ROBERTO TEIXEIRA ALVES

UM SISTEMA IMUNOLÓGICO ARTIFICIAL PARA CLASSIFICAÇÃO HIERÁRQUICA E *MULTI-LABEL* DE FUNÇÕES DE PROTEÍNAS

Tese apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial Universidade Tecnológica Federal do Paraná, na área de concentração de Informática Industrial, como requisito parcial para à obtenção do título de "Doutor em Ciências" – área de concentração Informática Industrial.

Orientadora: Profª. Drª. Myriam R. Delgado
Co-orientador: Prof. Dr. Alex A. Freitas

Curitiba
2010

Dados Internacionais de Catalogação na Publicação

A474 Alves, Roberto Teixeira
Um sistema imunológico artificial para classificação hierárquica e multi-label de funções de proteínas / Roberto Teixeira Alves. — 2010.
219 p. : il. ; 30 cm

Orientador : Myriam Regattieri de Biase da Silva Delgado
Co-orientador : Alex Alves Freitas
Tese (Doutorado) – Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial, Curitiba, 2010
Bibliografia : p. 157-166 e índice

1. Bioinformática. 2. Sistema imunológico – Simulação por computador. 3. Mineração de dados (Computação). 4. Engenharia elétrica – Teses. I. Delgado, Myriam Regattieri de Biase da Silva, orient. II. Freitas, Alex Alves, co-orient. III. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Elétrica e Informática Industrial. III. Título.

CDD (22. ed.) 621.3

Biblioteca Central da UTFPR, Campus Curitiba

UTFPR
CAMPUS CURITIBA

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Campus Curitiba

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA E INFORMÁTICA INDUSTRIAL
cpgei
21 anos

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

Título da Tese Nº 52:

"Um Sistema Imunológico Artificial para Classificação Hierárquica e Multi-label de Funções de Proteínas"

por

Roberto Teixeira Alves

Esta tese foi apresentada, às 14n do dia 26 de fevereiro de 2010, como requisito parcial para a obtenção do título de DOUTOR EM CIÊNCIAS – Área de Concentração: Informática Industrial, pelo Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial – CPGEI – da Universidade Tecnológica Federal do Paraná – UTFPR. O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

Profª. Dr. Myriam R. de Biase da S. Delgado
(Orientadora – UTFPR)

Profª. Dr. Deborah Ribeiro Carvalho
(UTP)

Prof. Dr. Julio Cesar Nievola
(PUC-PR)

Profª. Dr. Elaine Machado Benelli
(UFPR)

Profª. Dr. Denise Fukumi Tsunoda
(UFPR)

Profª. Dr. Celso Antônio Alves Kaestner
(Suplente - UTFPR)

Visto da coordenação:

Prof. Dr. Humberto Remígio Gamba
(Coordenador do CPGEI)

Dedico à amada minha esposa
Lauren, com muito amor.

Agradecimentos

Agradeço a Deus pela saúde dada nesse período e por ter conspirado a meu favor para conclusão desta tese.

À minha esposa Lauren pelo carinho, dedicação e compreensão na longa jornada percorrida. Te amo do tamanho do universo.

Aos meus pais, Joel e Irair, pelos valiosos ensinamentos sendo fontes de inspiração pessoal e profissional.

À minha orientadora e madrinha professora Myriam R. Delgado pelos preciosos apontamentos, sobretudo pela amizade, admiração e carinho que permanecerá eternamente.

Ao professor e *lord* Alex A. Freitas pela fundamental e efetiva participação nas discussões e sugestões durante todo o período de pesquisa, excercendo a importante função de co-orientador.

À professora e madrinha Deborah R. Carvalho queno passado, ainda durante a graduação, apostou no meu potencial e me incentivou a ingressar na vida acadêmica.

Aos muitos amigos pelas discussões e momentos de descontração.

À banca pelas contribuições.

À CAPES pela bolsa.

*"Se você rouba idéias de um autor, é plágio. Se você rouba de muitos autores, é pesquisa." (**Wilson Mizner**)*

*"A grandeza não consiste em receber honras, mas em merecê-las." (**Aristóteles**)*

*"Deve-se aprender sempre, até mesmo com um inimigo." (**Isaac Newton**)*

*"Se, a princípio, a ideia não é absurda, então não há esperança para ela." (**Albert Einstein**)*

Resumo

ALVES, R. T. Um sistema imunológico artificial para classificação hierárquica e *multi-label* de funções de proteínas. 2010. 219 p. Tese de Doutorado – Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI), Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, 2010.

Esta tese propõe um novo algoritmo baseado em Sistemas Imunológicos Artificiais (SIA) para classificação hierárquica e *multi-label*, onde os classificadores gerados são representados na forma de regras SE-ENTÃO. A classificação hierárquica e *multi-label* é considerada desafiadora uma vez que um exemplo está associado a uma ou mais classes organizadas hierarquicamente, sendo que esta organização estrutural de classes deve ser considerada na construção dos classificadores. A técnica proposta aborda a construção de classificadores hierárquicos locais (onde cada classificador processa apenas exemplos de classes em uma região local da hierarquia de classes) e globais (onde um único classificador processa exemplos de todas as classes ao mesmo tempo). A área de aplicação utilizada para validação desta tese é a predição de função biológica de proteínas usando termos da ontologia gênica como classes a serem preditas pelo SIA, onde as classes são hierarquicamente organizadas em um grafo acíclico direcionado. O desempenho do algoritmo é avaliado experimentalmente para 10 bases de proteínas. Os critérios de avaliação do algoritmo nos experimentos computacionais serão a correção preditiva (taxa de acerto e área da curva *precision-recall*) e a simplicidade do conhecimento descoberto (medida pelo número de regras e número total de condições nas regras descobertas). Os experimentos computacionais permitem identificar parâmetros e procedimentos que influenciam no desempenho da técnica proposta. Os testes comparativos com os algoritmos PART e Clus mostram que sobre alguns conjuntos de experimentos a abordagem proposta se mostrou superior, enquanto em outros conjuntos não foi possível superar as técnicas da literatura usada para comparação.

Palavras-chave: Ontologia Gênica, Bioinformática, Classificação Hierárquica e *Multi-Label*, Sistemas Imunológicos Artificiais, Mineração de Dados.

Abstract

ALVES, R. T. An Artificial Immune System for hierarchical and multi-label classification of protein functions. 2010. 219 p. Tese de Doutorado – Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial (CPGEI), Universidade Tecnológica Federal do Paraná (UTFPR), Curitiba, 2010.

This thesis proposes a new approach based on Artificial Immune System (AIS) for hierarchical multi-label classification, where the classifiers produced by the system are represented in the form of IF-THEN classification rules. Hierarchical multi-label classification is a challenging problem, because an example is associated with one or more classes organised into a hierarchy and the class hierarchy must be considered in the construction of the classifiers. The proposed method addresses the construction of local hierarchical classifiers (where each classifier processes only examples of classes in a local region of the hierarchy) and global hierarchical classifiers (where a single classifier processes examples of all classes at the same time). The application domain used to validate the proposed methods was the prediction of the biological function of proteins, using terms of the Gene Ontology as classes to be predicted by the AIS, where the classes are hierarchically organised into a directed acyclic graph . The performance of the algorithm is evaluated in computational experiments with 10 datasets of proteins. The evaluation criteria in these experiments will be the predictive accuracy (accuracy rate and the area under the precision-recall curve) and the simplicity of the discovered knowledge (measured by the number of rules and total number of conditions in the discovered rules). The computational experiments allowed the identification of parameter settings and procedures that significantly influence the performance of the proposed method. The experiments comparing the proposed method with algorithms PART and Clus have shown that in some datasets the proposed method outperformed other methods, whilst in other datasets it was not possible to outperform other methods proposed in the literature.

Keywords: Gene Ontology, Bioinformatics, Multi-label Hierarchical Classification, Artificial Immune Systems, Data Mining.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

AD	Árvore de Decisão
aiNet	*Artificial Immune Network*
AIRS	*Artificial Immune Recognition System*
APC	*Antigen-Presenting Cell*
AUPRC	*Area Under the Precision-Recall Curve*
CC	Componente Celular
CD	*Critical Distance*
Copt-aiNet	*Artificial Immune Network for Combinatorial Optimization*
DAG	*Directed Acyclic Graph*
EBI	*European Bioinformatics Institute*
EC	*Enzyme Commission*
EMBL	*European Molecular Biology Laboratory*
FM	Função Molecular
GO	*Gene Ontology*
HMM	*Hidden Markov Model*
IA	Inteligência Artificial
InterPro	*Integrated Resource of Protein Families, Domains and Sites*
JIPID	*Japan International Protein Information Database*
KDD	*Knowledge Discovery in Databases*
MHCAIS	*Multi-Label Hierarchical Classification with an Artificial Immune System*
MIPS	*Munich Information Center for Protein Sequences*
MLP	*Multi-Layer Perceptron*
NB	*Näive Bayes*
NBRF	*National Biomedical Research Foundation*

PB	Processo Biológico
PIR	*Protein Information Resource*
PIR-NREF	PIR-*Non-redundant Reference*
PIR-PSD	PIR-*Protein Sequence Database*
PST	*Probabilistic Suffix Trees*
RB	Rede Bayesiana
RBF	*Radial-Basis Function*
RNA	Rede Neural Artificial
ROC	*receiver operating characteristic*
SIA	Sistemas Imunológicos Artificiais
SIB	*Swiss Institute of Bioinformatics*
SVM	*Support Vector Machines*
TrEMBL	*Translated* EMBL
UniParc	UniProt *archive*
UniProt	*Universal Protein Resource*
UniProtKB	UniProt *Knowledge Base*
UniRef	UniProt *Reference Clusters*

LISTA SÍMBOLOS

β_{FT} peso entre *precision* e *recall* na função *fitness*

δ_{AF} limiar de afinidade (*matching*)

δ_{CL} limiar de classificação

δ_{FT} limiar de *fitness* – versão global

δ_{SIM} limiar de similariada entre ab_j e ab'_j

AB população de Anticorpos

ab_j j-ésimo anticorpo (regra) da população

AG população de antígenos

AG' população de antígenos considerando a estrutura hierárquica $\mathscr{H}$

ag'_i i-ésimo antígeno considerando a estrutura hierárquica $\mathscr{H}$

ag_i i-ésimo antígeno da população

B_d^j marcador ativo/inativo da d-ésima condição de ab_j

CL população de clones

cl clone de ab

CR conjunto de regras descobertas

$\mathscr{D}$ atributos previsores do domínio da aplicação

FN Falso Negativo

FP Falso Positivo

$HiperClTx$ parâmetro que estimula a hiperprodução de clones

$\mathbf{l}'_i$ vetor binário de classes associadas a ag'_i considerando $\mathscr{H}$

l'_i classe associada a ab'_j considerando $\mathscr{H}$ – versão local

l_{ik} k-ésima classe associada a ag_i

L_i conjuntos de classes anotadas para ag_i

l_q classe do domínio da aplicação

m número de classes representadas em ab_j

$MelhorRegra$ melhor regra descoberta na iteração t

$MutMax$ taxa máxima de mutação

$MutMin$ taxa mínima de mutação

$\#MaxCl$ quantidade máxima de clones de ab_j

$\#MaxIter$ número máximo de iterações do período evolutivo

$\#MaxExemBD$ número de exemplos não cobertos até a iteração t

OP_d^j operador lógico da d-ésima condição de ab_j

P *precision*

R *recall*

r_j j-ésima regra descoberta

t_i i-ésimo exemplo da base de teste

T base de teste

T conjunto das distintas classes que ocorrem em AG

t iteração corrente do procedimento de extração de regras

t' iteração corrente de um período evolutivo

T_{A_i} conjunto de classes ancestrais de L_i

T_A conjunto de classes ancestrais de T

T_{H_i} conjunto de classes associadas a ab_i' considerando $\mathscr{H}$

T_H $T \bigcup T_A$

$TPMax$ número máximo de anticorpos na população

V_d^j valor válido da d-ésima condição de ab_j

VN Verdadeiro Negativo

VP Verdadeiro Positivo

$\mathbf{x}_i$ valores dos atributos previsores de ag_i

x_{id} valor do d-ésimo atributo previsor de ag_i

$y_{jq'}$ classes ancestral de y_{jq}

y_{jq} q-ésima classe predita pela regra ab_j

Y_j classes preditas (consequente) pela regra ab_j – versão global

y_j classe predita (consequente) pela regra ab_j – versão local

y_j^l classe prevista por ab_j^l – versão local

$\mathbf{z}_j$ antecedente da regra ab_j

z_{jd} d-ésima condição codificada em ab_j

SUMÁRIO

1 INTRODUÇÃO

Desde o início do projeto de sequenciamento do genoma na década de 90, os bancos de dados biológicos têm sido sobrecarregados com dados experimentais. A quantidade de dados armazenados nestas bases tem aumentado exponencialmente ao longo do tempo, inviabilizando a exploração e análise manual destes dados na busca de informações biológicas úteis.

A bioinformática surgiu como uma alternativa viável para atender a demanda por busca de conhecimento biológico contido nestas bases. Geralmente, define-se como bioinformática uma área multidisciplinar que se baseia no conhecimento especialista de matemáticos, cientistas da computação, biólogos, físicos, entre outros (FOGEL; CORNE, 2003). A bioinformática tem como objetivo o desenvolvimento de técnicas e métodos computacionais para a exploração, análise e extração automática de informações biológicas úteis armazenadas nestas bases de dados (HUGHEY; KARPLUS, 2001). Vários são os problemas que têm sido tratados pela bioinformática, entre estes se destaca a predição de funções biológicas de proteínas, que é abordado nesta tese.

A predição de funções de proteínas consiste em associar função (funções) biológica(s) para proteínas cujas funções não são ainda conhecidas. Diversas são as razões que justificam o interesse de pesquisadores em determinar estas funções, tais como: um melhor entendimento de doenças, desenvolvimento de medicamentos mais efetivos, medicina preventiva, entre outras. Este procedimento pode ser realizado através de experimentos em laboratórios biológicos ou métodos computacionais, como por exemplo, aqueles baseados em técnicas de mineração de dados.

A mineração de dados trata-se de um campo multidisciplinar de pesquisa, envolvendo áreas como aprendizado de máquina, estatística, bases de dados, sistemas especialistas e visualização de dados. Este campo utiliza métodos, algoritmos e técnicas oriundas de diversas áreas, com o objetivo principal de extrair conhecimento a partir de bases de dados do mundo real (FAYYAD et al., 1996). Dentre as principais tarefas de mineração de dados, destaca-se a classificação, a qual é o foco deste trabalho.

Em geral, a predição de funções de proteínas é tratada como um problema de classificação. Em classificação, o objetivo é prever ou associar uma ou mais classes a um exemplo (registro) da base de dados, dentre um conjunto pré-definido de classes (FREITAS, 2002). No contexto de classificação, uma proteína é considerada um exemplo de uma base de dados (biológica), e as funções biológicas são tratadas como classes a serem previstas.

A principal técnica para prever funções de proteínas consiste na busca de similaridade entre sequências de proteínas (ALTSCHUL et al., 1996). Este procedimento é dividido em duas fases. A primeira fase consiste em buscar em um conjunto de dados, proteínas cuja sequência de aminoácidos é mais semelhante possível à sequência da proteína alvo. Na segunda fase, a abordagem assume que a(s) função(ões) biológica(s) da proteína alvo é(são) a(s) mesma(s) que a(s) da(s) proteína(s) semelhante(s). Entretanto esta metodologia possui algumas limitações, tais como (PAPPA; BAINES; FREITAS, 2005):

- proteínas podem possuir sequências parecidas e desempenhar funções diferentes;
- proteínas podem possuir sequências diferentes e desempenhar as mesmas funções biológicas;
- as proteínas comparadas podem possuir alto nível de similaridade em regiões das sequências que não são determinantes para a função biológica;
- por se basear apenas na similaridade entre sequências, ignoram-se propriedades bioquímicas relevantes de proteínas; e
- não constrói um modelo de predição que auxilie biológos no entendimento das propriedades e relacionamento que determinaram a função biológica da proteína.

Modelos de predição baseados em indução de regras representam uma alternativa aos modelos baseados em busca por similaridades. Técnicas para indução de regras despertam o interesse de pesquisadores, principalmente por representarem conhecimento compreensível que auxiliam no entendimento dos relacionamentos que determinam funções biológicas de proteínas. Estes relacionamentos são representados na forma de regras do tipo SE *antecedente* ENTÃO *consequente*, onde o antecedente indica padrões encontrados na base de dados que determinam a classe, expressa no consequente, para exemplos da base (WITTEN; FRANK, 2005).

Em geral, os classificadores construídos associam apenas uma classe para um exemplo cuja classe não é conhecida. Estes classificadores são conhecidos como *single-label*. Entretanto, no contexto da predição de funções biológicas para proteínas, este tipo de método de classificação

é bastante limitado, pois proteínas podem executar inúmeras funções em organismos. Modelos de classificação que permitam associar uma ou mais classes para um exemplo são chamados de classificadores *multi-label* (TSOUMAKAS; KATAKIS, 2007).

Basicamente, dois tipos de classificadores *multi-label* são descritos na literatura. O primeiro, chamado de local ou binário, constrói um classificador para cada classe prevista. Assim, uma proteína é submetida a vários classificadores para prever sua(s) função (funções) biológica(s). No segundo caso, chamado de global, um único classificador é construído para prever as classes do domínio da aplicação. Portanto, uma proteína é submetida a um único classificador que possui a capacidade de retornar todas as funções biológicas daquela proteína (KIRITCHENKO, 2005).

Atualmente, tem sido cada vez mais frequente o uso de ontologias em predição de funções de proteínas. Uma ontologia é um conjunto (vocabulário) de termos descrevendo o domínio de aplicação em uma forma padronizada e organizada (SMITH, 2003). Os diversos projetos da área de biologia têm gerado uma grande quantidade de dados biológicos, e consequentemente uma proliferação de bases orientadas para este fim (The Gene Ontology Consortium, 2004). Com o aumento do número de bases biológicas surgiram basicamente dois problemas: falta de integração entre as bases e não-padronização dos dados anotados (armazenados) nestas bases. O uso de ontologias permite adequar um vocabulário comum que facilita a comunicação entre os pesquisadores da área de biologia e os diversos usos ou propósitos destas informações para o processamento das mesmas. Dentre as ontologias criadas para o uso na biologia, a ontologia gênica (GO – *Gene Ontology*) tem obtido destaque em diversos trabalhos para predição de funções de proteínas (LEWIS, 2004).

A ontologia gênica é um vocabulário de termos consistente e estruturado para descrever domínios chaves da biologia molecular, incluindo atributos e produtos gênicos, assim como sequências biológicas. Estes termos podem ser anotados para sequências, genes ou produtos gênicos armazenados em bases de dados biológicos. Os termos da GO são estruturalmente representados na forma de um grafo acíclico direcionado, no qual um termo “filho” pode estar conectado a um ou mais “pais”.

Há muito tempo os sistemas biológicos vêm servindo como fonte de inspiração para o desenvolvimento de ferramentas computacionais poderosas para a solução de problemas complexos, incluindo aqueles que a bioinformática se propõe a resolver. Neste sentido destaca-se a área da computação natural que inclui todas as estratégias inspiradas por algum mecanismo biológico ou natural (DE CASTRO, 2006). Como exemplos têm-se as redes neurais artificiais (HAYKIN, 1999), a computação evolucionária (BäCK; FOGEL; MICHALEWICZ,

2000a, 2000b), os sistemas nebulosos (PEDRYCZ; GOMIDE, 1998) e os sistemas imunológicos artificiais (DE CASTRO; TIMMIS, 2002).

Os Sistemas Imunológicos Artificiais (SIA) são um paradigma computacional relativamente novo e foram propostos para resolver problemas do mundo real a partir do uso de princípios e teorias do sistema imunológico biológico (DASGUPTA, 1999; DE CASTRO; TIMMIS, 2002). Uma vantagem do SIA sobre outras estratégias computacionais de busca se deve ao fato da capacidade de manter a diversidade da população de soluções candidatas e encontrar boas soluções simultaneamente, caso elas existam (CASTRO et al., 2005).

Esta tese apresenta um inédito método computacional que utiliza conceitos de SIA para a construção de classificadores hierárquicos e *multi-label* de funções de proteínas descritas na GO. Esta nova técnica é avaliada sobre 10 bases de dados, sendo 9 de domínio público e 1 construída exclusivamente para uso nos resultados apresentados nesta tese.

O método proposto foi comparado com outros 2 algoritmos apresentados na literatura: PART e Clus. Os resultados computacionais mostraram superioridade do SIA proposto frente ao PART, porém este desempenho não se repetiu nos experimentos comparativos com o Clus.

1.1 OBJETIVOS

O objetivo principal desta tese é apresentar um novo algoritmo de indução de regras, descobertas por um sistema imunológico artificial, para classificação hierárquica e *multi-label*. Como objetivos específicos pretende-se abordar os seguintes itens:

- Estudar a capacidade do método proposto em construir classificadores globais e locais;
- descobrir conjuntos de regras com menor número de regras e condições possível, porém não em detrimento da correção preditiva;
- aplicar e avaliar o algoritmo proposto sobre o problema de predição de funções de proteínas;
- avaliar a capacidade do método proposto em lidar com classes a serem preditas hierarquicamente organizadas em um DAG – *Directed Acyclic Graph*, que representam estruturas potencialmente mais complexas para construção de classificadores;
- apresentar um algoritmo que permite ao usuário a flexibilidade de escolha entre o uso de *matching* parcial ou total.
- identificar as vantagens e desvantagens na utilização do *matching* parcial e total.

