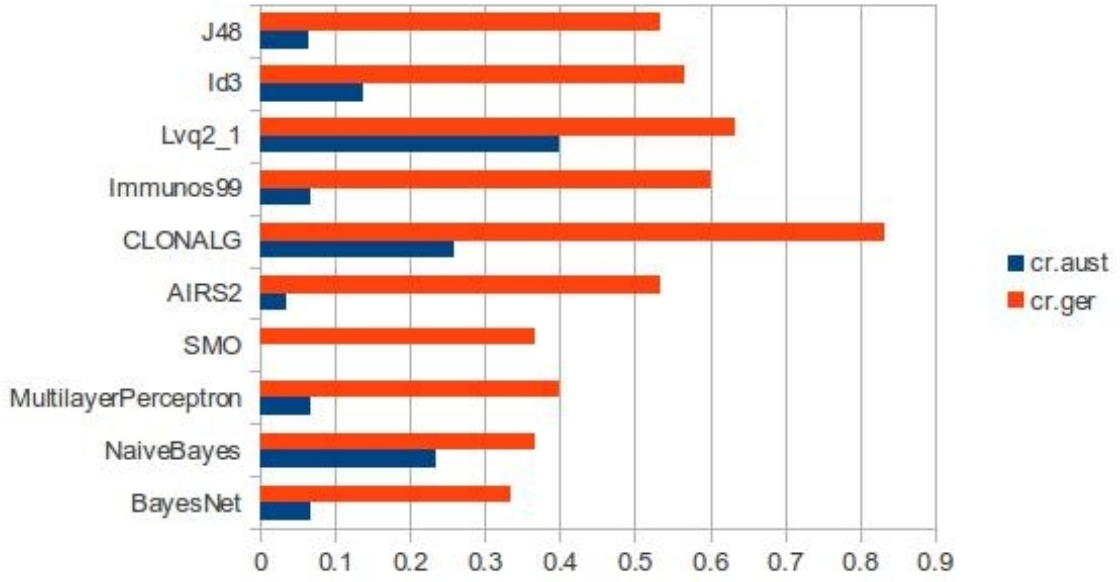
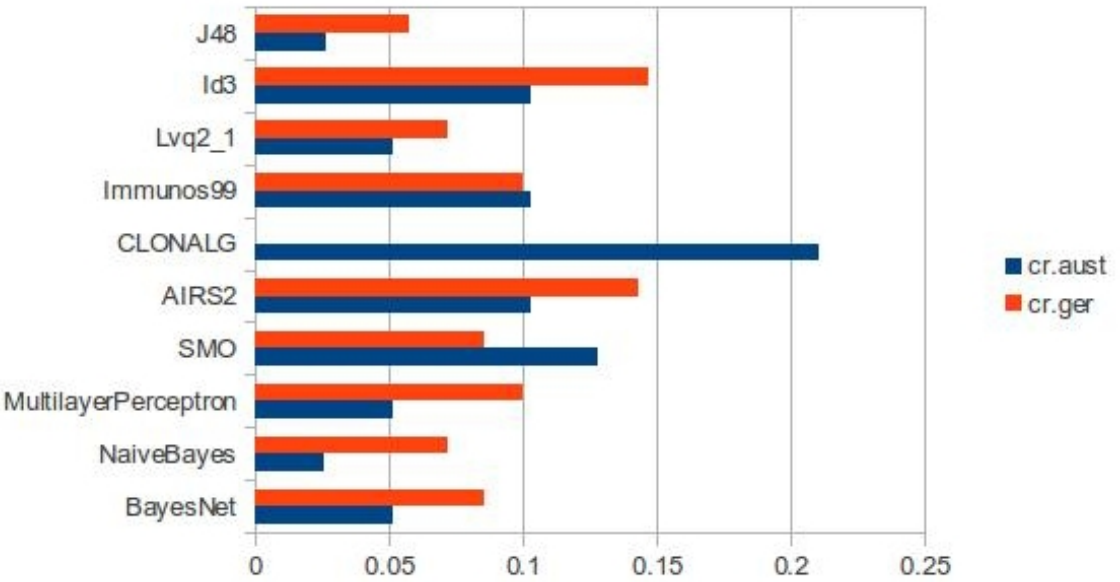


Figura 7.8: Taxa de falsos negativos



Desse ponto de vista, um algoritmo como o CLONALG, que classificou corretamente apenas 75% das instâncias no conjunto de dados *cr.ger* mas que teve taxa de falsos positivos 0 (ou seja, todas as fraudes foram identificadas) pode ser mais interessante do que as redes bayesianas, que classificaram 9% instâncias corretamente a mais, mas tiveram taxa de falsos negativos entre 5% e 10%.

Dependendo do contexto onde o sistema será aplicado, mesmo uma taxa de falsos negativos de 10% pode ser inaceitável. Essa restrição não é incomum e eliminaria

Tabela 7.11: BayesNet

	cr.aust		cr.ger	
	P	N	P	N
P	37	2	64	10
N	28	2	20	6

Tabela 7.12: NaiveBayes

	cr.aust		cr.ger	
	P	N	P	N
P	38	7	65	11
N	23	1	19	5

Tabela 7.13: MultilayerPerceptron

	cr.aust		cr.ger	
	P	N	P	N
P	37	2	63	12
N	28	2	18	7

Tabela 7.14: SMO

	cr.aust		cr.ger	
	P	N	P	N
P	34	0	64	11
N	30	5	19	6

Tabela 7.15: AIRS

	cr.aust		cr.ger	
	P	N	P	N
P	35	1	60	16
N	29	4	14	10

Tabela 7.16: CLONALG

	cr.aust		cr.ger	
	P	N	P	N
P	30	8	70	25
N	23	8	5	0

Tabela 7.17: Immunos

	cr.aust		cr.ger	
	P	N	P	N
P	35	2	63	18
N	28	412	7	

Tabela 7.18: Lvq2_1

	cr.aust		cr.ger	
	P	N	P	N
P	37	12	65	19
N	18	2	11	5

Tabela 7.19: ID3

	cr.aust		cr.ger	
	P	N	P	N
P	35	4	58	17
N	25	4	13	10

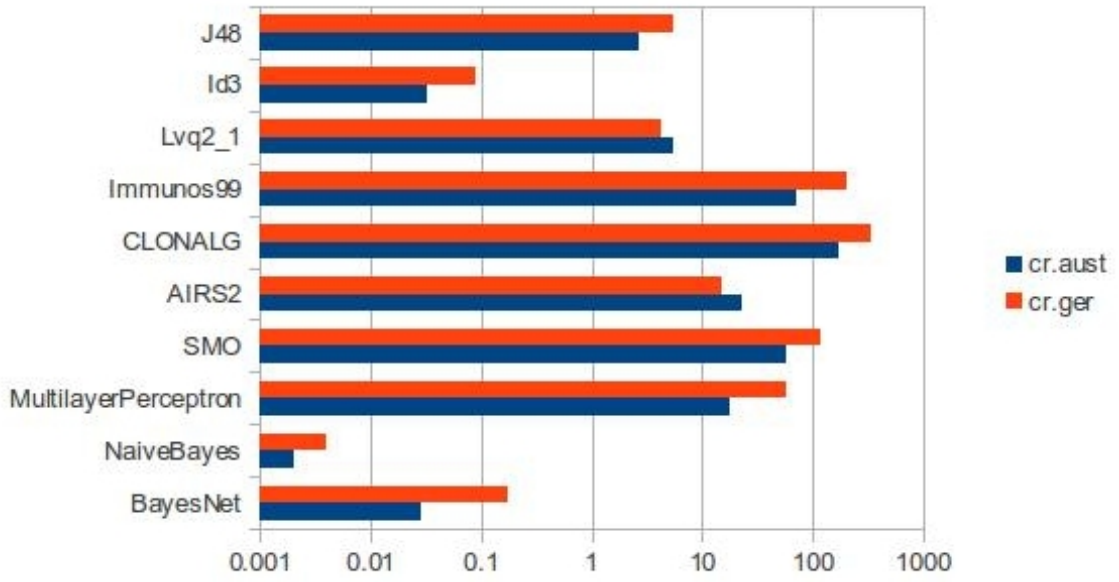
Tabela 7.20: J48

	cr.aust		cr.ger	
	P	N	P	N
P	37	2	66	16
N	29	1	14	4

quase todos os algoritmos testados, baseado nos resultados apresentados. Uma observação interessante é que a taxa de falsos positivos é muito maior no *cr.ger* e que os algoritmos imunológicos tendem a ter um taxa maior que os outros algoritmos, mas uma taxa de falsos negativos similar.