1.2 CONTRIBUIÇÕES

Os resultados preliminares obtidos com a abordagem proposta foram publicados em Alves et al. (2007), Alves, Delgado e Freitas (2008). Este trabalho apresenta as seguintes contribuições:

- A principal contribuição desta tese é a formalização e apresentação de um novo algoritmo baseado em SIA para classificação hierárquica e *multi-label*, chamado de MHCAIS (*Multi-label Hierarchical Classification with an Artificial Immune System*).
- O MHCAIS pode ser considerado o primeiro SIA para classificação hierárquica e *multi-label*.
- Seguindo esta linha de originalidade, o MHCAIS é o primeiro algoritmo para classificação hierárquica e *multi-label* de indução de regras usando o procedimento de extração sequencial de regra (*sequential covering*).
- A técnica proposta constrói tanto classificadores globais, quanto locais. Os experimentos computacionais são apresentados para análise de desempenho de ambos.
- As classes a serem preditas são hierarquicamente organizadas em um grafo acíclico direcionado, que potencialmente torna mais complexa a construção de classificadores.
- Os classificadores construídos são expressos na forma de regras SE *antecedente* ENTÃO *consequente* e possibilitam ao usuário o entendimento das previções realizadas.
- Análise do comportamento do MHCAIS com uso de *matching* parcial e total.
- Experimentos computacionais comparativos com algoritmos disponíveis na literatura.
- Criação de uma nova base de proteínas que será futuramente disponibilizada para uso da comunidade acadêmica.

1.3 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em 5 capítulos. O Capítulo 2 apresenta a fundamentação teórica para a compreensão da pesquisa realizada, e envolve conceitos gerais de biologia molecular, bioinformática, ontologia gênica, mineração de dados, sistemas imunológicos artificiais e trabalhos relacionados. O Capítulo 3 relata em detalhes o algoritmo proposto baseado em SIA

para classificação hierárquica e *multi-label*. O capítulo 4 descreve a metodologia dos experimentos computacionais, bases e critérios de avaliação. O Capítulo 5 apresenta os resultdos dos experimentos computacionais, além das análises realizadas. Finalmente, o Capítulo 6 traz as conclusões e trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados conceitos envolvendo proteínas (no contexto biológico) e como a bionformática, através do uso de técnicas de mineração de dados e da ontologia gênica, trata o problema de predição de funções biológicas destas estrutura moleculares. Os sistemas imunológicos artificias, como técnica computacional para solução de problemas do mundo real, serão detalhados. Por fim, abordagens relacionadas com este trabalho serão discutidas.

2.1 PROTEÍNAS

Uma proteína analisada sob o aspecto molecular é uma cadeia de aminoácidos (STANSFIELD, 1985). Estes aminoácidos são frequentemente chamados de primários, padrões ou normais e são assim denominados para que possam ser distinguidos de outros aminoácidos presentes em organismos vivos, mas não em proteínas (LEHNINGER; NELSON; COX, 2005). Os aminoácidos primários existentes são 20, cada qual representado por um acrônimo de uma ou três letras (Tabela 1) e com propriedades químicas distintas. A estrutura molecular dos aminoácidos é formada por um grupo amina primário (NH_2) e um grupo carboxila (COOH) como ligantes em um mesmo átomo de carbono (carbono α). Esta parte é comum a todos os aminoácidos. Em uma proteína cada aminoácido é conectado a outro através de ligação covalente peptídica. Esta parte repetitiva em uma sequência de proteínas é conhecida como *cadeia polipeptídica*. Os aminoácidos diferem uns dos outros quanto à estrutura de suas cadeias laterais ou grupamentos **R** (STRACHAN; ANDREW, 2002). A Figura 1 ilustra a estrutura básica de um aminoácido.

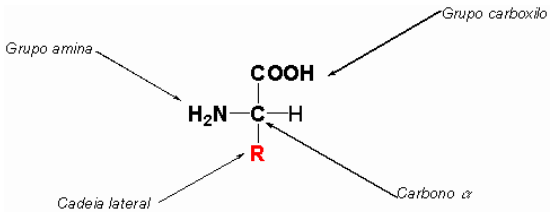


Figura 1: Estrutura básica de um aminoácido.

Nos organismos vivos os 20 aminoácidos possuem propriedades químicas diferentes, especialmente no que diz respeito à sua hidrofobicidade, tamanho e carga (BACKOFEN; GILBERT, 2001). Na presença de água, aminoácidos hidrofóbicos (não-polares) tendem a se manter juntos para minimizar o contato com a água. Aminoácidos hidrofílicos (polares) não apresentam o mesmo comportamento que os hidrofóbicos ao contato com água. Os hidrofílicos podem ser carregados positivamente, negativamente ou eletricamente neutros (ALBERTS et al., 2002). Os nomes dos aminoácidos são apresentados na Tabela 1 com seus respectivos acrônimos e tipos de cadeias laterais.

Tabela 1: Tabela de aminoácidos.

Hidrofílicos ou Polares				Hidrofóbicos ou não-Polares			
Aminoácido			Cadeia lateral	Aminoácido			Cadeia lateral
Ácido Aspártico	Asp	D	negativa	Alanina	Ala	A	não-polar
Ácido Glutâmico	Glu	E	negativa	Glicina	Gly	G	não-polar
Arginina	Arg	R	positiva	Valina	Val	V	não-polar
Lisina	Lys	K	positiva	Leucina	Leu	L	não-polar
Histidina	Hys	H	positiva	Isoleucina	Ile	I	não-polar
Asparagina	Asn	N	neutra	Prolina	Pro	P	não-polar
Glutamina	Gln	Q	neutra	Fenilalanina	Phe	F	não-polar
Serina	Ser	S	neutra	Metionina	Met	M	não-polar
Treonina	Thr	T	neutra	Triptofano	Trp	W	não-polar
Tirosina	Tyr	Y	neutra	Cisteína	Cys	C	não-polar

As proteínas são formadas através de ligações covalentes[1] entre aminoácidos, conhecidas como *ligações peptídicas*. Uma ligação peptídica é formada com a perda de um grupo hidroxila (OH) do grupo carboxila de um aminoácido e a perda de um hidrogênio (H) do grupo amina do outro aminoácido, formando água como um coproduto da reação (ALBERTS et al., 2002), conforme ilustra a Figura 2. Como uma proteína é formada por vários aminoácidos, ela pode ser considerada uma cadeia polipeptídica. As unidades de aminoácidos presentes em proteínas são chamadas de *resíduos de aminoácidos*, pois não correspondem mais aos aminoácidos originais já que perderam um hidrogênio do grupo amina e parte do grupo carboxila (LEHNINGER; NELSON; COX, 2005). O grupo amina do primeiro aminoácido e o grupo carboxila do último em uma cadeia polipeptídica permanecem sempre inalterados, o que dá uma orientação direcional na proteína. Assim, a proteína sempre começa com um grupo amina, também chamado de terminal-N, e termina com um grupo carboxila, ou terminal-C.

[1] Quando dois átomos compartilham um par de elétrons.

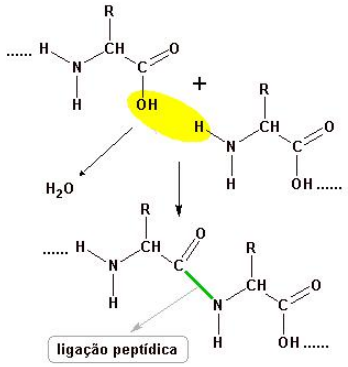


Figura 2: Ligação peptídica de dois aminoácidos.

2.1.1 FUNÇÕES DE PROTEÍNAS

As proteínas são máquinas moleculares e desempenham papéis cruciais em praticamente todos os processos biológicos, pois executam e controlam as funções essenciais nos organismos vivos; por isso estão no centro da ação nos processos biológicos. A sequência de aminoácidos de uma proteína particular não é absolutamente fixa. Estima-se que na espécie humana 20 a 30% das proteínas sejam polimórficas, tendo sequência variável de aminoácidos. Muitas destas variações de sequência têm pouco ou nenhum efeito sobre a função da proteína (ALBERTS et al., 2002).

As principais funções das proteínas são (STRYER; TYMOCZKO; BERG, 2004):

Catálise enzimática: quase todas as reações químicas em sistemas biológicos são catalisadas por enzimas, que geralmente aumentam a velocidade das reações.

Transporte e armazenamento: muitas moléculas pequenas são transportadas por proteínas específicas.

Movimento coordenado: os principais componentes dos músculos são proteínas. A contração muscular é conseguida pelo movimento de deslizamento de dois tipos de filamentos protéicos.

Sustentação mecânica: a alta força de tensão da pele e do osso é devido a presença de uma proteína fibrosa, o colágeno.

Proteção imunológica: os anticorpos são proteínas específicas que reconhecem substâncias estranhas, como vírus e bactérias.

Geração e transmissão de impulsos nervosos: a resposta de células nervosas a estímulos específicos é intermediada por proteínas receptoras.

Nos experimentos computacionais realizados neste trabalho são utilizados dados de proteínas que exercem funções de catálise enzimática.

2.1.2 ESTRUTURA ORGANIZACIONAL

As funções desempenhadas pelas proteínas nos organismos estão relacionadas diretamente às suas estruturas tridimensionais, que por sua vez dependem da sequência de aminoácidos que as compõem (BRANDEN; TOOZE, 1999). Assim, proteínas são descritas na literatura de acordo com 4 diferentes níveis crescentes de complexidade (LEHNINGER; NELSON; COX, 2005). Estes níveis são chamados: estrutura primária, secundária, terciária e quaternária conforme ilustra a Figura 3.

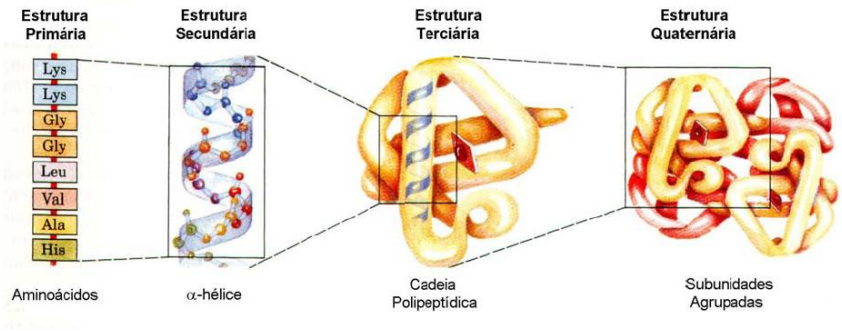


Figura 3: Níveis estruturais de uma proteína (LEHNINGER; NELSON; COX, 2005).

A **estrutura primária** é o nível estrutural mais simples e mais importante, pois dela deriva todo o arranjo espacial da proteína, e é dada pela sequência de aminoácidos e ligações peptídicas da molécula. Este arranjo é específico para cada proteína, sendo em geral determinado geneticamente. Duas proteínas podem se diferenciar em três aspectos: número, sequência (ordem) e natureza dos aminoácidos. A **estrutura secundária** é formada pelo arranjo espacial de aminoácidos próximos entre si na sequência primária da proteína. Os dois principais arranjos secundários são chamados de α-hélice e folhas-β; e podem acontecer de forma regular em uma proteína. A **estrutura terciária** é definida pelo arranjo espacial de aminoácidos distantes entre si na sequência polipeptídica e determina a forma tridimensional de uma proteína, mantendo a estrutura secundária da sequência. A **estrutura quaternária** é criada através de associações de proteínas já organizadas em nível terciário. Proteínas conseguem exercer função quando atingem o nível terciário. Entretanto, algumas apenas conseguem executar suas funções quando associadas a outras já em nível terciário (LEHNINGER; NELSON; COX, 2005).

2.2 BIOINFORMÁTICA

A bioinformática pode ser definida como uma área multidisciplinar que se baseia no conhecimento especialista de matemáticos, cientistas da computação, biólogos, físicos, entre outros (FOGEL; CORNE, 2003). Os avanços no sequenciamento do genoma de diversas espécies e o aumento exponencial de informações biológicas armazenadas em bases de dados podem ser considerados os principais motivadores para o surgimento da bioinformática. A bioinformática tem como objetivo a aplicação e o desenvolvimento de técnicas e métodos computacionais para a geração, análise, modelagem e extração de informações armazenadas em bases de dados biológicos no contexto da interpretação e predição (HUGHEY; KARPLUS, 2001). Vários são os problemas que têm sido tratados pela bioinformática, entre estes se destacam: alinhamento de sequências, reconstrução de árvores filogenéticas, análise de dados de expressão gênica, análise de interação entre genes, predição de estruturas tridimensionais de proteínas, predição de função de proteínas (discutido em detalhes na seção 2.2.1), entre outros.

2.2.1 DETERMINAÇÃO E PREDIÇÃO DE FUNÇÃO DE PROTEÍNAS

A predição e compreensão das funções de proteínas são umas das principais áreas de pesquisa da biologia molecular por diversas razões, tais como: um melhor entendimento de doenças, desenvolvimento de medicamentos mais efetivos, medicina preventiva, entre outras. Em laboratório, a *determinação* destas funções é geralmente realizada por métodos experimentais, como cristalografia de raios-X ou ressonância magnética nuclear; ou modelos teóricos, como modelagem molecular por homologia (baseado em conhecimento prévio de proteínas similares) ou *ab-initio* (totalmente teórico) (ALBERTS et al., 2002). Entretanto, o grande volume de dados armazenados nas bases biológicas torna inviável a determinação manual de funções para cada proteína armazenada. Assim, diversos estudos em bioinformática têm sido realizados no intuito de desenvolver métodos computacionais capazes de realizar a *predição* destas funções, conforme ilustra Figura 4.

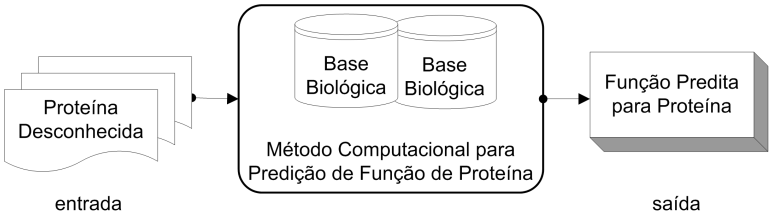


Figura 4: Predição de função de uma proteína desconhecida.

Os métodos utilizados para predição de função de proteínas devem ser precisos, úteis, tra-

balhar com enormes quantidades de dados e interpretáveis ou *transparentes* (SZAFRON et al., 2003). Estas características são as mesmas que aquelas desejáveis em mineração de dados (seção 2.5.1) (FREITAS, 2002). Assim, diversos métodos de aprendizado de máquina[2] têm sido aplicados para atender esta demanda, tais como: as máquinas de vetores de suporte (SVM - *Support Vector Machines*), árvores de decisão, redes neurais, algoritmos genéticos, redes bayesianas, entre outros.

2.3 BASES DE DADOS BIOLÓGICOS

As bases de dados biológicos armazenam as informações biológicas obtidas através de experimentos em laboratório ou métodos computacionais. O volume de dados armazenados aumenta constantemente de maneira exponencial (EISNER et al., 2005). As principais motivações para armazenar estas informações em bases de dados são: disponibilizar informações para cientistas de todo o mundo, tornando mais fácil a busca destas informações; e disponibilizar dados em um formato que possa ser lido por computadores. Deste modo, os dados contidos nestas bases podem ser vistos como matéria prima para a bioinformática. Atualmente, há várias bases de dados biológicos. Algumas contêm informações mais gerais e outras mais específicas, por exemplo, informações de apenas um tipo de organismo vivo. Cada base de dados contém informações voltadas para o objetivo ao qual foi criada. De maneira geral as bases de dados biológicos podem ser divididas em:

- Bases de sequências de nucleotídeos;
- Bases de sequências de proteínas;
- Bases de motivos (*motifs*) ou padrões; e
- Bases de estruturas.

Cada registro em uma base de dados biológica é chamado de *entrada* e as informações associadas a este registro *anotações*. Por exemplo, uma proteína *X* pode ter vários termos da ontologia gênica (seção 2.4) anotados. Algumas bases de dados mais gerais contêm informações biológicas voltadas para o motivo pelo qual a base foi criada; informações de outras bases através de referências cruzadas; e informações bibliográficas, como por exemplo, artigos relacionados a uma determinada proteína.

[2] **Aprendizado de Máquina** estuda como construir programas de computador que melhoram seu desempenho em alguma tarefa, por meio de experiência (MITCHELL, 1997)

2.3.1 BASES DE SEQUÊNCIAS DE PROTEÍNAS

Dentre as bases de dados de proteínas mais gerais, destacam-s as bases Swiss-Prot/TrEMBL[3] [4], PIR[5] e UniProt [6] [7].

2.3.1.1 *SWISS-PROT/TREMBL*

A base Swiss-Prot foi criada em 1986 e desde 2003 é mantida pelos SIB (*Swiss Institute of Bioinformatics*) e EBI (*European Bioinformatics Institute*). Basicamente, esta base mantém um alto nível de anotações, com o mínimo de redundância e integração com outras bases biológicas (BOECKMANN et al., 2003). Cada entrada da base é composta por duas partes. A primeira contém informações como sequências, referências bibliográficas e taxonomia (descrição da fonte biológica da proteína). A segunda é composta pelas anotações associadas àquela entrada, como por exemplo, funções da proteína, domínios e sítios, estrutura secundária e quaternária, similaridade com outras proteínas, etc. Todas as informações associadas a entrada mantêm um formato padronizado, o mais semelhante possível, com a base de sequências de nucleotídeos EMBL[8] (*European Molecular Biology Laboratory*). Para obter redundância mínima e confiabilidade de uma sequência, todas as proteínas codificadas por um mesmo gene são agrupadas em uma única entrada. A integração com outras bases biológicas é realizada através de referências cruzadas, tais como, bases de sequências de nucleotídeos (EMBL, GenBank[9]), bases de estruturas tridimensionais (PDB[10]), padrões e domínios (PROSITE, PFAM, PRODOM). Atualmente, a base Swiss-Prot possui integração com aproximadamente 60 bases de dados[11].

O TrEMBL (*Translated* EMBL) é um suplemento do Swiss-Prot com o objetivo de disponibilizar o mais rapidamente possível os dados de novas estruturas para eventuais anotações sem comprometer a qualidade das estruturas já incluídas no Swiss-Prot. Enquanto estiverem no TrEMBL, as estruturas estarão sendo verificadas e estudadas para evitar que sejam adicionadas no Swiss-Prot estruturas incorretas ou duplicadas.

[3] http://expasy.org/sprot/
[4] http://www.ebi.ac.uk/swissprot/
[5] http://pir.georgetown.edu/
[6] http://www.pir.uniprot.org/
[7] http://www.ebi.uniprot.org/index.shtml
[8] http://www.ebi.ac.uk/embl/
[9] http://www.ncbi.nlm.nih.gov/GenBank
[10] http://www.rcsb.org/pdb
[11] http://www.expasy.org/cgi-bin/lists?dbxref.txt

2.3.1.2 *PIR*

A base PIR (*Protein Information Resource*) foi criada em 1984 através do esforço conjunto do NBRF (*National Biomedical Research Foundation*), MIPS (*Munich Information Center for Protein Sequences*) e JIPID (*Japan International Protein Information Database*) como uma fonte para auxiliar na identificação e interpretação das informações de sequências de proteínas. Esta base foi criada com os seguintes objetivos (BARKER et al., 1998):

- manter uma base de dados abrangente, não redundante e biologicamente organizada, incluindo relacionamentos evolucionários, estruturais e funcionais;
- distribuir a base de dados por diversos meios, como por exemplo a internet, CDs, etc., e colaborar com outras organizações com objetivos semelhantes; e
- fornecer uma ferramenta de pesquisa para estudos de proteínas.

A base PIR é subdividida em outras três bases de dados: the *Protein Sequence Database* (PIR-PSD), *Non-redundant Reference* (PIR-NREF) e *Integrated Protein Classification* (PIR-iProClass). A base PIR-PSD contém sequências de proteínas cobrindo diversos organismos. Estas entradas são classificadas em famílias provendo uma melhora na identificação e detecção de erros, e anotações consistentes. A base PIR-PSD possui também um sistema que atribui e mapeia referências bibliográficas para as entradas. A base PIR-NREF é uma base sem redundâncias de referências com outras bases biológicas, entretanto esta base deixou de ser atualizada a partir de 16 de janeiro de 2006. A base PIR-iProClass contém informações de famílias de proteínas, funções, estruturas e ontologias de aproximadamente 90 outras bases de dados.

2.3.1.3 *UNIPROT*

A base UniProt (*Universal Protein Resource*) é vista como um repositório central de proteínas e foi criada em 2002 através de um esforço conjunto do EBI, PIR e SIB dando origem ao *UniProt Consortium*. A UniProt contém informações interligadas com diversas outras bases biológicas e é dividida em três camadas de bases de dados: UniProt *Knowledge Base* (UniProtKB), UniProt *archive* (UniParc) e UniProt *Reference Clusters* (UniRef) (The UniProt Consortium, 2007).