No gráfico de tempo de treinamento (7.9), pode-se ver que existe uma diferença muito grande entre os algoritmos (note a escala logarítmica). Em um extremo, as redes bayesianas e o algoritmo ID3 têm tempo de treinamento ínfimo, inferior a um segundo. Em outro, os algoritmos imunológicos levaram de 15 a 330 segundos. Embora esses números não sejam considerados altos em relação a esse tipo de atividade computacional, são várias ordens de magnitude maiores que os algoritmos mais rápidos. Apesar disso, o tempo de execução geralmente não é considerado um fator decisivo na comparação dos algoritmos se mantiver-se em uma faixa aceitável para a execução.

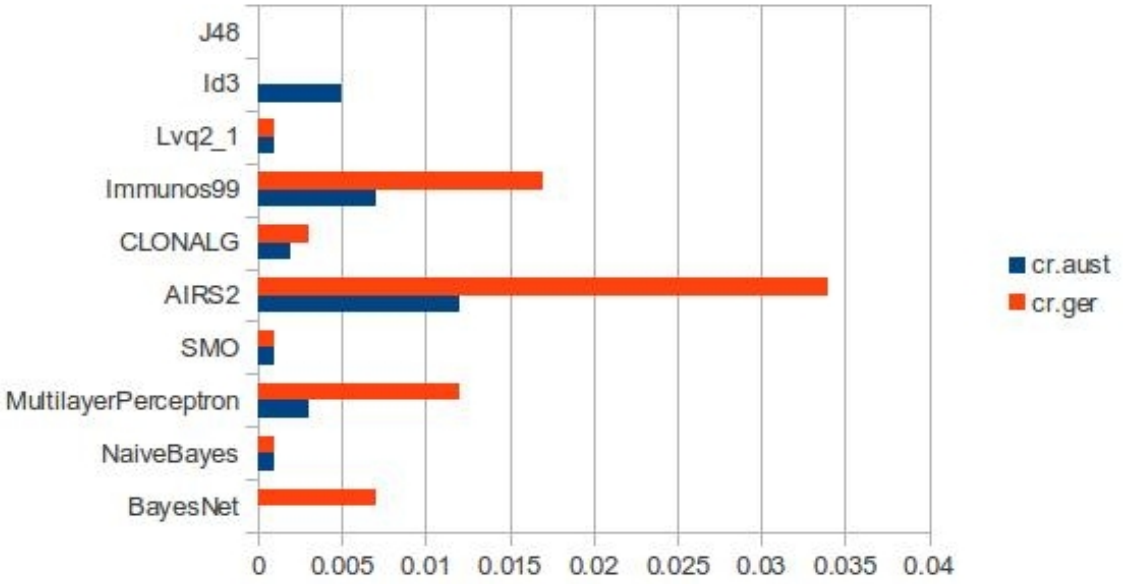
Figura 7.9: Tempo de treinamento em segundos



É interessante notar que o tempo de treinamento dos algoritmos para o *cr.aust* é aproximadamente a metade em relação ao *cr.ger*. Levando em consideração que o número de instâncias também é aproximadamente a metade, pode-se ver que o número de instâncias não é uma fator significativo no tempo de treinamento. O mesmo pode ser observado para o tempo de teste (7.10), mas esses valores são tão baixos, tanto isolados quanto em relação ao tempo de treinamento, que são praticamente irrelevantes.

Novamente, a análise dos tempos de treinamento e testes dos algoritmos imunológicos deve levar em consideração que o critério de parada desses algoritmos não é um número de iterações fixo. O processo de treinamento só é finalizado quando um

Figura 7.10: Tempo de teste em segundos

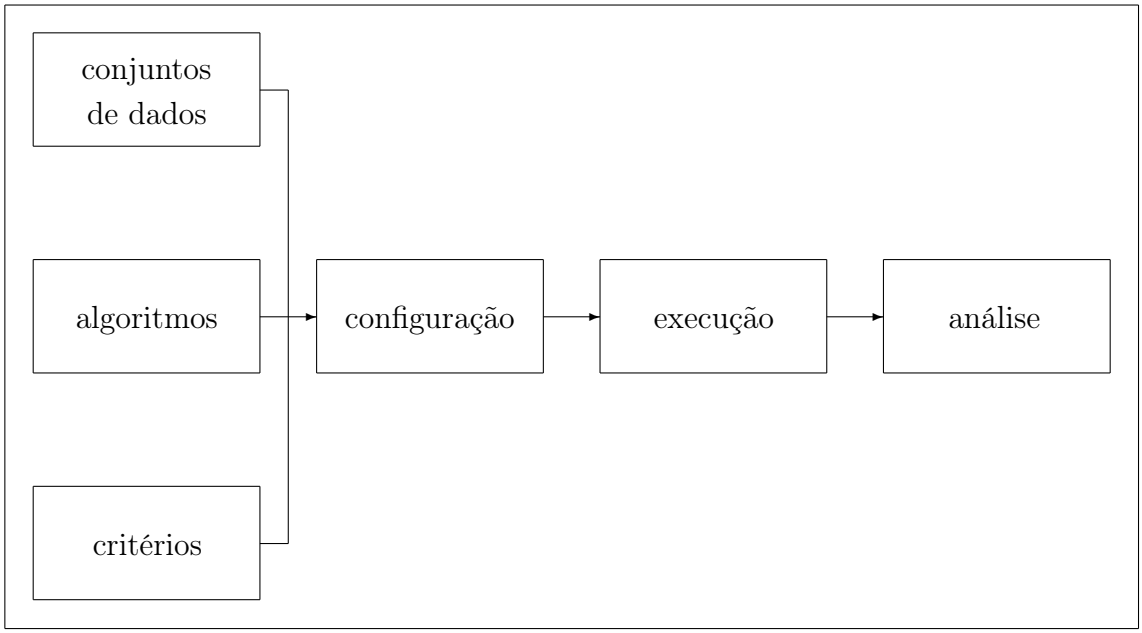


limiar pré-definido de similaridade é atingido. Esse limiar é um parâmetro para a execução do algoritmo e pode ser configurado (assim como outros parâmetros) caso o tempo de execução seja considerado um fator mais importante que os outros.

8 CONCLUSÃO

Esse trabalho buscou comparar a aplicação de algoritmos tradicionais com algoritmos imunológicos para o problema da detecção de fraude. Para atingir esse objetivo seguiu-se o método ilustrado na figura 8.1.

Figura 8.1: Etapas do trabalho



No início do trabalho, após a definição da detecção de fraude como escopo de pesquisa, foram selecionados os conjuntos de dados e algoritmos que foram testados. Para os conjuntos de dados, foram consultadas fontes tradicionais na área de mineração de dados e inteligência artificial. Para os algoritmos, foram consultados trabalhos recentes na área da detecção de fraude e foram escolhidos os principais representantes dos tipos mais usados de algoritmos.

Após as definições iniciais, foi escolhida a arquitetura para os testes do experimento, utilizando a plataforma WEKA, padrão para grande parte dos experimentos

em mineração de dados. Essa escolha facilitou a implementação do experimento em diversos aspectos. Nenhum dos algoritmos teve que ser implementado, já que todos os algoritmos tradicionais são implementados pela plataforma. Os algoritmos imunológicos não são implementados na versão padrão do WEKA, mas o pacote de algoritmos de Jason Brownlee foi utilizado, o que permitiu utilizar esses algoritmos como qualquer outro algoritmo implementado no WEKA.

Finalizando a fase de planejamento, foram definidos os critérios utilizados para a comparação dos resultados. Essa definição teve base nos padrões da mineração de dados e, mais especificamente, na detecção de fraude. Essa escolha não alterou a definição da plataforma, já que os critérios selecionados são gravados pelo WEKA por padrão em cada teste.

Na segunda fase do trabalho, foram criados os arquivos de configuração que descrevem o experimento para cada algoritmo no WEKA. A criação dos arquivos foi simples, mas grande parte do trabalho nessa etapa consistiu em identificar os parâmetros a serem testados para cada algoritmo. Isso foi feito para que os resultados representassem a melhor configuração de cada algoritmo, sem que fossem prejudicados por uma escolha ruim de parâmetros.

A etapa de execução foi separada no diagrama, mas na prática ocorreu junto com a configuração dos experimentos, já que foi necessária uma familiarização com a ferramenta WEKA e com a configuração dos algoritmos e experimentos. Nessa etapa, todos os testes foram executados na máquina de testes e os arquivos com os resultados foram criados e combinados nos arquivos de resultado finais.

A última etapa do trabalho foi a análise dos resultados compilados. Nessa etapa, os resultados foram analisados, seguindo os critérios definidos nas etapas anteriores. A partir dos arquivos de resultados, foram geradas tabelas e gráficos para a análise dos critérios de avaliação. A partir desses, foram identificados os melhores algoritmos para cada critério, além de observações relevantes sobre os resultados.