A base UniProtKB é um agrupamento, sem redundâncias, das proteínas armazenadas nas bases SwissProt/TrEMBL e PIR, mantendo os objetivos originais pelos quais cada uma foi criada. Assim como as bases originais, cada entrada possui informações da sequência, descrições funcionais, bibliografias, anotações, e referências cruzadas com outras bases de dados.

A base UniParc contém o histórico, sem redundâncias e de forma compreensível, de todas as sequências de proteínas ao longo do tempo. Cada entrada na base representa uma sequência de proteínas. Entretanto, para evitar redundância de dados, proteínas com 100% de similaridade na sequência são agrupadas em uma mesma entrada. Como exemplo, há casos onde organismos diferentes possuem a mesma sequência de proteína, e proteínas diferentes com a mesma sequência podem desempenhar funções distintas. Basicamente, cada entrada contém as proteínas associadas àquela sequência; a base de dados de onde a sequência foi originada; indicativo se a sequência está ativa ou obsoleta; e referências cruzadas que permitem o acesso da sequência em sua base de dados original. A base UniParc não contém anotações.

A base UniRef contém as informações das bases UniParc e UniProtKB; e tem como objetivo agrupar sequências com alto nível de similaridade. Esta base é dividida em outras três: UniRef100 agrupando sequências com 100% de similaridade; UniRef90 agrupando sequências com nível de similaridade $> 90\%$; e UniRef50 agrupando sequências com nível de similaridade $> 50\%$.

A Figura 5 ilustra uma visão geral das três camadas da base de dados UniProt e apresenta o esquema de interação entre elas.

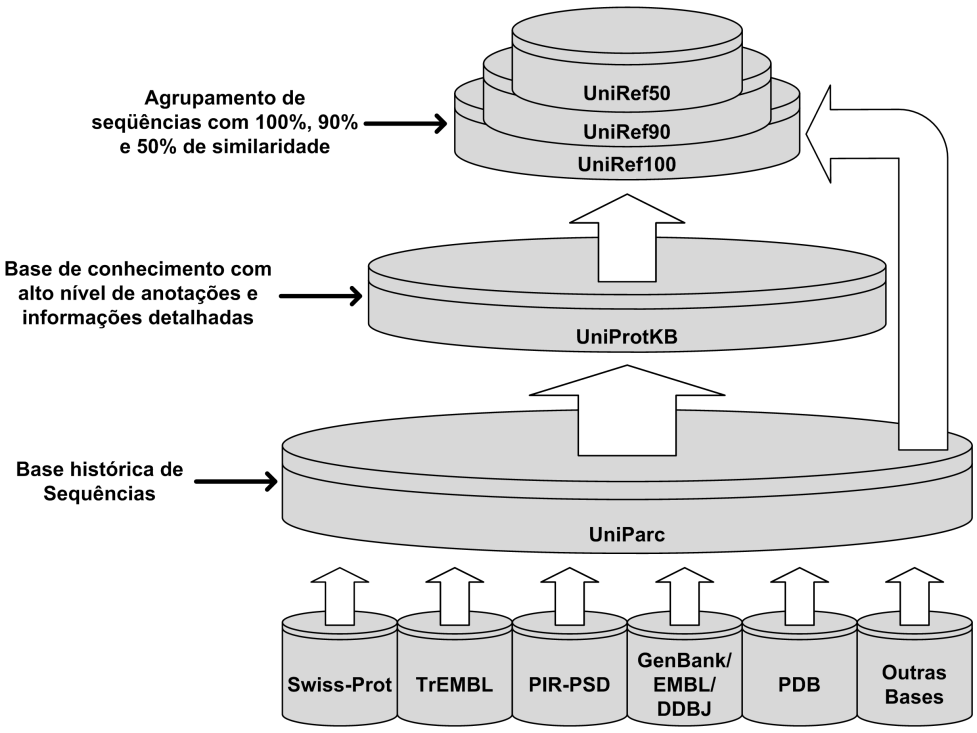


Figura 5: Base UniProt dividida em suas três camadas.

2.3.2 BASES DE PADRÕES DE PROTEÍNAS

Um padrão ou motivo (*motifs*) é um tipo de informação caracterizando uma região localmente conservada de uma sequência de proteína. Geralmente, um padrão caracteriza a função de uma molécula, uma característica estrutural ou membro de uma família. Os motivos são gerados a partir de alinhamentos de várias sequências e podem ser representados como padrões flexíveis, matrizes de pontuação específicas da posição (perfis) ou Cadeias Escondidas de Markov (*Hidden Markov Models*), conforme será detalhado posteriormente. Através de alinhamento de proteínas é comum encontrar não apenas um padrão, mas um conjunto de padrões de uma mesma família. Assim, este conjunto de padrões é agrupado em *assinaturas* ou *impressão digital* desta família. As principais bases de padrões são: PROSITE[12], PFAM[13], PRODOM[14], PRINTS[15] e SMART[16], entre outras.

2.3.2.1 *PROSITE*

A base PROSITE armazena dois tipos de motivos: padrões flexíveis e perfis. Cada entrada na base de dados possui descrição sobre a família ou domínio onde o motivo foi encontrado, origem do nome dado, taxonomia, função, principais características da sequência, referências, entre outras informações (HULO et al., 2006). A base PROSITE é derivada da Swiss-Prot e se baseia na filosofia de que famílias de proteínas podem ser caracterizadas pelo motivo mais conservado durante a evolução. Estes motivos estão geralmente associados à função biológica da proteína, como por exemplo, sítios catalíticos de enzimas e regiões de ligações com outras moléculas ou proteínas (SIGRIST et al., 2002).

Os padrões flexíveis são codificados na forma de *expressões regulares* e são compostos por 10 a 20 aminoácidos aproximadamente. Como exemplo de uma expressão regular tem-se: M–x–G–x(3)–[IV]2–x(2)–{FWY}. Em linguagem natural, interpreta-se a expressão da seguinte forma: metionina – 1 aminoácido qualquer – glicina – 3 aminoácidos quaisquer seguidos – dois aminoácidos seguidos, desde que sejam isoleucina ou valina – 2 aminoácidos quaisquer seguidos – 1 aminoácido qualquer, desde que não sejam fenilalanina, triptofano ou tirosina. Para que uma proteína seja anotada com um padrão PROSITE é necessário que a expressão regular toda, respeitando inclusive a posição dos aminoácidos, ocorra totalmente na sequência.

Os perfis foram criados na base PROSITE devido à dificuldade de encontrar padrões que

[12] http://www.expasy.org/prosite/
[13] http://www.sanger.ac.uk/Software/Pfam/
[14] http://prodom.prabi.fr/prodom/current/html/home.php
[15] http://www.bioinf.manchester.ac.uk/dbbrowser/PRINTS/
[16] http://smart.embl-heidelberg.de/

satisfaçam toda a expressão regular em algumas sequências. Assim, perfis são métodos alternativos construídos também através de alinhamentos e representados na forma de matrizes de peso. Este perfis definem:

- quais aminoácidos são permitidos e em que posição;
- quais posições são muito ou pouco conservadas;
- quais posições são degeneradas; e
- quais posições ou regiões podem tolerar inserções.

2.3.2.2 *PFAM*

A base PFAM é originada do Swiss-Prot e possui uma coleção de alinhamentos múltiplos de sequências e perfis baseados em Cadeia de Markov Escondida (HMM - *Hidden Markov Model*) de famílias de proteínas (BATEMAN et al., 2004). HMM's são modelos probabilísticos que consistem em coleções de *estados* intercomunicantes nos quais as transições entre estes estados ocorrem segundo algum mecanismo probabilístico. Os estados possíveis em alinhamentos múltiplos são: "ocorrência" (*match*), "remoção" e "inserção". Atribui-se o estado "ocorrência" a cada coluna conservada no alinhamento das sequências. O estado "inserção" permite inserções relativas ao estado "ocorrência" enquanto que, o estado "remoção" permite que posições de "ocorrências" (*matched*) sejam ignoradas. Assim a construção de um HMM para um alinhamento múltiplo entre sequências requer que cada posição dentro do alinhamento esteja associada a um dos três estados definidos.

A base PFAM é composta de duas partes: PFAM-A e PFAM-B. A primeira parte, PFAM-A, é curada e contém famílias de proteínas bem caracterizadas e com alta qualidade de alinhamento; e mantidas manualmente. A segunda parte, PFAM-B, contém pequenas famílias da base PRODOM que não estão na PFAM-A. Embora as famílias da PFAM-B sejam de menor qualidade, elas podem ser úteis quando não são encontradas na PFAM-A. Para cada família de proteínas na base PFAM é possível: observar os alinhamentos múltiplos, examinar a distribuição de espécies, visualizar estruturas e domínios de proteínas conhecidas e percorrer ligações com outras bases de dados (BATEMAN et al., 2004).

2.3.2.3 *PRODOM*

A base PRODOM consiste de conjuntos de domínios de proteínas gerados automaticamente das bases SwissProt/TrEMBL por métodos de agrupamentos de sequências. Basicamente, estes

padrões são encontrados por alinhamentos múltiplos para agrupar seguimentos de proteínas em domínios de famílias homólogas (BRU et al., 2005). Entretanto, como estes agrupamentos são gerados automaticamente, sem considerar informação biológica alguma, agrupamentos biologicamente inválidos podem ser gerados.

2.3.2.4 *INTERPRO*

A base InterPro (*Integrated Resource of Protein Families, Domains and Sites*) é um esforço conjunto para criar uma base integrada com o mínimo possível de redundância de famílias, domínios e sítios funcionais de proteínas. Esta base foi criada para integrar as maiores e mais importantes bases de assinaturas e padrões. Atualmente, a base InterPro é formada pelas seguintes bases: PROSITE, PFAM, PRINTS, PRODOM, SMART, TIGRMAs, PIRSF e SUPERFAMILY(Figura 6[17]). Estas assinaturas são manualmente curadas e integradas dentro da base InterPro provendo informações biológicas e funcionais. Informações sobre estruturas tridimensionais de proteínas são disponibilizadas, além de possuir referências cruzadas para a base UniProt (MULDER et al., 2005).

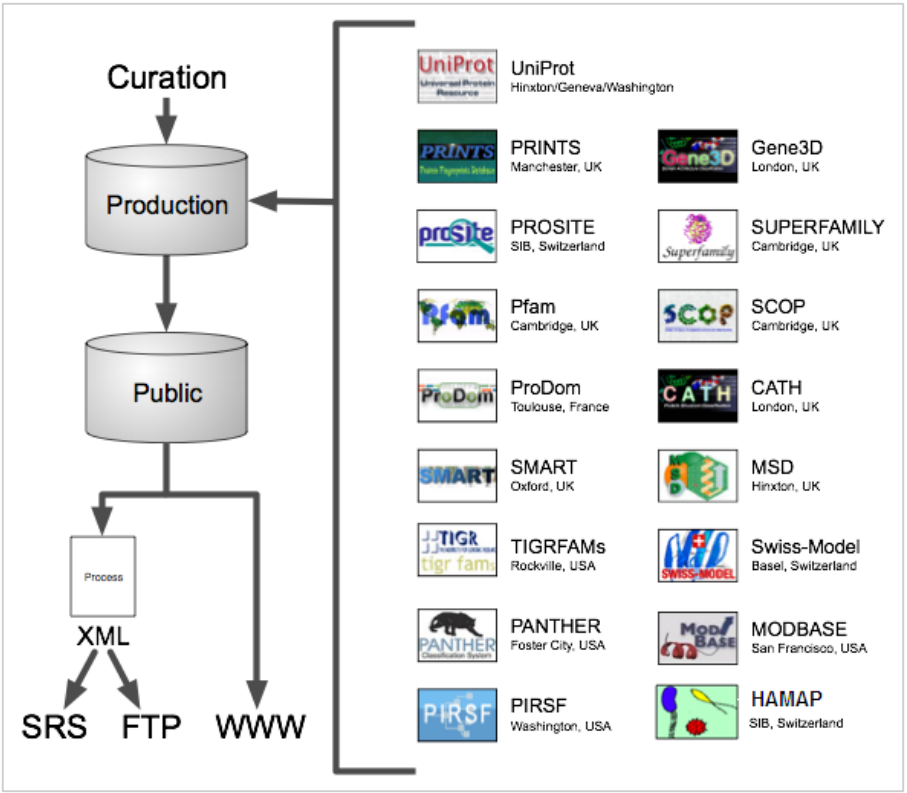


Figura 6: Fluxograma do relacionamento do InterPro com as demais bases.

Neste trabalho são utilizadas a base de proteínas Uniprot e a base de padrões PROSITE.

[17] http://www.ebi.ac.uk/interpro/project_outlines.html

2.4 ONTOLOGIA GÊNICA

Na filosofia, *ontologia* é o estudo da natureza dos tipos e estrutura dos seres, objetos, propriedades, eventos, processos e relacionamentos dentro de cada área a qual esta ontologia é aplicada (SMITH, 2003). O termo "ontologia" em Inteligência Artificial (IA) representa um vocabulário usado para descrever certa realidade entre "artefatos" e um conjunto de definições que consideram o significado pretendido dos termos que constituem este vocabulário (GUARINO, 1998). Assim, ao usar um termo desta ontologia com uma outra linguagem, o significado deve permanecer o mesmo, como por exemplo, traduzir um termo de um idioma para o outro. Atualmente, ontologias têm sido constantemente usadas em diversas áreas relacionadas a Sistemas de Informação (GUARINO, 1998), como por exemplo, a área de biologia com o uso da **ontologia gênica**.

Nos últimos anos, os diversos projetos da área de biologia têm gerado uma grande quantidade de dados biológicos, e consequentemente uma proliferação de base de dados orientadas para este fim (The Gene Ontology Consortium, 2004). Além disso, surgem outros dois problemas: falta de integração entre as bases e não-padronização dos dados anotados (armazenados) nestas bases. A partir deste momento, percebeu-se que o uso de ontologias poderia fornecer um conjunto de conceitualizações para atender esta demanda. Sendo assim, um vocabulário comum facilitaria a comunicação entre os pesquisadores da área de biologia e os diversos usos ou propósitos destas informações para o processamento das mesmas.

Por volta de 1998, surgiu um grupo de pesquisa conhecido como *Gene Ontology Consortium* (The Gene Ontology Consortium, 2004) com a preocupação de promover a integração dos dados biológicos sob dois aspectos: fornecer uma descrição consistente dos produtos gênicos em diferentes bases de dados e uma padronização na classificação de sequências e características de sequências. A partir deste momento surgiu a "ontologia gênica" ou *Gene Ontology* (GO)[18]. O projeto iniciou com apenas três bases de dados, mas este número tem aumentado constantemente (LEWIS, 2004).

A ontologia gênica é um vocabulário de termos consistente e estruturado para descrever domínios chaves da biologia molecular, incluindo atributos e produtos gênicos, assim como sequências biológicas. Estes termos podem ser anotados para sequências, genes ou produtos gênicos armazenados em bases de dados biológicos. A GO descreve atributos de produtos gênicos em três domínios disjuntos da biologia molecular: processo biológico, função molecular e componente celular. *Função molecular* descreve atividades realizadas por produtos gênicos no nível celular, tais como, atividades catalíticas ou de ligação. *Processo biológico* descreve

[18] http://www.geneontology.org/

uma série de eventos realizados por um ou mais grupos ordenados de funções moleculares. *Componente celular* descreve o local onde um produto gênico pode ser encontrado.

Os termos da GO são estruturalmente representados na forma de um grafo acíclico direcionado (DAG - *Directed Acyclic Graphs*), no qual um termo "filho" pode estar conectado a um ou mais "pais". Os termos se relacionam através de dois tipos de relacionamentos: "é um(a)" ou "parte de". O relacionamento "é um(a)" indica que um termo filho é uma instância do termo pai, como por exemplo, cromossomo mitótico é uma instância de um cromossomo. Por outro lado, o relacionamento "parte de" indica que um termo filho é um componente de um termo pai, como por exemplo, um telômero é parte de um cromossomo.

A Figura 7 ilustra graficamente um subgrafo da GO relativo ao domínío de termos associados à *função molecular*.

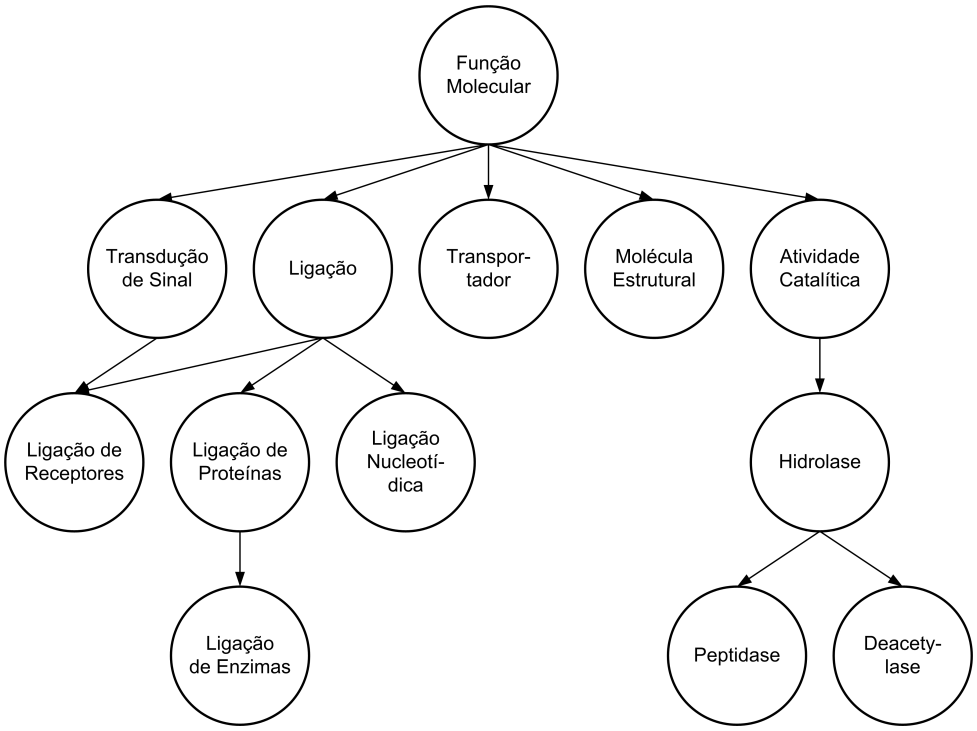


Figura 7: Parte da estrutura hierárquica da GO - Função Molecular.

Nesta tese são abordados apenas os termos descritos no domínio função molecular, uma vez que esses termos definem as funções biológicas de proteínas.

2.4.1 DEFINIÇÃO FORMAL DA ESTRUTURA HIERÁRQUICA DA GO

Formalmente a estrutura hierárquica da GO e os relacionamentos existentes são definidos da seguinte forma:

Definição 2.4.1 (Grafo) *Um grafo G consiste de um par (V,E), onde V é um conjunto de vértices (nodos) e E um conjunto de arestas, onde cada aresta representa uma relação* (u,v) *que liga um nodo u a um nodo v. Assim,* $E = \{(u,v)|u,v \in V\}$.

Definição 2.4.2 (DAG - (*Directed Acyclic Graph*)) *Em um DAG cada aresta tem uma direção de um nodo u para nodo v e não há ciclos neste grafo.*

A hierarquia de um DAG é formalizada por uma estrutura $\mathscr{H} = \langle V, \preceq \rangle$. A relação binária $\preceq$ define que os nodos $n \in V$ estão parcialmente ordenados. Assim, se $v < u$, então u é predecessor imediato de v ou v é sucessor imediato de u. Toda relação $\preceq$ por definição é:

- reflexiva: $u \preceq u$;
- anti-simétrica: se $v \preceq u$ e $u \preceq v$, então $u = v$, para todo $u,v \in V$;
- transitiva: se $x \preceq v$ e $v \preceq u$, então $x \preceq u$, para todo $u,v,x \in V$.

Definição 2.4.3 (Pai) *Dados os nodos* $u,v \in V$*, u é pai de v em* $\mathscr{H} = \langle V, \preceq \rangle$*, se* $v < u$ *e* $\nexists x \in V \mid v < x < u$.

Definição 2.4.4 (Filho) *Dado os nodos* $v,x \in V$*, x é filho de v em* $\mathscr{H} = \langle V, \preceq \rangle$*, se* $x < v$ *e* $\nexists y \in V \mid x < y < v$.

Definição 2.4.5 (Ancestral) *Para* $u \in V$ *em uma estrutura* $\mathscr{H} = \langle V, \preceq \rangle$*, o conjunto de nodos n que o antecede é formalizado por* $Ancestrais(u) = \{\bigcup n, \forall n \in V \mid u \preceq n\}$.

Definição 2.4.6 (Descendente) *Para* $v \in V$ *em uma estrutura* $\mathscr{H} = \langle V, \preceq \rangle$*, o conjunto de nodos m que o sucede é formalizado por* $Descendentes(v) = \{\bigcup m, \forall m \in V \mid m \preceq v\}$.

Formalizando a GO, os termos (nodos) são representados por c e o conjunto de termos por $\mathscr{C}$, assim $c \in \mathscr{C}$. A estrutura hierárquica é representada por $\mathscr{H} = \langle \mathscr{C}, \preceq \rangle$ e o relacionamentos no DAG são, respectivamente: pai, filho, ancestrais e descendentes.

2.5 MINERAÇÃO DE DADOS

A descoberta de conhecimento em bases de dados (*Knowledge Discovery in Databases* - KDD) tem se tornado um processo consideravelmente interessante para pesquisadores de diferentes áreas (FRAWLEY; PIATETSKY-SHAPIRO; MATHEUS, 1991; FAYYAD et al., 1996;

FREITAS, 2002; WITTEN; FRANK, 2005). Esse interesse se deve ao fato que o volume de dados armazenados em bases de dados é cada vez maior, o que torna a análise manual das informações contidas nestas bases ineficiente e muitas vezes inviável.

O processo de KDD envolve basicamente a extração de conhecimento possivelmente útil e previamente oculto em dados. Existe a tendência de se pensar, por esta definição, em uma nova técnica. Mas, na verdade, trata-se de um campo multidisciplinar de pesquisa (Figura 8), envolvendo áreas como aprendizado de máquina, estatística, bases de dados, sistemas especialistas e visualização de dados. Este processo utiliza métodos, algoritmos e técnicas oriundas de diversas áreas, com o objetivo principal de extrair conhecimento a partir de bases de dados do mundo real.

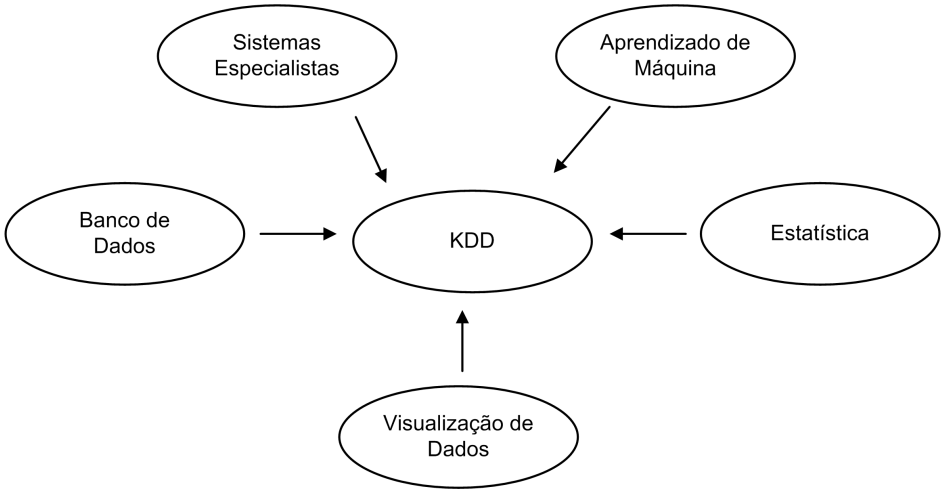


Figura 8: KDD - campo multidisciplinar.

Os termos KDD e mineração de dados (*data mining*) muitas vezes são vistos como sinônimos. Porém, em 1995, na Conferência Mundial de KDD em Montreal, foi definido como KDD todo o processo de extração de conhecimento dos dados, enquanto que a mineração de dados foi definida como uma etapa particular do processo de KDD (ADRIAANS; ZANTINGE, 1996) . O processo de KDD é constituído das seguintes etapas: preparação dos dados, seleção, limpeza, transformação, mineração de dados e interpretação dos resultados (FAYYAD et al., 1996).

2.5.1 CARACTERÍSTICAS DESEJÁVEIS DO CONHECIMENTO

O objetivo principal em mineração de dados é extrair conhecimento, portanto, torna-se necessário que este conhecimento descoberto tenha algumas características importantes. Atualmente, ainda não há um consenso sobre quais são as características realmente relevantes. No entanto, é possível mencionar que o conhecimento descoberto deve contemplar ao menos três

características (FREITAS, 2002): ser preciso (correção preditiva), compreensível e útil (interessante). O grau de importância de cada uma destas propriedades depende de vários fatores, tais como o domínio da aplicação e objetivos do usuário.

A correção preditiva especifica a acuidade do conhecimento descoberto. Apesar da importância da correção preditiva, em muitas aplicações há uma tendência para que a compreensibilidade e o grau de interesse do conhecimento descoberto tenham uma relevância maior, principalmente sob o ponto de vista do usuário deste conhecimento (FAYYAD et al., 1996; WITTEN; FRANK, 2005). A importância da compreensibilidade deve-se a dois fatores principais. Primeiro, o conhecimento descoberto tem como objetivo auxiliar o usuário em uma tomada de decisão. Segundo, se o conhecimento descoberto não é compreensível, o usuário se vê impossibilitado de validar e refinar este conhecimento. Finalmente, o grau de interesse ou utilidade se refere ao fato de que o conhecimento descoberto deve ser novo, surpreendente e potencialmente útil para o usuário. O conhecimento não é útil para o usuário se representar algo que já era anteriormente conhecido (redundante). Estas características desejáveis do conhecimento em mineração de dados são as mesmas características mencionadas para a predição de funções de proteínas (seção 2.2.1).

2.5.2 TAREFA DE CLASSIFICAÇÃO

Existem diferentes tipos de tarefas de mineração de dados, dentre elas pode-se citar: **classificação**, agrupamento (*clustering*) e regras de associação (WITTEN; FRANK, 2005). Cada tarefa tem como base um conjunto de algoritmos que são utilizados para extração de padrões relevantes dentro de uma massa de dados (CHEN; HAN; YU, 1996). As tarefas citadas diferem quanto aos tipos de problemas que cada algoritmo é capaz de resolver.

A classificação é considerada em aprendizado de máquina como uma técnica de *aprendizado supervisionado* (MITCHELL, 1997) e provavelmente é a mais estudada dentre as tarefas de mineração de dados. A tarefa de classificação consiste em associar a um exemplo (registro, caso ou objeto) uma ou mais classes, dentre um domínio de classes previamente definidas, tomando como base a observação dos atributos ou características de cada exemplo (FREITAS; LAVINGTON, 1998). Cada exemplo é composto por atributos previsores e atributo meta, cujos valores deste último representam classes previamente definidas. É importante que os atributos previsores tenham relevância suficiente para prever as classes (valores do atributo meta) de um exemplo.

No que diz respeito ao número de classes associadas a um exemplo da base de dados, a tarefa de classificação é dividida em duas vertentes: exemplos que são associados a uma

única classe (*single-label*) e exemplos que são associados a *várias* classes (***multi-label***). A classificação de exemplos para uma única classe se refere à tarefa tradicional de classificação, onde o objetivo é associar a cada exemplo apenas uma classe $c_k \in \mathscr{C}$, onde $|\mathscr{C}| > 1$ e $\mathscr{C} = \{c_1, \ldots, c_{|\mathscr{C}|}\}$. Se $|\mathscr{C}| = 2$, o problema é chamado de classificação binária, enquanto que, se $|\mathscr{C}| > 2$ é chamado de multi-classe (*multi-class*). Formalmente, a classificação *single-label* pode ser definida na seguinte forma (Tabela 2):

- a base de dados pode ser representada por $\mathbf{V} = \{\mathbf{X}_1, \ldots, \mathbf{X}_{|\mathbf{V}|}\}$, onde $|\mathbf{V}|$ representa o número de exemplos da base de dados;
- cada exemplo é dado por $\mathbf{X}_i = \langle \mathbf{x}_i, c_{ik} \rangle$, onde $i = 1, \ldots, |\mathbf{V}|$ e $k = 1, \ldots, |\mathscr{C}|$;
- os atributos previsores são tais que $\mathbf{x}_i = [x_{i1}, \ldots, x_{i|\mathscr{D}|}]^T$, onde $i = 1, \ldots, |\mathbf{V}|$, $x_{id} \in D_d$, $d = 1, \ldots, |\mathscr{D}|$, D_d é o conjunto dos valores válidos para o d-ésimo atributo e $|\mathscr{D}|$ representa o número de atributos previsores da base de dados;
- Cada valor do atributo meta (classe) $c_{ik} \in \mathscr{C}$;.

Tabela 2: Representação tabular de uma base de dados.

	$\mathbf{X}_i$, onde $i = 1, \ldots, \|\mathbf{V}\|$				
	Atributos Previsores ($\mathbf{x}_i$)				Classe ($c_{ik} \in \mathscr{C}$)
Exemplos ($\mathbf{V}$)	1	2	...	$\|\mathscr{D}\|$	$\mathscr{C} = \{c_{i1}, \ldots, c_{i\|\mathscr{C}\|}\}$
1	x_{11}	x_{12}	...	$x_{1\|\mathscr{D}\|}$	c_{1k}
2	x_{21}	x_{22}	...	$x_{2\|\mathscr{D}\|}$	c_{2k}
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$
$\|\mathbf{V}\|$	$x_{\|\mathbf{V}\|1}$	$x_{\|\mathbf{V}\|2}$	...	$x_{\|\mathbf{V}\|\|\mathscr{D}\|}$	$c_{\|\mathbf{V}\|k}$

A classificação associando um determinado exemplo a **várias** classes segue as mesmas definições anteriores, distinguindo-se apenas nos seguintes itens:

- cada exemplo é dado por $\mathbf{X}_i = \langle \mathbf{x}_i, C_i \rangle$, onde $i = 1, \ldots, |\mathbf{V}|$;
- Cada conjunto de valores do atributo meta (classes) $C_i \subseteq \mathscr{C}$, onde $|C_i| \geq 1$;

No passado, a classificação envolvendo várias classes estava restrita à *mineração de dados em textos* (MCCALLUM, 1999; SCHAPIRE; SINGER, 2000), onde por exemplo, um documento poderia pertencer às classes *política* e *economia*. Entretanto, várias aplicações têm surgido em contextos onde um exemplo pode estar associado a mais de uma classe. Por exemplo: *classificação de músicas* (LI; OGIHARA, 2003), onde um grupo musical ou uma música podem estar associados aos estilos de músicas *pop* e *rock*; *classificação semântica de cenas*

ou *imagens* (BOUTELL et al., 2004), onde uma fotografia pode pertencer às classes *pôr-do-sol* e *praia*; e **bioinformática** (JENSEN et al., 2003; VINAYAGAM et al., 2004; EISNER et al., 2005), onde uma proteína pode desempenhar várias funções biológicas (seção 2.1.1) no organismo.

Um importante conceito em classificação é a divisão, para fins de avaliação, da base de dados original (**V**) em dois conjuntos disjuntos, sendo um para treinamento (**S**), chamado de *base de treinamento*; e outro para teste (**T**) ou avaliação, chamado de *base de teste*, assim $\mathbf{V} = \mathbf{S} \cup \mathbf{T}$ e $\mathbf{S} \cap \mathbf{T} = \emptyset$. Inicialmente, o conjunto de treinamento **S** é disponibilizado para o algoritmo de classificação; e este algoritmo construirá um modelo para predição (conhecimento extraído) baseado nas características destes dados. Em seguida, este modelo é utilizado para classificar os exemplos que compõem o conjunto de teste **T**, onde será calculado o desempenho do algoritmo. O conjunto de teste **T** é formado por exemplos não vistos na fase de treinamento (FREITAS, 2002).

O principal objetivo na classificação é descobrir uma forte relação entre os atributos previsores e as classes. Mais formalmente, encontrar um classificador ou um conjunto de classificadores $h \in H$ que classifique $\mathbf{x}_i \rightarrow C$, maximizando a probabilidade de um exemplo estar associado a uma ou mais classes previstas para aquele exemplo. A Figura 9 ilustra uma visão geral de todo processo de classificação, e as etapas de treinamento e avaliação do modelo de classificação gerado por um algoritmo de classificação.

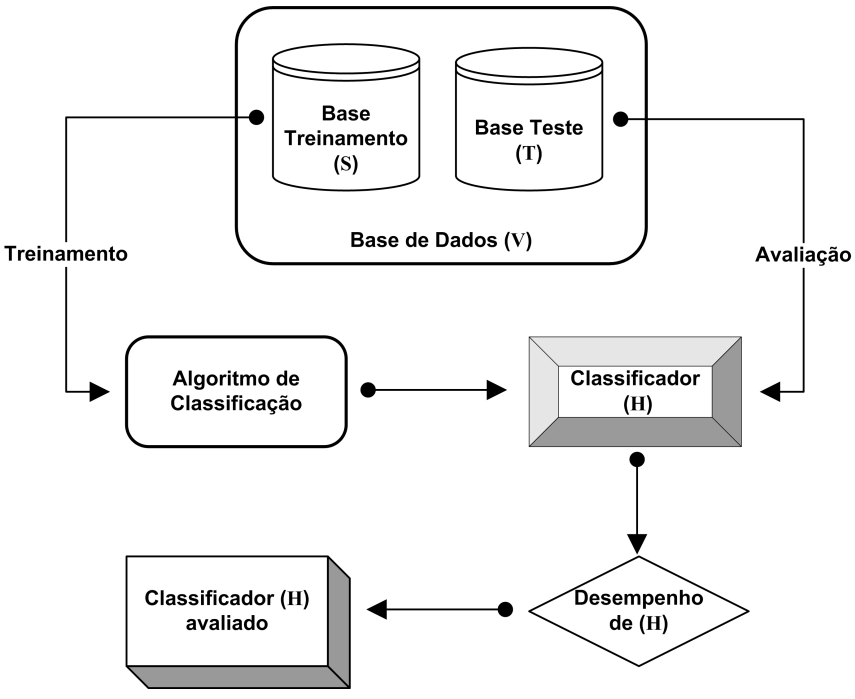


Figura 9: Visão geral do processo de classificação.

A tarefa de classificação tem sido amplamente aplicada em bioinformática, principalmente em predição de funções biológicas de proteínas (seção 2.2.1); e diversas técnicas de aprendizado de máquina têm sido usadas para tal: *métodos estatísticos* (redes bayesianas (KING et al., 2003; TROYANSKAYA et al., 2003; BARUTCUOGLU; SCHAPIRE; TROYANSKAYA, 2006), regressão logística (LU et al., 2004) e máquinas de vetores de suporte (VINAYAGAM et al., 2004; EISNER et al., 2005; BARUTCUOGLU; SCHAPIRE; TROYANSKAYA, 2006)); *árvores de decisão* (CLARE; KING, 2001; KING et al., 2003); *aprendizado baseado em instâncias* (K-vizinhos mais próximos (ZHANG; ZHOU, 2005; SCHUG et al., 2002)); *computação bioinspirada* (redes neurais (JENSEN et al., 2003; TU et al., 2004) e os algoritmos genéticos (LAEGREID et al., 2003)), entre outras. O Apêndice B descreve de forma geral algumas técnicas de aprendizado de máquina utilizadas em mineração de dados.

2.5.3 TIPOS DE CLASSIFICADORES

Em geral, classificadores (previsores) construídos são categorizados em dois tipos: local ou global. Na construção de classificadores *locais* (binários), para cada classe c do domínio de classes a serem previstas $\mathscr{C}$ um classificador h é descoberto (KIRITCHENKO, 2005). Assim, um exemplo t é submetido a todos os classificadores gerados a fim de identificar as classes associadas ao exemplo t. Em classificação *single-label*, algum critério de escolha é utilizado para definir "o melhor" classificador h ativado para o exemplo t; e assim determinar a classe prevista para t. Em problemas de classificação *multi-label*, o objetivo é determinar "os melhores" previsores h que classificam o exemplo t, também segundo algum critério de escolha.

Os classificadores chamados *globais* são aqueles onde, para um dado domínio de classes $\mathscr{C}$ a serem preditas, um único previsor H é construído para discriminar todas as classes de $\mathscr{C}$ (SUN; LIM; NG, 2003). No processo de previsão de classes, esta característica torna a construção de classificadores globais potencialmente mais complexa, comparada aos classificadores locais, pois em uma única etapa de treinamento, um problema qualquer de classificação é tratado em sua totalidade. Assim, no processo de previsão de classe(s), um exemplo t é submetido a apenas um classificador, tanto para problemas de classificação *single-label* quanto *multi-label*. Em classificação *single-label*, o classificador H deve ser capaz de prever para t "apenas uma" classe $c \in \mathscr{C}$. Todavia, para classificação *multi-label*, o classificador H deve ter a capacidade de prever um conjunto de classe $C \subseteq \mathscr{C}$ para o exemplo t (ALVES; DELGADO; FREITAS, 2008).

O trabalho descrito nesta tese aborda a construção de classificadores globais e locais para problemas de classificação *multi-label*.

2.5.4 CLASSIFICAÇÃO HIERÁRQUICA

No tocante à organização estrutural das classes a serem preditas, a tarefa de classificação é dividida em: não-hierárquica, usualmente chamada plana (*flat*) e hierárquica (FREITAS; CARVALHO, 2007). Nos problemas de classificação *plana* não há qualquer relacionamento estrutural entre as classes do domínio que caracterize uma organização hierárquica. Um típico problema de classificação plana *single-label* seria prever se um cliente representa ou não um potencial fraudador de cartão de crédito. Nota-se neste exemplo que, para as classes a serem preditas (potencial ou não fraudador) não há qualquer relação hierárquica. Em classificação plana *multi-label*, Boutell et al. (2004) propôs um método baseado em SVM (mais detalhes no Apêndice B.2) cujo classificador construído foi aplicado no domínio de classificação semântica de cenas, onde uma fotografia poderia pertencer a mais de uma cena (classe), por exemplo, às classes *pôr-do-sol* e *praia*. Nesse problema abordado por Boutell et al. (2004) as classes a serem preditas também não possuem qualquer relação hierárquica.

Nos problemas de classificação hieráquica, as classes a serem preditas possuem relacionamentos entre si de subclasses e superclasses, caracterizando uma disposição hierárquica (FREITAS; CARVALHO, 2007). Tipicamente, em estruturas hierárquicas classes podem apresentar a seguinte relação: $x \overset{pai\,de}{\succeq} y$. A Figura 10 ilustra um domínio formado por 6 classes a serem preditas, mais o nodo raiz, e duas relações possíveis entre as classes: "pai de" e "filho de". De acordo com esta Figura, pode-se dizer que a classe (nodo) A é "pai da" classe $A.1$. Outra relação possível seria: classe $B.2$ é "filho da" classe B.

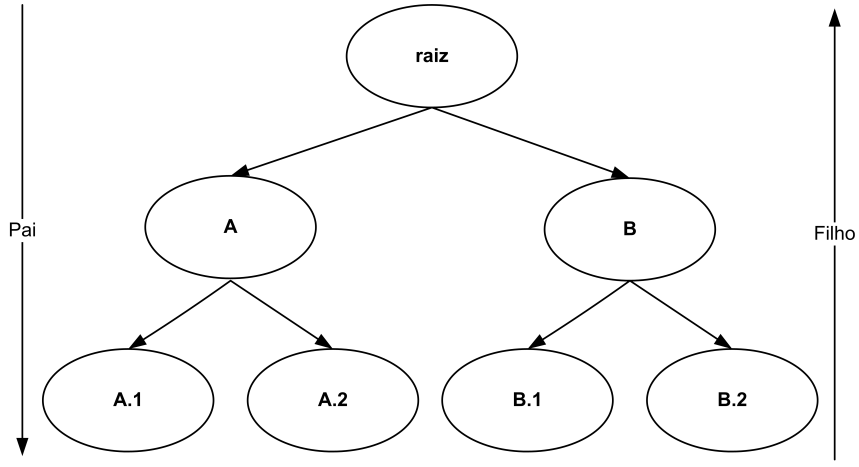


Figura 10: Exemplo de hierarquia de classes formada por 7 nodos.

Usualmente, os trabalhos descritos na literatura abordam a organização hierarárquica das classes preditas na forma de: árvore simples ou grafo acíclico direcionado (DAG – detalhes na

Seção 2.4.1) (FREITAS; CARVALHO, 2007; SUN; LIM; NG, 2003; ALVES; DELGADO; FREITAS, 2008). Em estruturas hierárquicas na forma de *árvore*, todo nodo (classe) "filho" está associado a somente um nodo "pai", exceção feita ao nodo raiz que não possui um nodo pai. Outra característica importante desta estrutura se deve ao fato dos níveis hierárquicos serem muito bem definidos, ou seja, um nodo pertence a apenas um único nível na hierarquia de classes, conforme ilustrado na Figura 11. A estrutura hierárquica na forma de árvore frequentemente utilizada em bioinformática é a Enzyme Comission (IUPAC-IUB, 1972), onde os termos definidos neste domínio descrevem funções biológicas de proteínas.