8.1 Contribuições

Esse trabalho envolveu diversas áreas da ciência da computação e mineração de dados. As contribuições foram, também, variadas:

a) **Detecção de fraude**: o último trabalho a fazer uma comparação de vários métodos no contexto da detecção de fraude foi PHUA et al. (2010). Esse trabalho busca servir como base para trabalhos futuros que busquem expandir a pesquisa nessa área.

b) **Mineração de dados**: com o constante avanço dos algoritmos e técnicas para a mineração de dados, é necessária constante avaliação da eficiência desses métodos,

principalmente para comparação com os métodos já existentes.

c) **Sistemas Imunológicos Artificiais**: uma das principais contribuições desse trabalho foi possibilitar a comparação dos algoritmos imunológicos com algoritmos tradicionais da inteligência artificial. Os Sistemas Imunológicos Artificiais são uma área relativamente recente de desenvolvimento e não ainda não existem muitos trabalhos comparando esses algoritmos às técnicas tradicionais.

d) **WEKA**: apesar de ser uma ferramenta bastante usada nas áreas de mineração de dados e inteligência artificial, esse trabalho utilizou uma versão do WEKA que inclui algoritmos que não estão presentes na versão padrão. Com o estabelecimento dos algoritmos imunológicos como uma área da inteligência artificial, é possível que esses algoritmos sejam incorporados como mais uma categoria de algoritmos na versão padrão.

e) **Computação**: a mineração de dados é uma área extensa e existem muitas aplicações na computação em geral. Os sistemas imunológicos foram e ainda podem ser usados em vários problemas da computação, como análise de tráfego de rede e otimização de código. Qualquer trabalho que pretenda avaliar a aplicação de novos algoritmos deve utilizar um processo semelhante ao utilizado nesse trabalho para a validação dos resultados.

8.2 Continuidade

Apesar da extensão desse trabalho, diversos aspectos podem ser melhorados no método utilizado. Os principais são:

a) O número de conjuntos de dados testados foi satisfatório, mas um número maior poderia ser utilizado para garantir a veracidade dos resultados. Apesar disso, não existem muitos conjuntos de dados de detecção de fraude disponíveis livremente.

b) O número e a faixa de parâmetros utilizada para os algoritmos foi definido levando em consideração o tempo de execução do trabalho. Mais e faixas maiores de parâmetros poderiam ser testados em um período maior de tempo.

c) O estudo dos algoritmos testados teve escopo limitado pelo tempo, já que um número relativamente grande de algoritmos foi testado. Uma análise mais refinada de cada algoritmo geraria resultados mais próxima da aplicação real desses algoritmos.

Além das melhorias para esse trabalho, foram encontrados diversos problemas durante a execução das etapas desse trabalho nas ferramentas utilizadas que poderiam ser melhorados:

a) Os algoritmos do WEKA fazem parte da versão padrão há anos e foram testados em diversos trabalhos. Por outro lado, os algoritmos no pacote de algoritmos imunológicos foram desenvolvidos por apenas uma pessoa e foram usados em poucos trabalhos. Provavelmente podem ser feitas melhorias e otimizações no código desses algoritmos.

b) O WEKA tem várias funcionalidades que facilitam a execução de experimentos, mas não existem muitas facilidades para a análise e comparação dos resultados. As ferramentas que existem não cobriam as necessidades desse trabalho, e a sua adição facilitaria trabalhos semelhantes. O fato do WEKA ser uma plataforma de código aberto facilita a adição de novas funcionalidades.

REFERÊNCIAS

ACFE. **Report to the Nations on Occupational Fraud and Abuse**. Austin, Texas, Estados Unidos: Association of Certified Fraud Examiners, 2012. 76p. Disponível em: <http://www.acfe.com/rttn.aspx>. Acessado em 20 jun. 2012.

AICKELIN, U., CAYZER, S. The Danger Theory and Its Application to Artificial Immune Systems. **Computing Research Repository**, Nova Iorque, Estado Unidos, 2002. Disponível em <http://arxiv.org/abs/0801.3549>. Acessado em 24 jun. 2012.

AICKELIN, U., DASGUPTA, D. **Artificial Immune Systems**. Nova Iorque, Estado Unidos: Springer, 2005. 620p. ISBN 9780387234601.

AICKELIN, U., DASGUPTA, D. Advances in artificial immune systems. **IEEE Computational Intelligence Magazine**, Nova Iorque, Estado Unidos, 2006. Disponível em <ieeexplore.ieee.org/iel5/10207/4012713/04013594.pdf>. Acessado em 22 nov. 2012.

ANDERSON, J. P. **Computer Security Technology Planning Study**. Massachusetts: Hanscom AFB, 1972. Disponível em: <http://nob.cs.ucdavis.edu/history>. Acessado em 14 jun. 2012.

ARAL, K. D. et al. A Prescription Fraud Detection Model. **Computer Methods and Programs in Biomedicine**, Amsterdan, 2011. Disponível em <http://dl.acm.org/citation.cfm?id=2151621>. Acessado em 16 out. 2012.

BACE, R. G. **Intrusion Detection**. Indiana: Sams Publishing, 2000. 339p. (MacMillan Technology Series). ISBN 9781578701858.

BARR, R. S. et al. Designing and reporting on computational experiments with heuristic methods. **Journal of Heuristics**, Nova Iorque, Estados Unidos, v.1, p.9–32, 1995. Disponível em <http://link.springer.com/article/10.1007out. 2012.

BAYES, T. An Essay towards Solving a Problem in the Doctrine of Chances. **Philosophical Transactions**, Londres, Reino Unido, v.53, p.370–418, 1763. Disponível em <http://rstl.royalsocietypublishing.org/content/53/370>. Acessado em 1 set. 2012.

BEWICK, V., CHEEK, L., BALL, J. Statistics review 13: receiver operating characteristic curves. **Crit Care**, Bethesda, Estados Unidos, v.8, n.6, p.508–512, 2007. Disponível em <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1065080/>. Acessado em 15 out. 2012.

BRONWLEE, J. **WEKA Classification Algorithms**. 2011. Disponível em <http://wekaclassalgos.sourceforge.net/>. Acessado em 29 nov. 2012.

BROWNLEE, J. Artificial Immune Recognition System (AIRS): a review and analysis. **Technical report**, Melbourne, Austrália, 2005. Disponível em <http://www.ict.swin.edu.au/personal/jbrownlee>. Acessado em 14 nov. 2013.

BROWNLEE, J. The 'shape-space' and 'affinity landscape' Immunological Paradigms. **Technical Report 070310A**, Melbourne, Australia, 2007. Disponível em <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.73.128>. Acessado em 15 out. 2012.

BROWNLEE, J. **Clever Algorithms**: nature-inspired programming recipes. Raleigh, Estados Unidos: Lulu, 2011. 436p. ISBN 9781446785065.

BURNET, F. M. **The Clonal Selection Theory of Acquired Immunity**. Nashville, Tenesse: Vanderbilt University Press, 1959. 236p. Disponível em <http://archive.org/details/clonalselectiont00burn>. Acessado em 23 jun. 2012.

CARNEIRO, J., STEWART, J. Rethinking "shape space": evidence from simulated docking suggests that steric shape complementarity is not limiting for antibody-antigen recognition and idiotypic interactions. **Journal of theoretical biology**, Paris, 1994. Disponível em <http://www.cs.unm.edu/ forrest/classes/immunoclass/readings/>. Acessado em 08 out. 2012.

CARTER, J. H. The Immune System as a Model for Pattern Recognition and Classification. **Journal of the American Medical Informatics Association**, Bethesda, Estados Unidos, 2000. Disponível em <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC61453/>. Acessado em 27 nov. 2012.

CASTRO, L. N. de, ZUBEN, F. J. V. The Clonal Selection Algorithm with Engineering Applications. **Genetic and Evolutionary Computation Conference**, Nova Iorque, Estados Unidos, 2002. Disponível

em <http://www.dca.fee.unicamp.br/ vonzuben/research/lnunes_dout/artigos/>. Acessado em 29 out. 2012.

CASTRO, L. N. de, ZUBEN, F. J. von. Learning and Optimization Using the Clonal Selection Principle. **IEEE Transactions on Evolutionary Computation**, Nova Iorque, Estado Unidos, 2002. Disponível em <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1011539>. Acessado em 02 jul. 2012.

CAYZER, S., SULLIVAN, J. Modelling danger and anergy in artificial immune systems. In: GENETIC AND EVOLUTIONARY COMPUTATION, 9., Nova Iorque, Estados Unidos. **Proceedings...** ACM, 2007. Disponível em <http://dl.acm.org/citation.cfm?doid=1276958.1276963>. Acessado em 22 nov. 2012.

CORTES, C., VAPNIK, V. Support-Vector Networks. **Mach. Learn.**, Massachusetts, Estados Unidos, v.20, n.3, p.273–297, 1995. Disponível em <http://dl.acm.org/citation.cfm?id=218929>. Acessado em 1 set. 2013.