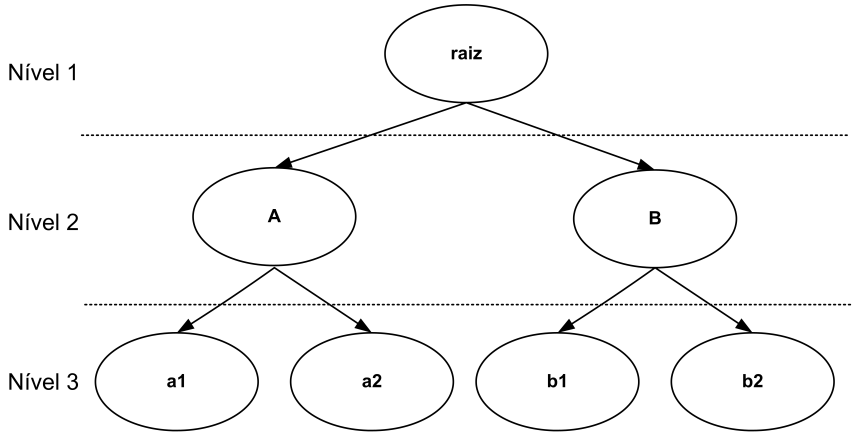


Figura 11: Estrutura na forma de um árvore.

Diferentemente das estruturas na forma de árvore, classes (nodos) organizadas em um DAG podem ter mais de um nodo "pai". Não obstante, em um DAG níveis hieráquicos não são bem definidos, pois um nodo pode pertencer a múltiplos níveis na estrutura. A Figura 12, ilustra uma organização estrutural de classes na forma de um DAG. Na Figura 12 o nodo *a2.1-b1* possui mais de um nodo pai (*a2* e *B*) e pertence a diferentes níveis dentro da estrutura (nível 4 considerando *a2* e nível 3 considerando B). Em bioinformática a estrutura hierárquica de classes na forma de um DAG mais bem conhecida é a ontologia gênica (Seção 2.4).

Problemas de classificação hierárquica são implicitamente *multi-label*, pois dentro da estrutura hierárquica, uma classe "filha" (subclasse) é uma especialização de uma classe "pai" (superclasse). Portanto, classificadores hierárquicos devem ser capazes de prever classes em toda estrutura hierárquica, tanto em níveis superiores quanto inferiores. A Figura 13 ilustra uma estrutura hierárquica na forma de um DAG, onde um exemplo t está associado à classe *a2.1-b1*. Considerando toda a estrutura hierárquica, t está implicitamente associado também as classes *A*, *a2* e *B*, pois são ancestrais de *a2.1-b1*. Portanto, qualquer classificador H deve prever como classes de t o conjunto {*A*, *B*, *a2*, *a2.1-b1*}. Em Freitas e Carvalho (2007) são descritos diversos trabalhos e técnicas de classificação hierárquica.

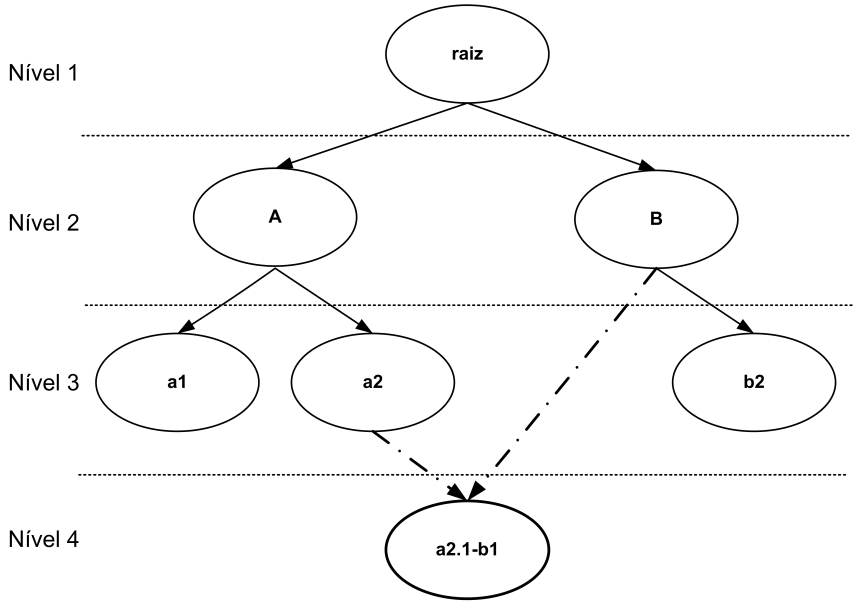


Figura 12: Estrutura na forma de DAG.

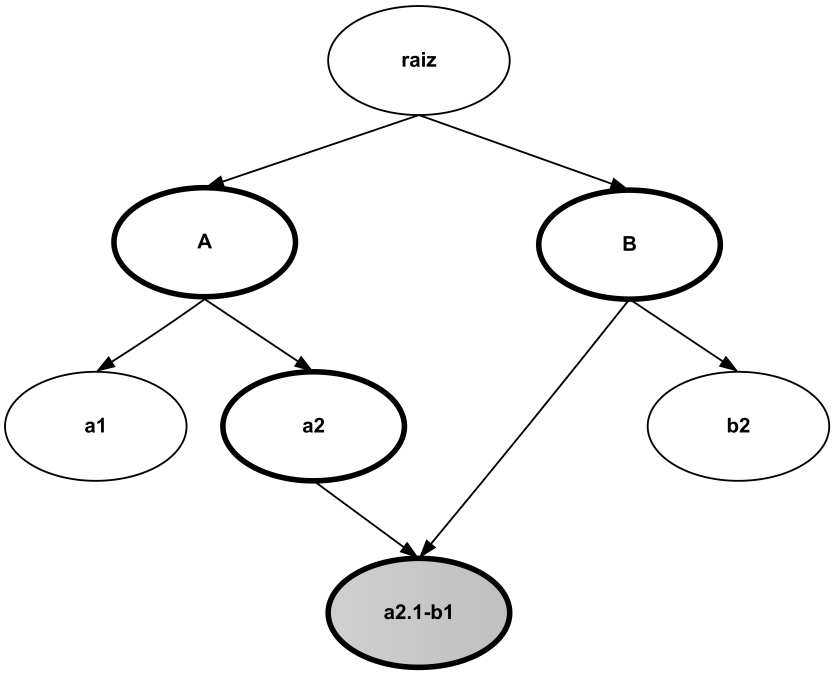


Figura 13: Complexidade de previsão em estruturas hierárquica. Seja *a2.1-b1* a classe mais específica associada a um exemplo *t*. Um classificador hierárquico deve prever todas as classes em destaque.

De acordo com os aspectos anteriormente descritos, nota-se que a construção de classificadores hierárquicos é mais desafiadora quando comparada à classificação plana. Esta tese aborda a construção de classificadores capazes de prever classes hierarquicamente organizadas em um DAG.

2.5.5 REPRESENTAÇÃO DO CONHECIMENTO EM CLASSIFICAÇÃO NA FORMA DE REGRAS SE-ENTÃO

Nesta subseção será discutida apenas a representação de regras de classificação do tipo **SE-ENTAO**, devido ao fato que esta é a representação do conhecimento adotada nesta tese, conforme será discutido posteriormente. Informações detalhadas sobre este tipo de representação e vários outros usados em mineração de dados podem ser encontradas em Witten e Frank (2005).

Regras do tipo SE-ENTÃO são muito populares em mineração de dados porque sua representação tem um alto nível de abstração, tornando-se mais compreensível para a maioria dos usuários (QUINLAN, 1993). Este tipo de representação baseia-se na divisão da regra em duas partes. A primeira é o *antecedente* da regra (parte SE) que é composta por condições envolvendo a combinação de valores de alguns atributos previsores da base de dados. A segunda, o *consequente* (parte ENTÃO) é formado por uma ou mais classes (valor(es) do atributo meta) da base de dados. Os valores do atributo meta são importantes para o usuário, uma vez que a previsão destes valores tem como objetivo melhorar o processo de tomada de decisão. A semântica deste tipo de representação é a seguinte: se as condições especificadas no antecedente de uma regra forem satisfeitas pelos atributos previsores de um exemplo, então o valor(es) previsto(s) para o atributo meta será(ão) o valor(es) especificado(s) no consequente da regra (FREITAS, 2002). A seguinte regra SE *salário* > 1000 E *idade* > 20 E *sexo* = "masculino" ENTAO *classe* = "Potencial Comprador" representa um exemplo de regra do tipo SE-ENTÃO.

2.6 SISTEMAS IMUNOLÓGICOS

O organismo animal é constantemente ameaçado pela possível invasão de vários tipos de microorganismos como bactérias, vírus e parasitas, conhecidos como patógenos (agentes causadores de doenças). A condição na qual um organismo reage a estes patógenos é chamada na área médica de *imunologia*. Assim, o *sistema imunológico* é o mecanismo dedicado principalmente a eliminar o patógeno, agindo através de um conjunto integrado de estruturas, células e elementos do sangue que permitem ao organismo reconhecer e responder a agentes agressores causadores de danos, provenientes do meio ambiente. A ação de proteger o organismo contra

estes invasores é chamada de *resposta imune* (ABBAS; LITCHMAN, 2000). Patógenos são usualmente chamados de *antígenos*.

O sistema imunológico pode ser considerado um sistema de proteção multicamadas, formado por uma barreira anatômica (pele), sistema imune inato e sistema imune adaptativo (ALBERTS et al., 2002). O *sistema imune inato* possui natureza congênita e não específica, com uma capacidade limitada de diferenciar um agente infeccioso de outro, reagindo de maneira semelhante a todas as substâncias estranhas. O *sistema imune adaptativo* é assim chamado por sua capacidade de identificar e eliminar de maneira específica um patógeno, e sua capacidade de adquirir uma memória imunológica, permitindo uma resposta imunológica mais eficiente. Quando o organismo é invadido por um patógeno, primeiramente, o sistema inato é responsável pela eliminação do invasor, caso a ação inata falhe, o sistema adaptativo assume a responsabilidade de eliminar o patógeno.

Diversos são os elementos que compõem o sistema imunológico, sendo que os principais são as células apresentadoras de antígenos (APC - *Antigen-Presenting Cell*), os linfócitos e os anticorpos. As células APC's, tais como os *macrófagos*, circulam pelo sangue ingerindo patógenos e, através da digestão, fragmentam estes invasores em peptídeos por um processo chamado *fagocitose*. Em seguida, estes peptídeos são apresentados na superfície da célula APC, onde podem ser reconhecidos por linfócitos (células T). Os dois principais tipos de linfócitos são chamados de células B e T. As *células T* reconhecem apenas peptídeos de antígenos que são fragmentados por células APC's. Este reconhecimento é realizado através de receptores que as células T possuem em sua superfície. Através deste reconhecimento, células T liberam substâncias chamadas *citocinas*, que estimulam outras células do sistema imunológico. As *células B* também possuem em sua superfície receptores (*anticorpos*); e têm a capacidade de reconhecer antígenos diretamente. Quando uma célula B reconhece um antígeno e é estimulada por citocinas liberadas por células T, inicia-se um processo de proliferação, onde cópias (clones) idênticas de si são geradas; em seguida, estas cópias são diferenciadas em células de memória e células plasmócitas que são produtoras de anticorpos solúveis de seus receptores. As células de memória permanecem no sistema imune para manter uma resposta imune mais eficiente a antígenos iguais ou semelhantes àqueles reconhecidos no passado; entretanto, quando não estimuladas após um longo período, células de memória são eliminadas (ABBAS; LITCHMAN, 2000).

2.6.1 SISTEMA IMUNOLÓGICO ARTIFICIAL (SIA)

Os Sistemas Imunológicos Artificiais (SIA) são compostos por metodologias inteligentes, inspiradas na imunologia biológica (teórica ou experimental), suas funções, modelos e princípios para solução de problemas do mundo real (DASGUPTA, 1999; DE CASTRO; TIMMIS, 2002; TARAKANOV; SKORMIN; SOKOLOVA, 2003). Diversas propriedades de sistemas imunológicos biológicos são relevantes para motivar o desenvolvimento dos SIAs, como por exemplo: reconhecimento de padrões, autonomia, tolerância a falhas, memória, aprendizagem, detecção de anomalias, elementos que atuam de maneira paralela, entre outras (DE CASTRO; TIMMIS, 2002).

Diversos são os problemas onde SIAs têm sido aplicados tais como: detecção de anomalias (DASGUPTA; FORREST, 1996; DASGUPTA, 1999), segurança computacional (KIM; BENTLEY, 2002), detecção de falhas (BRADLEY; TYRRELL, 2000), **mineração de dados** (ALVES et al., 2004b, 2004a; WATKINS; TIMMIS; BOGGESS, 2004), entre outros. Quando se deseja buscar uma solução em um domínio de aplicação, três itens são importantes na modelagem de um SIA (DE CASTRO; TIMMIS, 2002):

- *representação* das soluções candidatas para criação do modelo imune,
- *medidas de afinidades* que são usadas para quantificar as interações dos elementos do sistema; e
- *princípios imunológicos* que implementam um método de busca no espaço de soluções candidatas.

2.6.2 REPRESENTAÇÃO

A imunologia biológica descreve vários elementos que podem ser representados em um SIA. Em problemas de classificação (seção 2.5.2) é muito comum a representação de antígenos e anticorpos, onde antígenos podem representar exemplos da base de dados, enquanto que os anticorpos representam possíveis soluções que irão compor o modelo de classificação, como por exemplo, uma regra do tipo SE-ENTÃO (ALVES et al., 2004a, 2004b). Outros elementos da imunologia biológica que frequentemente são representados em SIAs são detalhados em DE CASTRO e Timmis (2002).

2.6.3 MEDIDAS DE AFINIDADE

Na imunologia biológica o nível ou qualidade de reconhecimento entre um elemento do sistema imune e o patógeno é chamado de afinidade. Assim, quanto maior o nível de afinidade, maior o nível de reconhecimento a um invasor do organismo. Em SIAs, a afinidade entre um antígeno e um anticorpo está geralmente associada à distância entre eles, que pode ser estimada por qualquer medida de distância conhecida, como por exemplo, distância de Hamming, Euclidiana, Manhattan, entre outras (DE CASTRO; TIMMIS, 2002). A decisão sobre qual medida de afinidade usar está relacionada ao domínio da aplicação (FREITAS; TIMMIS, 2003). Em Alves et al. (2004a) a medida de afinidade usada foi o grau de ativação de uma regra dado por uma norma-**t**, normalmente utilizada para agregação de atributos em sistemas nebulosos (PEDRYCZ; GOMIDE, 1998). A distância Euclidiana como medida de afinade é utilizada em Watkins, Timmis e Boggess (2004).

2.6.4 PRINCÍPIOS IMUNOLÓGICOS

Definida a representação dos elementos que irão compor o SIA e definida(s) a(s) medida(s) de afinidade que quantificará (ou quantificarão) as interações entre estes elementos, há a necessidade de estabelecer quais princípios imunológicos guiarão a busca no espaço de soluções candidatas. Os mais conhecidos são a seleção clonal e a rede imunológica que são usados nesta tese.

2.6.4.1 SELEÇÃO CLONAL

O princípio da seleção clonal descreve características de uma resposta imune a um estímulo antigênico (BURNET, 1978). Basicamente, células B quando estimuladas através do reconhecimento de antígenos, se *proliferam* gerando clones de si mesmas; em seguida estes clones são submetidos a um processo chamado de *maturação da afinidade* ou *maturação da resposta imunológica*. O princípio da maturação da afinidade declara que clones gerados são submetidos a altas taxas de mutação (*hipermutação somática*) de seus receptores. O objetivo deste mecanismo é melhorar a resposta imunológica no reconhecimento antigênico. A idéia é que a taxa de mutação à qual os clones serão submetidos seja inversamente proporcional à afinidade da célula produtora dos clones. Assim, clones de células com baixa afinidade sofrerão altas taxas de mutação e caso a afinidade não seja melhorada, estas células são eliminadas do repertório imunológico, enquanto que células com níveis altos de afinidade sofrerão baixas taxas de mutação, podendo até mesmo não sofrer mutação alguma. Finalmente, estes clones iniciam um processo de *diferenciação*, onde os clones se tornarão células plasmáticas que são produtoras de

anticorpos ou células de memória, cuja resposta imunológica é mais eficiente quando reexposta ao estímulo antigênico já reconhecido no passado. O primeiro algoritmo utilizando o princípio da seleção clonal e maturação da afinidade foi o CLONAG (DE CASTRO; VON ZUBEN, 2002), com vários outros algoritmos propostos, em seguida, utilizando estes conceitos (ALVES et al., 2004a, 2004b; YU; HOU, 2004; SU; SHYR; SU, 2005; CUTELLO et al., 2006).

2.6.4.2 REDE IMUNOLÓGICA

A teoria da rede imunológica foi proposta para explicar a capacidade de memória e aprendizado do sistema imunológico (JERNE, 1974). Esta teoria descreve que as células do sistema imunológico interagem entre si, mesmo na ausência de antígenos. A teoria sugerida por Jerne (1974) propõe que antígenos possuem em suas superfícies regiões onde anticorpos se ligam, e assim iniciam a resposta imunológica. Esta região do antígeno é chamada de *epítopo*, enquanto que a região do anticorpo que reconhece epítopos antigênicos é chamada de *paratopo*. Os anticorpos possuem em suas superfícies regiões chamadas de *idiotopos*, que também são reconhecidas por outros anticorpos. Assim, paratopos de anticorpos se ligam a epítopos de antígenos e a idiotopos de outros anticorpos. Essa interação na rede imunológica gera uma resposta *positiva* (excitatória) ou *negativa* (inibitória). A resposta positiva estimula a produção de anticorpos, enquanto que a negativa suprime anticorpos do repertório imunológico conforme ilustra a Figura 14. Assim como algoritmos baseados no princípio da seleção clonal, vários modelos de redes imunológicas foram propostos para diferentes classes de problemas (HUNT; COOKE, 1996; TIMMIS; NEAL; HUNT, 2000; NASAROUI; GONZALES; DASGUPTA, 2002; SECKER; FREITAS; TIMMIS, 2003).

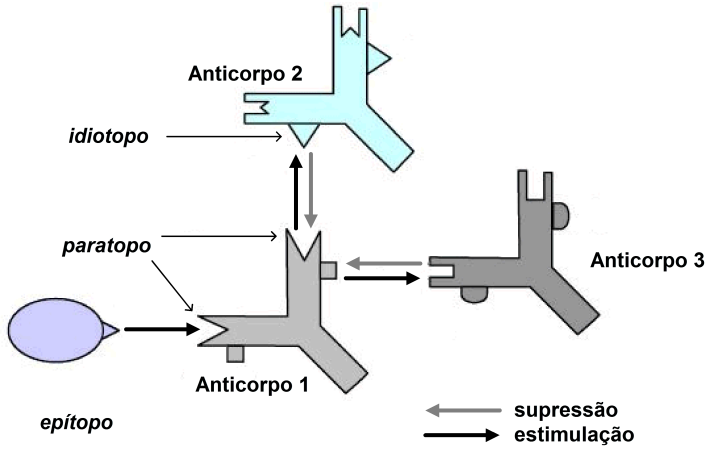


Figura 14: Esquema básico das interações em uma rede imunológica.

2.7 ABORDAGENS PARA PREDIÇÃO DE FUNÇÕES DE PROTEÍNAS BASEADA NA ONTOLOGIA GÊNICA

Muitos trabalhos têm sido desenvolvidos ao longo dos anos para predição de funções de proteínas. Entretanto, a tarefa de predição baseada em ontologia gênica é um campo de pesquisa relativamente novo, lembrando que a GO e seus termos começaram a ser estudados e divulgados na literatura a partir de 2000 (seção 2.4). Atualmente, a GO tem sido bastante usada para definição das classes de funções biológicas de proteínas. Alguns trabalhos são listados na Tabela 3 e se diferenciam em vários aspectos, tais como: o ramo da GO abordado pelo método, o método de aprendizado de máquina utilizado para predição, a maneira como é tratada a questão hierárquica da GO, entre outros. A Tabela 3 sumariza as principais características das abordagens estudadas e as informações nela contida são apresentadas da seguinte forma:

- **Método**: algoritmo de aprendizado de máquina usado para construir o modelo de predição;
- **Classificador**: esquema de treinamento utilizado para construção do classificador. No esquema *binário* (local) é construído um classificador para cada termo da GO. No esquema *global* é construído um único classificador para todas os termos considerados;
- **Ramo da GO**: ramo da GO abordado para predição, sendo Processo Biológico (PB), Função Molecular (FM) e Componente Celular (CC);
- **Hierarquia**: especifica se é considerada a questão hierárquica do DAG GO durante o treinamento do algoritmo: sim, não, ou parcial;
- **Representação do Conhecimento**: especifica o tipo de conhecimento extraído pelo classificador.

Estas 5 (cinco) características mencionadas na Tabela 3 serão discutidas em detalhes nas Seções 2.7.1 a 2.7.5

2.7.1 MÉTODOS DE CLASSIFICAÇÃO E ATRIBUTOS USADOS PARA PREDIZER FUNÇÕES DA GO

Apesar de todos os trabalhos estudados estarem voltados para predição de funções de proteínas de acordo com a GO, estes se diferenciam principalmente com relação ao método de classificação e os atributos selecionados e utilizados para classificação.

Tabela 3: Principais características das abordagens para predição de função de proteinas usando GO.