DASGUPTA, D., YU, S., NINO, F. Recent Advances in Artificial Immune Systems: models and applications. **Applied Soft Computing**, Amsterdan, 2010. Disponível em <http://www.sciencedirect.com/science/article/pii/S1568494610002723>. Acessado em 26 ago. 2012.

FARMER, J. D., PACKARD, N. H., PERELSON, A. S. The Immune System, Adaptation, and Machine Learning. **Physica D: Nonlinear Phenomena**, Los Alamos, Estados Unidos, 1986. Disponível em <http://www.sciencedirect.com/science/article/pii/016727898690240X>. Acessado em 29 out. 2012.

FBI. **Insurance Fraud**. 2010. Disponível em <http://www.fbi.gov/stats-services/publications>. Acessado em 20 jun. 2012.

FORREST, S., BEAUCHEMIN, C. Computer immunology. **Communications of the ACM**, Nova Iorque, Estados Unidos, 1997. Disponível em <http://www.cs.unm.edu/ forrest/publications/>. Acessado em 29 out. 2012.

FORREST, S. et al. Self-nonself Discrimination in a Computer. In: IN PROCEEDINGS OF THE 1994 IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY, Nova Iorque, Estados Unidos. **Anais...** IEEE Computer Society Press, 1994. p.202–212. Disponível em <http://www.cs.unm.edu/ immsec/publications/>. Acessado em 29 out. 2012.

GARRETT, S. M. How Do We Evaluate Artificial Immune Systems? **Evolutionary Computation**, Cambridge, Estados Unidos, 2005. Disponível em <http://www.mitpressjournals.org/doi/abs/10.1162/1063656054088512>. Acessado em 02 jul. 2012.

GREENSMITH, J., AICKELIN, U., CAYZER, S. Introducing Dendritic Cells as a Novel Immune-Inspired Algorithm for Anomaly Detection. In: INTERNATIONAL CONFERENCE ON ARTIFICIAL IMMUNE SYSTEMS, Nova Iorque, Estado Unidos. **Anais...** Springer, 2005. p.153–167. Disponível em <http://www.springerlink.com/content/equhb06n42emjb0w/>. Acessado em 02 jul. 2012.

GREENSMITH, J., AICKELIN, U., TWYCROSS, J. Articulation and Clarification of the Dendritic Cell Algorithm. **Lecture Notes in Computer Science**, Nova Iorque, Estado Unidos, p.404–417, 2006. Disponível em <http://link.springer.com/chapter/10.1007/11823940_31>. Acessado em 29 out. 2012.

HALL, M. et al. The WEKA Data Mining Software: an update. **Special Interest Group on Knowledge Discovery and Data Mining Explorations**, Nova Iorque, Estados Unidos, 2009. Disponível em <http://www.kdd.org/explorations/issues/11-1-2009-07/>. Acessado em 25 nov. 2012.

HAND, D. J., MANILLA, H., SMYTH, P. **Principles of Data Mining**. Massachusetts: MIT Press, 2001. 542p. (Adaptive Computation and Machine Learning). ISBN 9780262082907.

HUANG, R., TAWFIK, H., NAGAR, A. K. A Novel Hybrid Artificial Immune Inspired Approach for Online Break-in Fraud Detection. **Procedia Computer Science**, Nova Iorque, Estado Unidos, 2010. Disponível em <http://www.sciencedirect.com/science/article/pii/S187705091000308X>. Acessado em 30 out. 2012.

ISHIDA, Y. Fully distributed diagnosis by PDP learning algorithm: towards immune network pdp model. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, Nova Iorque, Estados Unidos. **Proceedings...** IEEE, 1990. p.777–782. Disponível em <http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5726623>. Acessado em 30 out. 2012.

JANEWAY, C. A. Approaching the asymptote? Evolution and revolution in immunology. **Cold Spring Harbor symposia on quantitative biology**, Connecticut,

1989. Disponível em <http://symposium.cshlp.org/content/54>. Acessado em 23 jun. 2012.

JERNE, N. K. Towards a network theory of the immune system. **Annual Immunology**, [S.l.], p.373–389, 1974. Disponível em <http://www.ncbi.nlm.nih.gov/pubmed/4142565>. Acessado em 18 abr. 2013.