Autores	Método	Classificador	Ramo GO[a]	Hierarquia	Representação do Conhecimento
Jensen et al. (2003)	RNA	binário	PB + FM + CC	não	caixa-preta
King et al. (2003)	AD[b] + RB[c]	*global*	PB + FM + CC	parcial	AD + RB
Letovsky e Kasif (2003)	Rede de Markov	*global*	PB + FM + CC	parcial	rede de Markov
Tu et al. (2004)	RNA	*global*	PB	parcial	caixa-preta
Lu et al. (2004)	Informação Mútua + Regressão Logística	binário – *global*	PB + FM + CC	parcial	caixa-preta
Vinayagam et al. (2004)	SVM	binário – *global*	FM	sim	caixa-preta
Eisner et al. (2005)	Agrupamento (SVM + BLAST + PST)	binário	FM	sim	caixa-preta
Barutcuoglu, Schapire e Troyanskaya (2006)	SVM + RB	binário	PB	sim	RB
Vens et al. (2008)	AD	global	PB + FM + CC	sim	AD

[a]PB: Processo Biológico; FM: Função Molecular; CC: Componente Celular
[b]AD: Árvore de Decisão
[c]RB: Rede Bayesiana

Em Jensen et al. (2003) uma RNA-MLP (rede neural artificial *multi-layer perceptron*) com uma camada escondida é treinada para classificar funções de proteínas. Os 16 atributos previsores utilizados são derivados da sequência primária da proteína. Os atributos utilizados foram selecionados por um outro algoritmo, também baseado em RNA (JENSEN et al., 2002). A vantagem deste método é que atributos são selecionados automaticamente baseado em informações biológicas que permitem identificar característica mais relevantes para classificar funções.

Um método baseado em rede de Markov é apresentado em Letovsky e Kasif (2003). O objetivo é classificar os termos da GO para proteínas com base no conceito de interação entre elas. O autor assume que proteínas que interagem entre si têm maior probabilidade de desempenhar as mesmas funções biológicas. O classificador é construído sobre a informação que indica quais proteínas interagem entre si; e os termos da GO anotados para estas proteínas. Basicamente, aplica-se a regra de Bayes (Teorema B.1 - Pag. 178) e assume-se uma distribuição binomial para construção da rede de Markov. Entretanto, nem sempre proteínas que interagem entre si apresentam as mesmas funções biológicas, o que representa uma limitação deste método.

O trabalho de King et al. (2003) apresenta um algoritmo baseado em AD (Árvores de Decisão) e RB (Redes Bayesianas). O objetivo é construir um modelo de classificação assumindo que se os termos da GO **A** e **B** quaisquer ocorrem juntos frequentemente em uma base de proteínas, então proteínas anotadas apenas com o termo **A**, deveriam ser anotadas também com o termo **B**. Primeiramente, uma AD é gerada para cada termo. Cada árvore representa os termos que ocorrerem com maior frequência associados àquele termo para qual a árvore foi gerada. Cada nodo está relacionado com um valor de confiança. Para melhorar a correção preditiva do algoritmo, uma RB é treinada. Os valores da tabela de probabilidades condicionais são representados pelos valores de confiança associados a cada nodo da AD gerada. Assim como em Letovsky e Kasif (2003), nem sempre funções correlacionadas ocorrem juntas em uma mesma proteína. Além disso, no processo de predição, uma proteína deve estar associada a pelo menos um termo GO, pois os classificadores são treinados para prever termos que se relacionam com frequência em uma base de dados. Os atributos previsores são baseados no conceito de frequência de ocorrência de termos na base de dados.

O algoritmo apresentado em Lu et al. (2004) tem como objetivo associar termos da GO a proteínas através de correlação entre GO e padrões PROSITE (seção 2.3.2). Esta correlação é obtida através de informação mútua e regressão logística. Os atributos previsores são baseados em frequência, tais como: ocorrência de cada termo GO e padrão PROSITE na base de dados, frequência com que um determinado termo GO ocorre com um padrão PROSITE específico, entre outros. A partir destes atributos previsores, um algoritmo baseado na medida de informação mútua distingue se uma associação GO-PROSITE é positiva ou não. Posteriormente, o

valor de informação mútua para cada associação GO-PROSITE é utilizado como atributo previsor para um segundo algoritmo baseado em regressão logística. A limitação destes algoritmos está na necessidade de que para associar um termo GO a proteína é necessário encontrar associações GO-PROSITE e em seguida submeter estas associações ao classificador gerado, e assim obter uma resposta positiva ou negativa para aquela associação. Uma outra desvantagem se deve ao fato que, intuitivamente, termos com baixas frequências provavelmente serão classificados como associações negativas, perdendo assim, a capacidade de classificar "exceções" verdadeiras em bases da dados.

O trabalho de Tu et al. (2004) também apresenta um RNA-MLP com uma camada escondida para predição de termos GO anotados para proteínas. Entretanto, o objetivo é prever para uma proteína termos da GO mais específicos (considerando o DAG da GO), do que os termos que já se encontram anotados para a mesma. Os atributos previsores utilizados são relativos às condições experimentais de expressão gênica. O trabalho não relata quantos atributos previsores foram utilizados para a classificação de funções.

Em Vinayagam et al. (2004) um classificador SVM é treinado para associar termos da GO às proteínas. Entretanto, esta classificação é baseada em similaridades entre proteínas com o uso do algoritmo BLAST (ALTSCHUL et al., 1996). De forma geral, neste trabalho, para uma proteína p, um conjunto P de proteínas com nível de similaridade e acima de um limiar (*e-value*) é obtido com o uso do BLAST. Os termos da GO que ocorrem em p e nas proteínas de P são considerados exemplos rotulados com classe positiva (+1). Caso termos que ocorrem nas proteínas de P possuam relacionamento hierárquico com os termo da proteína p, então são considerados também exemplos positivos. Os relacionamentos considerados são: ancestrais, descendentes e irmãos. Por fim, termos que ocorrem em P e não ocorrem na proteína p são exemplos rotulados com a classe negativa (-1). Este procedimento se repete para toda proteína p da base de dados, dando origem à base de treinamento para a construção do classificador pelo algoritmo SVM. Os atributos previsores que formam um exemplo de treinamento são obtidos através dos valores de similaridades retornados pelo BLAST nas comparações e através da estrutura do DAG da GO. Estes atributos previsores são por exemplo: nível de similaridade (*e-value*) de cada proteína, porcentagem de aminoácidos idênticos na sequência, entre outras informações obtidas pelo BLAST; distância do termo GO até a raiz, número de caminhos possíveis do termo GO até raiz, estes obtidos do DAG; e baseados na frequência que um termo GO ocorre nas sequências similares, entre outros. Para classificar novos termos GO para uma proteína, deve-se submetê-la ao BLAST que encontrará sequências similares. As sequências similares terão ou não termos GO anotados. Os termos anotados representarão exemplos de classificação que serão apresentados ao classsificador, o qual retornará uma predição positiva

ou negativa para aquele exemplo. A vantagem desse classificador se deve ao fato que parte dos atributos previsores são obtidos através do algoritmo BLAST extremamente utilizado em bioinformática. Esse classificador depende também do alto nível de similaridade de proteínas que tenham termos da GO anotados.

O trabalho de Eisner et al. (2005) apresenta um conjunto de classificadores (*ensemble*) construídos por algoritmos de diferentes paradigmas para associar termos da GO a uma proteína. O objetivo principal do trabalho é explorar amplamente a estrutura hierárquica da GO. Os algoritmos que compõem o conjunto são: classificador SVM usando como atributos previsores padrões PFAM (seção 2.3.1); classificador SVM com atributos previdores extraídos pela ferramenta *Proteome Analyst* (SZAFRON et al., 2004), árvore probabilística de sufixo (PST - *Probabilistic Suffix Trees*) que é uma forma de representar *strings* como uma cadeia de Markov; e o algoritmo BLAST (ALTSCHUL et al., 1996). Resumidamente, se dois ou mais classificadores apresentarem como saída um mesmo termo GO, então este é associado à proteína classificada. Todos os classificadores treinados são binários e construídos para prever apenas um termo GO, inferindo como saída uma resposta positiva ou negativa. Várias metodologias de treinamento são aplicadas no trabalho, tais como: construção de classificadores binários *top-down*; classificar exemplos para sequências com altos níveis de similaridades encontradas pelo BLAST; classificar exemplos com baixos níveis de similaridades encontradas pelo BLAST; entre outras metodologias. Este trabalho apresenta várias vantagens, principalmente por ter sido amplamente avaliado utilizando várias metodologias de treinamento; por utilizar um conjunto de classificadores construídos por algoritmos conhecidos e seleção de atributos importantes para predição. Entretanto, para cada termo GO 4 classificadores são criados de forma independente um do outro, impedindo a integração entre eles.

Em Barutcuoglu, Schapire e Troyanskaya (2006) é desenvolvido um classificador SVM e uma RB. O objetivo é associar termos da GO a uma proteína usando uma enorme quantidade de informações biológicas para construção dos classificadores. Como atributos previsores são utilizadas informações de interação de proteínas, *microarray* de expressão gênica, localização celular e fator de transcrição de sítios de ligação. Todas estas características são normalizadas e apresentadas para o classificador SVM. Para cada termo GO um classificador binário é treinado. Como saída, o classificador retorna um valor real que é utilizado para construir uma RB para melhorar a correção preditiva e explorar a estrutura hierárquica da GO. Como vantagem, os algoritmos são treinados sobre uma variedade de informações biológicas, entretanto, algumas são irrelevantes ou desnecessárias para o treinamento do classificador.

Vens et al. (2008) apresenta um algoritmo baseado em AD para construção de classificadores globais. Esse algoritmo possui a capacidade de lidar com estruturas hierárquicas na

forma de árvores e DAGs. O desempenho desse método é observado sobre 12 bases de dados biológicos de leveduras (*Saccharomyces cerevisiae*). Diversas características são usadas como atributos previsores, como por exemplo: taxa de aminoácidos, tamanho da sequência, peso molecurar, hidrofobicidade, estruturas secundárias, matrizes de expressão genica, entre outras. As vantagens desse métodos são, principalmente, a capacidade de lidar com diferentes estruturas hieráquicas e o uso de AD que é um paradigma computacional extensivamente estudado e validado pela literatura.

Nesta tese é utilizado um SIA para construção do classificador e os atributos previsores utilizados são padrões PROSITE, *peso molecular* e *número de aminoácidos* na sequência, entre outros.

2.7.2 ESQUEMAS DE CLASSIFICAÇÃO LOCAL VS. GLOBAL

Entre os trabalhos estudados, apenas os algoritmos desenvolvidos por Jensen et al. (2003) e Eisner et al. (2005) são tipicamente classificadores binários, também chamados de *classificadores locais*. Nestes trabalhos um classificador é construído para cada termo GO, tratado como uma classe prevista. Cada classificador retorna uma resposta *positiva* (pertence à classe prevista) ou *negativa* (não pertence à classe prevista). Assim uma proteína deve ser submetida a um conjunto de classificadores para obter o conjunto de termos GO associado àquela proteína.

Os trabalhos de Lu et al. (2004), e Vinayagam et al. (2004) podem ser caracterizados como classificadores binários pelo fato de que classificam um exemplo como positivo ou negativo na predição. Os termos da GO não são considerados explicitamente como classes a serem previstas, mas sim implicitamente. Atributos para cada termo da GO são obtidos em uma etapa de pré-processamento. Assim, cada vetor de atributos obtido para um termo GO é usado como exemplo de treinamento para construir o classificador. Para associação de termos da GO a um novo exemplo, o processo é análogo, deve-se obter um conjunto de termos da GO candidatos e seus respectivos vetores de atributos. Posteriormente, cada exemplo é submetido ao classificador treinado, obtendo como resposta uma saída positiva ou negativa. Apesar destes classificadores serem binários numa primeira avaliação, sob o ponto de vista de geral predição, ele podem ser enquadrados na categoria *global*. Isto porque a predição é realizada através de um único classificador para qualquer termo GO com seu vetor de atributos.

No trabalho de Barutcuoglu, Schapire e Troyanskaya (2006) também é criado um classificador para cada termo GO previsto. Cada nova proteína é classificada com um determinado valor de confiança que especifica se uma proteína é ou não associada ao termo GO para o qual o classificador foi treinado. Estes valores são usados para construir e treinar uma rede bayesiana

para complementar a predição. Posteriormente, os valores de confiança obtidos pelos classificadores são usados para a rede bayesiana que foi treinada, e assim é obtido o conjunto de termos GO associados àquela proteína. Esse modelo tem a desvantagem de que predições podem ter respostas distintas no classificador tipicamente binário e na rede bayesiana, assim fica a dúvida sobre qual resposta deve ser assumida.

Os algoritmos desenvolvidos em King et al. (2003), Letovsky e Kasif (2003) e Vens et al. (2008) são tipicamente *globais* pois um único classificador é criado para predizer todos os termos GO de uma proteína. Em King et al. (2003) o classificador é representado por ADs e complementado por uma RB, enquanto que em Vens et al. (2008) é baseado apenas em AD. No trabalho de Letovsky e Kasif (2003) o classificador é representado uma rede de Markov. A principal vantagem do esquema de classificação *global* se deve ao fato de que, durante a construção do classificador, o algoritmo de aprendizado de máquina tem acesso a todas as classes de uma única vez. Assim, o algoritmo tem a oportunidade de construir um classificador único e executar uma análise ampla de todas as classes.

O trabalho de Tu et al. (2004) também apresenta um classificador *global*. Entretanto, como a GO é representada por um DAG, o objetivo deste trabalho é construir um classificador para realizar predição a partir de um nodo "pai" para seus descendentes. Cada nodo descendente representa um classe passível de classificação. Assim, para cada nodo "pai" um classificador é treinado para prever classes mais específicas. A principal limitação deste método se dá pelo reduzido número de classes a serem previstas pelo classificador.

Esta tese aborda a construção de classicadores globais e locais, através de SIA, para associar funções biológicas a proteínas.

2.7.3 RAMO(S) DA GO ABORDADO(S)

Na seção 2.4 é descrito que a GO é dividida em três ramos ou sub-ontologias independentes, cada qual descrevendo genes e produtos gênicos específicos. Os três ramos são: processo biológico (PB), função molecular (FM) e componente celular (CC). Os trabalhos estudados podem ser categorizados quanto ao(s) ramo(s) da GO abordado(s) para predição de funções de proteínas.

Os trabalhos de Tu et al. (2004) e Barutcuoglu, Schapire e Troyanskaya (2006) abordam o ramo da GO referente aos processos biológicos. A limitação desta abordagem se deve ao fato que este ramo da GO descreve uma série de eventos realizados por um ou mais grupos ordenados de funções moleculares. Assim, o conhecimento detalhado da função de proteína é perdido, pois tem-se uma classificação em um nível mais geral da sua atividade.

Em Eisner et al. (2005) e Vinayagam et al. (2004) o ramo da GO referente às funções moleculares é abordado. Este ramo descreve atividades realizadas por produtos gênicos no nível celular. Intuitivamente, este é o ramo mais adequado para descrição e utilização para predição de funções de proteínas.

Todos os três ramos da GO são abordados em Jensen et al. (2002), King et al. (2003), Letovsky e Kasif (2003), Lu et al. (2004) e Vens et al. (2008). Esta abordagem parece a mais completa, pois é sabido que algumas funções biológicas dependem do ambiente celular das proteínas. Assim, informações adicionais do ramo da GO de componente celular e de processo biológico agregam conhecimento para a construção de classificadores mais precisos. Entretanto, como o objetivo é realizar predições de funções biológicas, um classificador que associa com alto grau de confiabilidade apenas termos do ramo componente celular, intuitivamente, não cumpre o objetivo principal de predizer funções. Assim, é importante garantir que os termos relativos aos ramos de processo biológico e componente celular sejam tratados como informações adicionais e complementares aos termos do ramo função molecular.

No trabalho descrito nesta tese são abordados apenas os termos da GO relativos ao ramo função molecular.

2.7.4 EXPLORAÇÃO DA ESTRUTURA HIERÁRQUICA DA GO PARA CRIAÇÃO DO CONJUNTO DE TREINAMENTO

A exploração da estrutura hierárquica da GO para construção de classificadores tem sido cada vez mais frequente. Por definição, proteínas são anotadas para termos mais específicos dentro da estrutura hierárquica da GO. Também por definição, se uma proteína é anotada para o termo mais especifico go_i, implicitamente está anotada também para os termos mais gerais daquele termo (ancestrais(go_i)). Este conceito é importante principalmente para a construção das bases de treinamentos dos classificadores.

Para a construção de classificadores binários existe a necessidade da criação de uma base de treinamento de um classificador prevendo um termo go_i, onde os exemplos positivos e negativos são bem definidos. Desconsiderando a hierarquia da GO, por exemplo, se proteínas são anotadas para o termo go_i e um classificador binário é treinado para este termo, então estas proteínas são tratadas como *exemplos positivos*. As proteínas anotadas com termos diferentes de go_i são consideradas como *exemplos negativos*. O trabalho de Jensen et al. (2003) utiliza esta abordagem para treinamento de classificadores binários. Esta metodologia tem a desvantagem que proteínas associadas aos ancestrais(go_i) são incorretamente classificadas como negativas, pois não considera a estrutura hierárquica da GO.

Em Lu et al. (2004) os exemplos positivos são representados pelas associações PROSITE-go_i, PROSITE-pais(go_i), PROSITE-filhos(go_i) e PROSITE-irmãos(go_i); caso contrário são tidos como exemplos negativos. Apesar de certa forma explorar a estrutura da GO, predições positivas são obtidas apenas para termos vizinhos de go_i, pois não são considerados os termos ancestrais de go_i.

O trabalho de Vinayagam et al. (2004) aborda os vários relacionamentos da estrutura hierárquica da GO. Para a construção da base de treinamento são considerados exemplos rotulados com classes positivas: ancestrais(go_i), descendentes(go_i) e irmãos(go_i). Apesar de explorar amplamente a estrutura hierárquica da GO, relacionamentos irmãos(go_i) intuitivamente não poderiam representar exemplos positivos. Pois, em geral termos irmãos(go_i) deveriam ser considerados como exemplos negativos.

A hierarquia também é considerada no trabalho de Barutcuoglu, Schapire e Troyanskaya (2006). Para cada classificador treinado para o termo go_i, os exemplos positivos são aqueles anotados com go_i e descendentes(go_i); caso contrário são considerados exemplos negativos. Entretanto, quanto aos exemplos associados aos termos ancestrais(go_i) o trabalho não especifica se são considerados positivos, negativos ou não usados para compor a base de treinamento. Esta metodologia tem a vantagem de ao considerar termos descendentes(go_i) como uma classe positiva para go_i, o número de exemplos positivos da base de treinamento aumenta consideravelmente, fornecendo maior suporte estatístico para construção de classificadores.

A estrutura hierárquica da GO é amplamente explorada e avaliada sob vários aspectos em Eisner et al. (2005). Os classificadores binários são construídos com base na abordagem *top-down*. Assim, classificadores são construídos a partir dos termos mais gerais (próximos da raiz) para os mais específicos. Os esquemas para exploração da estrutura hierárquica, usando as mesmas nomenclaturas adotadas pelo autor, são: *exclusivo*, *menos-exclusivo*, *inclusivo* e *menos-inclusivo*. Todos estes esquemas são detalhados na Tabela 4. O desempenho dos classificadores com o esquema inclusivo foi superior comparado aos outros esquemas, isto se deve em virtude da base possuir mais exemplos positivos e eliminação de ruídos representados por proteínas ambiguamente anotadas para termos ancestrais(go_i).

Em algoritmos *globais* esta distinção entre exemplos negativos e positivos não é muito bem definida; e é normalmente tratada dinamicamente durante o treinamento do classificador. Assim há diversas formas de abordar a questão hierárquica da GO. No trabalho de Tu et al. (2004) a hierarquia é tratada apenas no sentido de que um classificador é treinado para predizer exclusivamente os descendentes(go_i). A distinção entre exemplos positivos e negativos é realizada por um vetor binário. Cada proteína está associada a um vetor de tamanho n. Cada posição

Tabela 4: Esquemas de bases de treinamento em Eisner et al. (2005).

Esquema	Exemplos Positivos	Exemplos Negativos	Não Usados
Exclusivo	go_i	$\neg go_i$	
Menos-Exclusivo	go_i	$\neg[go_i$ + descendentes$(go_i)]$	descendentes(go_i)
Inclusivo	go_i + descendentes(go_i)	$\neg[go_i$ + descendentes$(go_i)]$	ancestrais(go_i)
Menos-Inclusivo	go_i + descendentes(go_i)	$\neg[go_i$ + descendentes(go_i) + ancestrais$(go_i)]$	

deste vetor possui o valor "1" para a proteína anotada para um termo GO; e "0" caso contrário. Uma RNA é treinada com *n* neurônios na camada de saída. O vetor de classes de cada proteína é comparado com o vetor de saída da RNA para avaliar o desempenho da classificação do algoritmo. Assim, um exemplo pode simultaneamente ser considerado como negativo e positivo durante o treinamento do classificador. A limitação desta abordagem está restrita ao fato que termos ancestrais(go_i) seriam classificados como negativos em todas as *n* saídas da RNA treinada. Entretanto, é importante ressaltar que nesta situação a limitação do algoritmo se deve ao fato da abordagem tratar de forma parcial a estrutura hierárquica da GO, e não pela classificação negativa, pois a RNA foi treinada para predizer apenas classes descendentes(go_i).