KOHONEN, T. **Self-organizing maps**. Secaucus, Nova Jérsia, Estados Unidos: Springer-Verlag New York, Inc., 1997. Disponível em <http://cis.legacy.ics.tkk.fi/research/reports/biennial02-03/>. Acessado em 2 set. 2013.

LEVI, M., BURROWS, J. Measuring the Impact of Fraud in the UK. **British Journal of Criminology**, Oxford, Inglaterra, 2002. Disponível em <http://bjc.oxfordjournals.org/content/48/3/293>. Acessado em 30 out. 2012.

LUGER, G. F. **Artificial intelligence**: structures and strategies for complex problem solving. California: Pearson Addison-Wesley, 2009. 754p. 9780321545893.

MATZINGER, P. **Tolerance, Danger and the Extended Family**. Palo Alto, California: Annual Reviews, 1994. 991-1045p. v.12. Disponível em <http://www.annualreviews.org/doi/abs/10.1146/annurev.iy.12.040194.005015>. Acessado em 24 jun. 2012.

MICHIE, D., SPIEGELHALTER, D. J., TAYLOR, C. C. (Ed.). **Machine Learning, Neural and Statistical Classification**. Nova Iorque, Estados Unidos: Ellis Horwood, 1994. Disponível em <http://www1.maths.leeds.ac.uk/ charles/statlog/>. Acessado em 24 jun. 2012.

PEARL, J. **Probabilistic Reasoning in Intelligent Systems**: networks of plausible inference. Austin, Texas, Estados Unidos: Morgan Kaufmann, 1988. Disponível em: <http://dl.acm.org/citation.cfm?id=52121>. Acessado em 1 set. 2013.

PERELSON, A. S., OSTER, G. F. Theoretical studies of clonal selection. **Journal of Theoretical Biology**, Nova Iorque, Estados Unidos, 1979. Disponível em <http://www.sciencedirect.com/science/article/pii/0022519379902753>. Acessado em 02 jul. 2012.

PHUA, C. et al. A Comprehensive Survey of Data Mining-based Fraud Detection Research. **Computing Research Repository**, Nova Iorque, Estado Unidos, 2010. Disponível em <http://arxiv.org/abs/1009.6119>. Acessado em 10 abr. 2012.

PLATT, J. Fast Training of Support Vector Machines using Sequential Minimal Optimization. In: SCHOELKOPF, B., BURGES, C., SMOLA,

A. (Ed.). **Advances in Kernel Methods - Support Vector Learning**. Massachussets, Estados Unidos: MIT Press, 1998. Disponível em <http://research.microsoft.com/apps/pubs/?id=68391>. Acessado em 2 set. 2013.

QUINLAN, J. R. Induction of Decision Trees. **Machine Learning**, Alphen aan den Rijn, Holanda, v.1, n.1, p.81–106, 1986. Disponível em <http://www.dmi.unict.it/ apulvirenti/agd/>. Acessado em 28 de ago. 2013.

QUINLAN, J. R. **C4.5**: programs for machine learning. San Mateo, Califórnia, Estados Unidos: Morgan Kaufman, 1993. Disponível em <http://dl.acm.org/citation.cfm?id=152181>. Acessado em 28 de ago. 2013.

SILVA, O. J. de Placido e. **Vocabulário Jurídico**. Rio de Janeiro: Forense, 1982. 526p. v.2.

STEINMAN, R. M., COHN, Z. A. Identification of a Novel Cell Type in Peripheral Lymphoid Organs Mice. **The Journal of Experimental Medicine**, Nova Iorque, Estado Unidos, 2002. Disponível em <http://jem.rupress.org/content/149/1>. Acessado em 02 jul. 2012.

TIMMIS, J., NEAL, M., HUNT, J. An artificial immune system for data analysis. **Biosystems**, Amsterdan, v.55, n.1–3, p.143–150, 2000. Disponível em <http://www.sciencedirect.com/science/article/pii/S0303264799000921>. Acessado em 08 out. 2012.

WATKINS, A., TIMMIS, J., BOGGESS, L. Artificial Immune Recognition System ("AIRS"): an "immune-inspired" supervised learning algorithm. **Genetic Programming and Evolvable Machines**, Massachusetts, Estados Unidos, v.5, n.3, p.291–317, 2004. Disponível em <http://www.cse.msstate.edu/ andrew/research/publications.html>. Acessado em 26 nov. 2012.

WERBOS, P. **Beyond Regression**: new tools for prediction and analysis in the behavioral sciences. 1974. Tese (Doutorado) — Harvard University, Cambridge, Massachussets, Estados Unidos.