No trabalho King et al. (2003) onde é construída uma AD e uma RB para classificação, a hierarquia é considerada parcialmente. Para geração da AD a hierarquia não é considerada de forma alguma. Entretanto, para construção da RB são considerados apenas os termos ancestrais(go_i). Essa mesma abordagem para tratar a hierarquia é utilizada em Letovsky e Kasif (2003) para construção de uma rede de Markov. A hierarquia, em ambos os trabalhos, é representada por um vetor binário para cada proteína. Este vetor possui o mesmo tamanho para todas as proteínas e cada posição deste vetor contém o valor "1" se a proteína é anotada um correspondente termo *go*; e "0" caso contrário.

Em Vens et al. (2008) exemplos pré-classificados são transformados em vetores binários de classes, contendo o valor "1" se a proteína está associada à uma classe; e "0" caso contrário. Além das classes anotadas, são especificadas com o valor "1" classes ancestrais de um termo *go*.

Nesta tese os exemplos positivos e negativos são tratados dinamicamente durante o treinamento do classificador global. Na construção dos classificadores locais, são considerados como exemplos positivos termos anotados e descendentes; caso contrário são considerados exemplos negativos (de acordo com a Tabela 4 – menos-inclusivo).

2.7.5 REPRESENTAÇÃO DO CONHECIMENTO

A representação do conhecimento em uma forma compreensível ao usuário é uma das características desejáveis em algoritmos de mineração de dados (FAYYAD et al., 1996; FREITAS, 2002) e bioinformática (SZAFRON et al., 2003). A importância de conhecimento interpretável em bioinformática se deve ao fato de que, em muitos casos, informações compreensíveis auxiliam biólogos na realização de experimentos para comprovação de hipóteses.

Os algoritmos apresentados em Jensen et al. (2003), Tu et al. (2004), Lu et al. (2004), Vinayagam et al. (2004), Eisner et al. (2005) foram desenvolvidos apenas para responder positivo ou negativo para uma classificação. Esses algoritmos são considerados "caixa-preta" (*black-box*) pois não fornecem informações que justifiquem a resposta do classificador. Assim, o usuário se vê obrigado a tomar decisões baseadas apenas em um valor, ou invés de avaliar o conhecimento extraído.

Em Barutcuoglu, Schapire e Troyanskaya (2006) o conhecimento é representado parcialmente. Primeiramente, classificadores SVMs são treinados. Estes classificadores não geram conhecimento. Entretanto, combinando as respostas dos vários classificadores treinados é construída uma RB. Na rede de classificação gerada é possível visualizar o relacionamento entre os termos da GO que compõem a rede. Entretanto, a rede não fornece conhecimento a respeito da influência dos atributos previsores sobre o modelo de classificação. Similarmente, os algoritmos apresentados em King et al. (2003) apresentam as mesmas limitações, pois constroem ADs e RBs que permitem apenas a visualização de relacionamento entre os termos da GO. A diferença entre Barutcuoglu, Schapire e Troyanskaya (2006) e King et al. (2003) se deve ao fato de que o primeiro utiliza uma variedade de atributos previsores para construção dos classificadores, situação que não ocorre no segundo.

Uma rede de Markov é utilizada em Letovsky e Kasif (2003) para representar uma rede de interações entre proteínas. Nesta rede cada nodo representa uma proteína, onde é possível visualizar os termos GO anotados. O modelo é limitado pelo fato que é construído apenas baseado no conceito de interação de proteínas. Entretanto, é adequado para o propósito para o qual foi desenvolvido.

O conhecimento extraído pelo método apresentado em Vens et al. (2008) é representado na forma de AD. Dependendo da base de dados, as árvores construídas são complexas e de díficil interpretação. Todavia, esse método possui como parâmetro, a opção do usuário transformar a árvore construída em um conjunto de regras. A limitação desse metodo se deve ao fato que árvores complexas resultam em regras, também complexas, com muitas condições no antecedente.

O conhecimento é representado neste trabalho na forma de regras SE-ENTÃO com o objetivo de extrair conhecimento útil e interpretável ao usuário, permitindo análise e uso em processos de tomada de decisão.

3 CLASSIFICAÇÃO HIERÁRQUICA E *MULTI-LABEL* COM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS

Neste capítulo serão apresentadas e discutidas em detalhes as abordagens baseadas em Sistemas Imunológicos Artificiais (SIA) para classificação hierárquica e *multi-label*. A metodologia proposta foi desenvolvida para ser aplicada no campo da bioinformática, mais especificamente em predição de funções biológicas de proteínas. O objetivo principal é construir um classificador, no qual o conhecimento adquirido durante o treinamento seja expresso na forma de regras do tipo SE-ENTÃO e as classes a serem preditas sejam termos da ontologia gênica (GO - *Gene Ontology*). Classificadores globais e locais são descritos e posteriormente avaliados com o objetivo de verificar a real capacidade dos SIA para resolver problemas de classificação hierárquica e *multi-label*.

3.1 VISÃO GERAL DO ALGORITMO

De maneira geral, algoritmos baseados em SIA são formados por anticorpos e antígenos. Em problemas de classificação, normalmente os anticorpos são soluções ou possíveis soluções para um dado problema; enquanto os antígenos são instâncias (casos ou exemplos) do mesmo problema. O problema abordado neste trabalho é a predição de funções biológicas de proteínas. Desta maneira, os antígenos representam proteínas da base de dados, enquanto que os anticorpos representam regras do tipo SE-ENTÃO.

O modelo proposto neste trabalho é chamado de MHCAIS (*Multi-Label Hierarchical Classification with an Artificial Immune System*), cujos princípios imunológicos são inspirados na Seleção Clonal e Rede Imunológica, descritos nas Subseções 2.6.4.1 e 2.6.4.2, respectivamente. O objetivo é construir, através de um SIA, um classificador hierárquico e *multi-label* que possa descobrir conhecimento compreensível representado na forma de regras do tipo SE-ENTÃO. Duas versões do MHCAIS são apresentadas: global e local. Na versão local, para cada classe do domínio um classificador é construído; enquanto que, na versão global um único classificador é gerado para discriminar todas as classes do domínio da aplicação tratada (Seção 2.5.3). Em geral, nos algoritmos que descobrem regras, o consequente (parte ENTÃO) apresenta ape-

nas um valor possível para o atributo meta (classe predita). No MHCAIS local, o consequente das regras descobertas apenas discrimina se um exemplo (instância) pode ou não ser associado à classe para a qual aquele classificador foi treinado. Por outro lado, no MHCAIS global uma ou mais classes são representadas no consequente.

O Algoritmo MHCAIS possui dois procedimentos fundamentais, um para Extração Sequencial de Regras (*Sequential Covering*) e outro para Evolução das Regras (*Rule Evolution*). O primeiro procedimento é usual em algoritmos de indução de regras (WITTEN; FRANK, 2005) para classificação *single-label* e portanto sofre adaptações nesta tese para classificação hierarárquica e *multi-label*. O segundo é específico para algoritmos evolutivos ou relacionados, incluindo os SIAs baseados no princípio de seleção clonal.

O procedimento de Extração de Regras inicia com um conjunto de regras vazio, onde o mecanismo de Evolução de regras é iterativamente executado. No procedimento de Evolução de Regras, os anticorpos participam de um processo iterativo (evolução), cujo objetivo é encontrar as melhores regras (anticorpos) que formarão a solução (classificador) para o problema. Durante este processo, cada anticorpo é continuamente avaliado de acordo com sua capacidade de reconhecimento (classificação) dos antígenos da base de treinamento. Os melhores anticorpos são selecionados e adicionados ao conjunto de regras descobertas. Em seguida, os exemplos corretamente classificados pelas regras descobertas na iteração corrente são removidos. A versão global remove exemplos da base de dados de forma diferente da versão local, esta diferença é apresentada nas Subseções 3.2.13 e 3.3.4. A construção do classificador termina quando o MHCAIS descobre o número de regras necessárias para classificar os exemplos da base de dados. Este processo é chamado em mineração de dados de *etapa de treinamento* (Seções 3.2 – versão global e 3.3 – versão local).

Após a etapa de treinamento, o conjunto de regras descobertas (classificador) é avaliado sobre uma base de dados de teste. Os exemplos que compõem a base de teste não participam da etapa de treinamento. Esta etapa de avaliação é chamada em mineração de dados de *etapa teste* (Seção 3.4).

3.1.1 ANTÍGENO

Uma base de dados é tratada como um conjunto (população) de antígenos, onde cada exemplo tipifica um antígeno. A base de dados é formada por exemplos pré-classificados, de tal maneira que cada exemplo é composto por duas partes. A primeira parte representa os valores possíveis para seus respectivos atributos previsores. A segunda parte representa um conjunto de classes (valores dos atributos meta) associadas ao exemplo considerado. No contexto do

problema de predição de função de proteínas tratado neste trabalho, os atributos previsores são características da proteína que, acredita-se serem capazes de associar uma função biológica à mesma; e as classes são termos da GO que, de acordo com suas especificações, definem estas funções. Formalizando, um antígeno é representado por uma tupla $ag_i = \langle \mathbf{x}_i, L_i \rangle$, onde i representa o i-ésimo ag da base de dados.

Os atributos previsores são representados por um vetor de tamanho $|\mathscr{D}|$ na forma:

$$\mathbf{x}_i = \langle x_{i1}, x_{i2}, \cdots, x_{id}, \cdots, x_{i|\mathscr{D}|} \rangle; \tag{1}$$

onde x_{id} representa o valor do d-ésimo atributo previsor do i-ésimo antígeno. Os atributos previsores de um domínio $\mathscr{D}$ podem ser *categóricos* (discretos) ou contínuos. Os atributos categóricos são definidos da seguinte forma:

$$x_{id} \in \mathrm{Alf}_d = \{a_1^d, a_2^d, \cdots a_n^d\}; \tag{2}$$

onde n é o n-ésimo valor possível para o atributo $d \subseteq \mathscr{D}$. O atributo d pode ser representado por um universo de valores simbólicos (por exemplo, *estado civil*) não ordenados ou uma quantidade finita de valores inteiros (ordenados). Embora matematicamente valores inteiros sejam discretos e não continuos, na prática em mineração de dados valores inteiros são tratados como categóricos (discretos) apenas se o atributo em questão tiver um pequeno número de valores inteiros. Se um atributo tiver muitos valores inteiros, ele é tipicamente tratado como atributo "contínuo", conforme a seguir. Para um universo de valores simbólicos $d \subseteq \mathscr{S}$ e para valores inteiros $d \subset \mathbb{Z}$, onde $\mathscr{S}$ representa um conjunto qualquer de valores simbólicos

No caso dos atributos contínuos $x_{id} \in [l_d^{inf}, l_d^{sup}] \subset \mathbb{R}$, onde l_d^{inf} e l_d^{sup} representam os limites inferior e superior do intervalo de valores possíveis do d-ésimo atributo.

O conjunto de classes associadas a um exemplo é representado por:

$$L_i = \{l_{i1}, l_{i2}, \cdots, l_{ik}, \cdots, l_{i|L_i|}\}, 1 \leq k \leq |L_i|; \tag{3}$$

onde l_{ik} é a k-ésima classe associada ao exemplo ag_i e $L_i \subseteq \mathscr{C}$. O domínio $\mathscr{C}$ é formado pelas classes a serem preditas pelos classificadores.

O conjunto de antígenos representado na base de treinamento é formalizado por $AG = \{ag_i\}_{i=1}^{|\mathbf{S}|}$, onde $\mathbf{S}$ é a base de treinamento.

A Figura 15 ilustra um exemplo de antígeno (proteína) da base de dados. Nesta figura, as

posições 1 e 2 de $\mathbf{x}_i$ representam, respectivamente, os atributos previsores padrão PROSITE A e PROSITE B e indicam se o padrão ocorre ou não na proteína naquela posição específica. Portanto, de acordo com a Figura 15, o valor da posição 1 indica que o padrão A ocorre na proteína, enquanto o valor da posição 2 indica que o padrão B não ocorre. A posição $|\mathscr{D}|$ pode representar, por exemplo, o atributo contínuo *peso molecular*, assumindo o valor 30580. No que diz respeito aos atributos meta L_i, a proteína assume os termos 166, 5524 e 17111, onde cada valor representa o identificador do termo na GO.

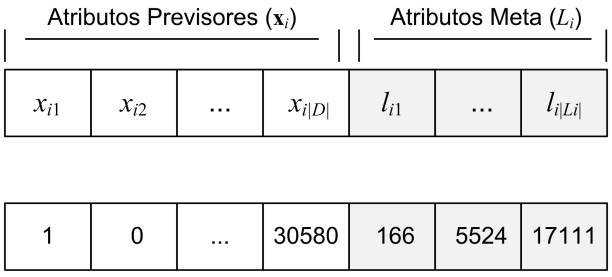


Figura 15: Exemplo de um antígeno da base de dados representando uma proteína.

3.1.2 ANTICORPO

O anticorpo representa uma regra SE-ENTÃO. A parte SE, chamada também de antecedente, é composta por um conjunto de condições. Cada condição está associada a um atributo previsor do domínio da aplicação. O MHCAIS possui a capacidade de lidar com atributos contínuos e categóricos. A parte ENTÃO, também chamada de consequente, é formada pelas classes preditas pela regra. As duas versões do MHCAIS representam o consequente da regra ligeiramente diferentes entre si. Na versão local, o anticorpo é representado por uma tupla $ab_j = \langle \mathbf{z}_j, y_j \rangle$; enquanto que na versão global, anticorpos estão na forma $ab_j = \langle \mathbf{z}_j, Y_j \rangle$, onde j indica o j-ésimo anticorpo da população. O antecedente é formado por uma conjunção (E) de condições e é definido por:

$$\mathbf{z}_j = \langle z_{j1}, z_{j2}, \cdots, z_{jd}, \cdots, z_{j|\mathscr{D}|} \rangle, \text{para } 1 \leq d \leq |\mathscr{D}|; \tag{4}$$

onde d é a d-ésima condição codificada em ab_j e cada d está associada a um atributo previsor do domínio $\mathscr{D}$. Toda condição é composta por três (tripla) elementos, $z_{jd} = \langle OP_d^j, V_d^j, B_d^j \rangle$, onde cada elemento representa:

- um operador lógico OP_d que pode ser: $(=)$ ou $(\neq)$ para atributos categóricos; ou $(\geq)$ ou $(<)$ para atributos contínuos;

- um valor válido $V_d \in \text{Alf}_d$ ou $V_d \in [l_d^{inf}, l_d^{sup}]$, onde Alf_d representa um alfabeto de valores do atributo $d \in \mathscr{D}$; e

- um marcador B_d (0 ou 1) que indica se a condição será ou não utilizada para classificar exemplos. Este recurso tem como finalidade a desativação de condições irrelevantes no antecedente e portanto é desconsiderada no cálculo da afinidade ou *matching* da regra (Subseção 3.2.4).

O segundo elemento da tupla que forma ab_j representa o consequente da regra. No MHC-AIS local o consequente é representado na seguinte forma:

$$y_j^l = \begin{cases} 1 & \text{se a regra prevê a classe } l \\ 0 & \text{caso contrário} \end{cases} ; \tag{5}$$

onde l representa a classe para qual o classificador local é treinado para prever.

Diferentemente, o consequente no MHCAIS global é formado por um conjunto de valores dos atributos meta:

$$Y_j = \{y_{j1}, y_{j2}, \cdots, y_{jq}, \cdots, y_{jm}\}, \text{para } Y_j \subseteq \mathscr{C}; \tag{6}$$

onde $\mathscr{C}$ é o domínio de classes a serem preditas e m é o número de classes representadas no consequente da regra. O valor de m é definido na Subseção 3.2.2 e é igual para todo anticorpo.

No processo de classificação *multi-label* desenvolvido neste trabalho a(s) classe(s) do consequente da regra representa(m) termo(s) da Ontologia Gênica (GO).

O conjunto de anticorpos é representado por $AB = \{ab_j\}_{j=1}^{tam}$, onde tam é o número de anticorpos da população. O tamanho da população de anticorpos não é constante durante a execução do procedimento de evolução de regras, conforme será descrito na Subseção 3.2.9.

A Figura 16 ilustra o exemplo de um anticorpo (regra) codificado na versão global. Neste exemplo, considera-se que as posições 1, 2 e $|\mathscr{D}|$ do antecedente $\mathbf{z}_j$ representam, respectivamente, aos atributos previsores PROSITE A, PROSITE B e *Peso Molecular*. Os atributos 1 e 2 são categóricos e assumem valores 0 ou 1 para indicar se os padrões PROSITE ocorrem ou não na proteína, e como ambos possuem o marcador $B = 1$ são considerados ativos. O atributo $|\mathscr{D}|$ é contínuo e representa um condição desativada de regra, pois $B = 0$. O consequente Y_j da regra prevê (cobre) os termos da GO, respectivamente, 166, 5524 e 17111. Este anticorpo decodificado representaria a seguinte regra:

SE (PROSITEA = 1) E (PROSITEB = 0) ENTÃO {166, 5524, 17111}

ou seja, se ocorre o padrão PROSITE A e não ocorre o padrão B, então a regra prevê os termos 166, 5524 e 17111. A condição associada ao atributo *peso molecular* não é representada na regra decodificada, pois o terceiro elemento (B^j) possui o valor 0 indicando que a mesma é irrelevante para a regra.

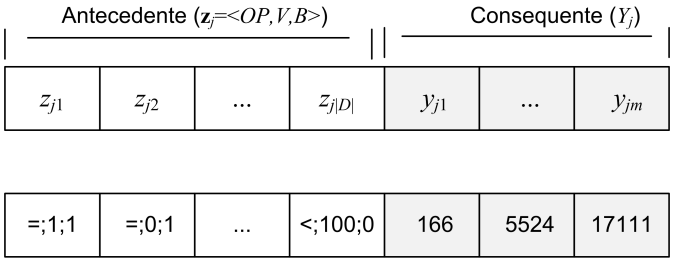


Figura 16: Exemplo de um anticorpo codificado no MHCAIS-global.

3.2 DESCRIÇÃO DO MHCAIS-GLOBAL

Nesta seção são detalhados e descritos os procedimentos utilizados pelo MHCAIS-global para construção de classificadores. Conforme mencionado em seções anteriores, o MHCAIS constrói um único classificador para discriminar (prever) todas as classes do domínio. As Figuras 17 e 18 ilustram o funcionamento dos procedimentos de Extração e Evolução de regras usados nesta versão do MHCAIS.

Inicialmente, o MHCAIS recebe como entrada uma base de dados **S** (proteínas) e a transforma em uma população de antígenos – *AG* – (Subseção 3.2.1). Em seguida, cada antígeno da base (proteína) é rerotulado para considerar a estrutura hierárquica de classes a serem previstas – AG' – (Subseção 3.2.2). As regras que compõem o classificador (*CR*) são descobertas, de forma iterativa, pelo procedimento de Evolução de Regras através de um SIA. No início do período iterativo ($t = 0$), o classificador (conjunto de regras) é dado por $CR = \emptyset$, pois nenhuma regra foi ainda descoberta. A cada execução t, o SIA descobre uma regra, que será adicionada em *CR*. Esta regra representa a melhor regra (*MelhorRegra*) encontrada pelo SIA naquela iteração t. Em seguida, exemplos cujas classes são todas cobertas por (*CR*), são retirados da base de dados (Subseção 3.2.13). Enquanto a condição de repetição $|AG'| \geq \#MaxExemBD$ for verdadeira, este ciclo é continuamente executado. O valor do parâmetro *#MaxExemBD* tem como objetivo limitar o número de exemplos que não são cobertos por nenhuma regra descoberta, e que serão então classificados por uma regra padrão (ALVES et al., 2004a, 2004b; ALVES; DELGADO;

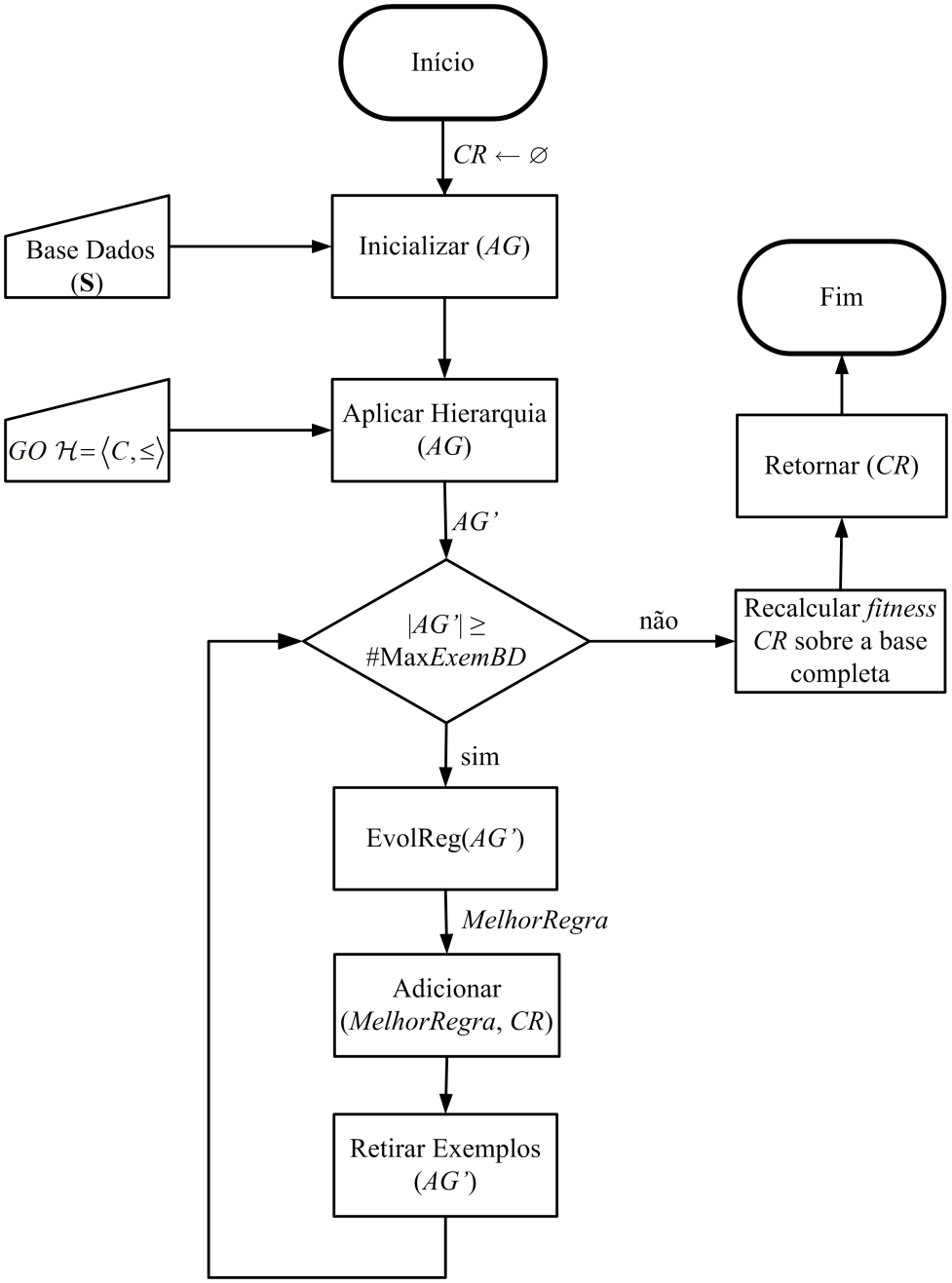


Figura 17: Fluxograma geral de funcionamento do procedimento de Extração de Regras para o MHCAIS-global.

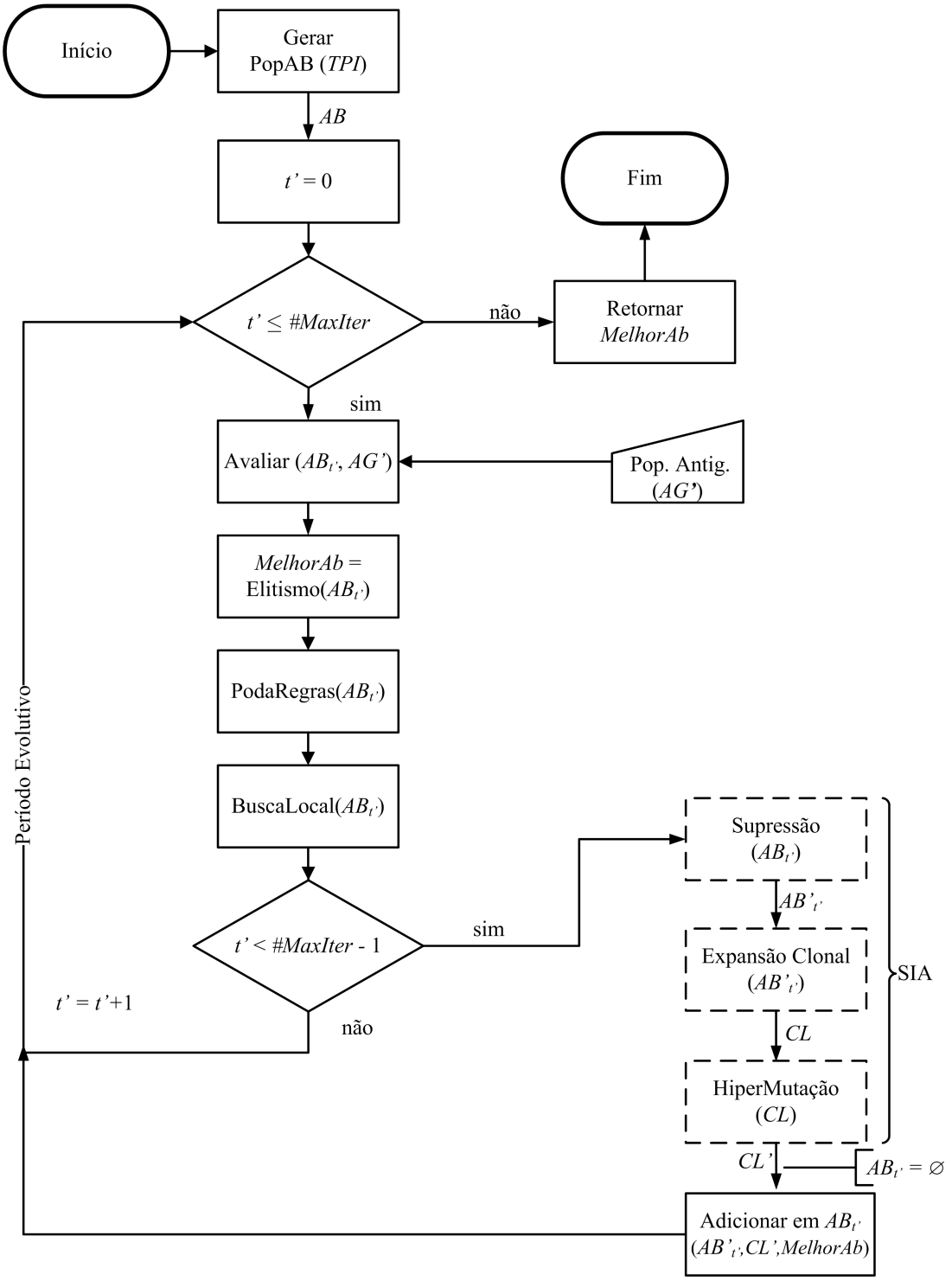


Figura 18: Fluxograma geral de funcionamento do procedimento de Evolução das Regras do MHCAIS-global. Retângulos com linhas não-contínuas representam procedimentos específicos de SIA.

FREITAS, 2008). O conceito de regra padrão será apresentado Seção 3.4. Em seguida, a qualidade (*fitness*) de todas as regras descobertas é recalculada sobre a base de treinamento completa. Finalmente, o conjunto de regras descobertas *CR* será submetido a uma avaliação na etapa de teste (Seção 3.4).

O procedimento de Evolução de regras, executado por um SIA, tem como objetivo descobrir regras. Neste procedimento, inicialmente, são geradas TPI (tamanho da população inicial) regras (anticorpos) de forma aleatória. Durante um período, mecanismos característicos dos SIAs e mineração de dados atuam sobre estas regras no intuito de melhorar as possíveis soluções do problema. Este ciclo é chamado de período evolutivo. Ao fim de cada período evolutivo, a melhor regra é retornada para o procedimento de Extração de Regras e são executados tantos períodos evolutivos quantos necessários para construção do conjunto de regras.

Todos os mecanismos utilizados pelo MHCAIS serão descritos nas subseções seguintes.

3.2.1 INICIALIZAÇÃO DA POPULAÇÃO DE ANTÍGENOS

Na inicialização, o procedimento recebe como entrada a base de treinamento de proteínas. Em seguida, os exemplos da base de treinamento **S** são atribuídos ao conjunto de antígenos AG, onde cada exemplo agora passa a representar um antígeno na seguinte forma:

$$AG = \{ag_i\}_{i=1}^{|AG|}, ag_i = \langle \mathbf{x}_i, L_i \rangle, |AG| = |\mathbf{S}|; \tag{7}$$

3.2.2 APLICAÇÃO DA HIERARQUIA DE CLASSES DA GO SOBRE AG

A estrutura hierárquica $\mathscr{H}$ dos termos (classes) da GO também é fornecida como entrada para o algoritmo. Conforme descrito na Seção 2.4.1 a estrutura da GO é definida da seguinte maneira: $\mathscr{H} = \langle \mathscr{C}, \preceq \rangle$, onde $\mathscr{C}$ representa o domínio de termos definidos e a relação $\preceq$ determina a estrutura hierárquica do grafo na forma de um conjunto parcialmente ordenado de termos.

Conforme já discutido, todo termo $l_{ik} \in L_i$ do i-ésimo antígeno representa um nodo no DAG $\mathscr{H}$. Assim, o MHCAIS explora a estrutura hierárquica de todos os elementos do conjunto L_i associado ao antígeno ag_i. Ao explorar o DAG $\mathscr{H}$, os termos ancestrais de $l_{ik} \in L_i$ passam a ser representados também em ag_i. Esta representação é definida da seguinte forma:

$$AG' = \{ag'_i = \langle \mathbf{x}_i, \mathbf{l}'_i \rangle \mid \exists ag_i \in AG\}_{i=1}^{|AG'|}, |AG'| = |AG|; \tag{8}$$

onde $\mathbf{l}'_i$ representa um vetor binário de termos da GO associados e não-associados à proteína ag'_i. O vetor $\mathbf{l}'_i$ é obtido após a exploração hierárquica dos elementos do conjunto L_i. O vetor $\mathbf{l}'_i$ representa linearmente o subgrafo do termos da GO que ocorrem em AG mais seus ancestrais, onde cada componente deste vetor assume o valor 1 para os termos l_{ik} e Ancestrais(l_{ik}) associados a ag_i, caso contrário assume o valor 0. O tamanho m de $\mathbf{l}'_i$ é igual para todo $ag_i \in AG$, onde $m = |T_H|$ e o conjunto T_H sendo obtido da seguinte forma:

$$T_H = T \bigcup T_A; \tag{9}$$

com:

$$T_A = \bigcup \text{Ancestrais}(l) - \text{NodoRaiz}(\mathscr{H}), \forall l \in T; \tag{10}$$

$$T = \bigcup L_i, \forall ag_i \in AG; \tag{11}$$

onde $T \in \mathscr{C}$ representa o conjunto de termos (classes) que ocorrem em AG e T_A é o conjunto de termos ancestrais de cada elemento de T obtidos através da exploração hierárquica $\mathscr{H}$. O nodo raiz é excluído de T_A, pois todo l está ligado hierarquicamente à raiz da estrutura.

A Figura 19 ilustra graficamente o conjunto T_H obtido de uma base qualquer de proteínas definido da seguinte forma, seja:

- O conjunto $T = \{17016, 3677, 3723, 4518, 16740\}$ obtido através da Equação 11, onde cada nodo está ilustrado na cor preta na figura;
- o conjunto $T_A = \{5488, 3824, 166, 3676, 16787, 16788\}$ é obtido pela Equação 10, nodos ilustrados na cor cinza da mesma figura; e
- por fim, como resultado da exploração de $\mathscr{H}$, através da Equação 9 se obtém o conjunto $T_H = \{5488, 3824, 166, 3676, 16787, 16740, 17016, 3677, 3723, 16788, 4518\}$.

Para o exemplo ilustrado na Figura 19, tem-se um vetor $\mathbf{l}'$ com o número de componentes $m = 11 = |T_H|$ para cada proteína ag' da população e q é o q-ésimo elemento deste vetor $q =$

$1, \ldots, m$.

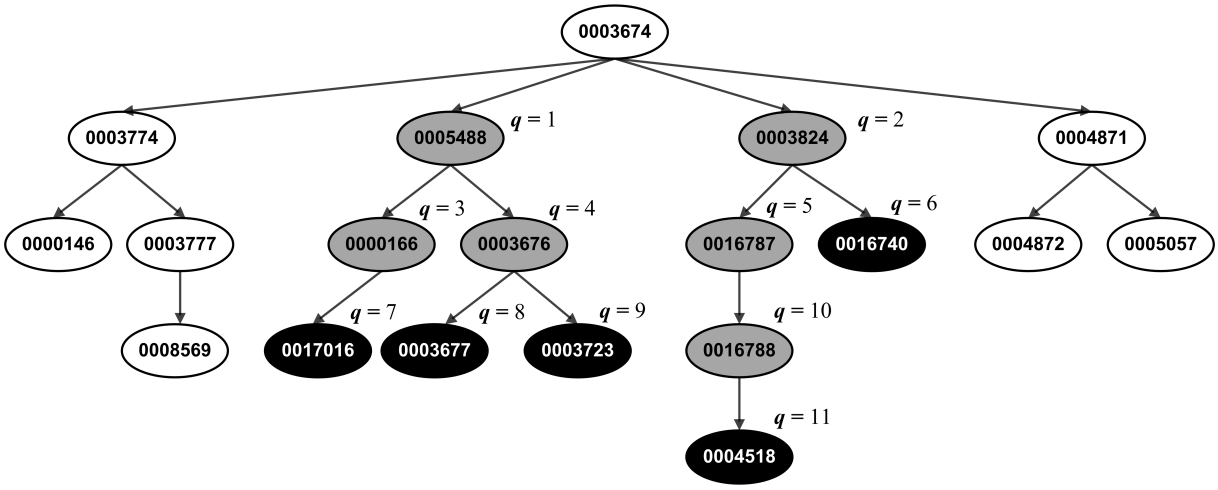


Figura 19: Exemplo de um sub-grafo resultante da exploração hierárquica dos termos da GO contidos em uma base de proteínas. Os nodos na cor preta representam os distintos termos l associados diretamente às proteínas da base, e os nodos na cor cinza são resultados da exploração da hierarquia destes termos no DAG $\mathscr{H}$. O índice q representa o q-ésimo elemento de $\mathbf{l}'_i$.

Portanto, o valor de m e o conjunto T_H especificam, respectivamente, o número de componentes e quais serão os elementos representados em $\mathbf{l}'$.

Em seguida, o algoritmo deve explorar novamente o DAG $\mathscr{H}$ para atribuir os valores para cada posição q de $\mathbf{l}'_i$. Inicialmente, a estrutura $\mathscr{H}$ é explorada para obter os termos ancestrais de todos os elementos de L_i do i-ésimo antígeno ag_i. Este passo é dado por:

$$T_{A_i} = \bigcup \text{Ancestrais}(l_{ik}) - \text{NodoRaiz}(\mathscr{H}), \forall l_{ik} \in L_i; \tag{12}$$

Conforme discutido anteriormente, o nodo raiz também não é representado em T_{A_i}. Em seguida, o conjunto resultante da exploração de $\mathscr{H}$ é definido por:

$$T_{H_i} = L_i \bigcup T_{A_i}, \text{ onde } T_{H_i} \subseteq T_H; \tag{13}$$

O conjunto T_{H_i} representa as classes associadas a ag_i considerando a estrutura hierárquica $\mathscr{H}$. Por fim, os valores de vetor $\mathbf{l}'_i$ são obtidos da seguinte forma:

$$\mathbf{l}'_i[q] = \begin{cases} 1 & \text{se } l_q \in T_{H_i} \\ 0 & \text{caso contrário} \end{cases}; \tag{14}$$

onde q representa o q-ésimo elemento de T_H.

Para exemplificar tal procedimento, seja:

- $L_i = \{17016, 4518\}$ os termos GO (classes) associados a um ag_i;
- T, T_H e m obtidos do exemplo anterior apresentado na Figura 19;
- $T_{A_i} = \{166, 5488, 16788, 16787, 3824\}$, obtido a partir da aplicação da Equação 12 sobre L_i;
- $T_{H_i} = \{17016, 4518, 166, 5488, 16788, 16787, 3824\}$ de acordo com a Equação 13; e
- o vetor de classes de ag_i seria $\mathbf{l}'_i = [1_1, 1_2, 1_3, 0_4, 1_5, 0_6, 1_7, 0_8, 0_9, 1_{10}, 1_{11}]$, após a aplicação da Equação 14.

A Figura 20 ilustra graficamente os componentes do exemplo apresentado, onde L_i é representado pelos nodos em cor preta e T_{A_i} os nodos em cor cinza. Os nodos em cor branca representam as classes que não estão associadas a ag_i.

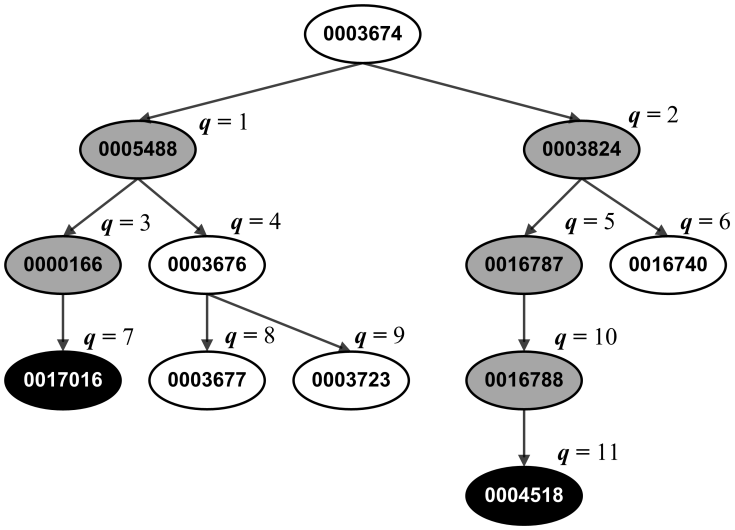


Figura 20: Aplicação da estrutura hierárquica sobre as classes de um exemplo ag_i, onde $L_i = \{17016, 4518\}$ (nodos em cor preta). Percorrendo o DAG $\mathscr{H}$ obtém-se o conjunto de termos ancestrais $T_{A_i} = \{166, 5488, 16788, 16787, 3824\}$ (nodos em cor cinza), onde $T_{A_i} = \{\bigcup \textbf{Ancestrais}(l_{ik}), \forall l_{ik} \in L_i\}$. O índice q representa o q-ésimo elemento de T_H.

3.2.3 CRIAÇÃO DA POPULAÇÃO INICIAL DE ANTICORPOS

Na geração da população inicial de anticorpos, as condições que formam o antecedente de $ab_j \in AB$ são geradas de forma aleatória. O número de termos representados no consequente é

igual a m (procedimento descrito na Seção 3.2.2). Portanto, inicialmente cada ab_j tem representado em seu consequente Y_j todos os termos $l \in T_H$. O tamanho da população inicial AB (TPI) é determinado pelo usuário.

3.2.4 CÁLCULO DA AFINIDADE (*MATCHING*)

A afinidade entre um ab_j e ag'_i, no contexto de mineração de dados, determina se os valores dos atributos previsores $\mathbf{x}_i$ de ag'_i satisfazem as condições ativas ($B^j_d = 1$) no antecedente $\mathbf{z}_j$ da regra ab_j (Equação 4). Este procedimento é geralmente chamado em mineração de dados de *matching*, e pode ser parcial ou total. Em um *matching* total, todos os valores dos atributos previsores devem satisfazer todas as condições ativas no antecedente da regras. Em um *matching* parcial, o número de valores dos atributos previsores satisfazendo as condições ativas no antecedente da regra deve estar acima de um limiar δ_{AF}. O MHCAIS está adaptado para lidar tanto com *matching* parcial, quanto *matching* total. O cálculo da afinidade (*matching*) é obtido pela seguinte equação:

$$\text{Afinidade}(ab_j, ag'_i) = \frac{\#CondSat^i_j}{\sum\limits_{\forall d \in \mathscr{D}} B^j_d}; \tag{15}$$

onde $\#CondSat^i_j$ indica a quantidade de condições ativas em ab_j que foram satisfeitas pelos atributos previsores de ag'_i. Deve-se ressaltar que o valor da afinidade determina se um exemplo é ou não classificado pelo antecedente de um regra. Portanto, um exemplo é classificado se $\text{Afinidade}(ab_j, ag'_i) \geq \delta_{AF}$ e não-classificado, caso contrário. O valor do limiar $\delta_{AF} \in [0,1]^{\Re}$ é especificado pelo usuário. Para que o MHCAIS considere *matching* total basta especificar o valor de $\delta_{AF} = 1$.

3.2.5 DEFINIÇÃO DAS FUNÇÕES DE *FITNESS*

A função de avaliação ou *fitness* é usada para determinar a qualidade de cada $ab_j \in AB$. Durante a evolução das regras, toda as classes $l \in T_H$ (Seção 3.2.2) são representadas no consequente Y_j de ab_j. É importante ressaltar que, T_H é resultado da exploração hierárquica dos termos l que ocorrem em AG, portanto, de forma implícita esta hierarquia é representada também no consequente Y_j.

O valor de *fitness* é calculado para cada termo (classe) $y_{jq} \in Y_j$ e é obtido através do uso de matriz de confusão, onde são contabilizados os erros e acertos de predição associados ao termo y_{jq} (WITTEN; FRANK, 2005). Desta forma, todo termo $y_{jq} \in Y_j$ possui uma matriz de

confusão $MatConf_{y_{jq}}$. A Tabela 5 ilustra a matriz de confusão para um termo y_{jq} onde:

- $VP_{y_{jq}}$ (Verdadeiro Positivo): número de exemplos com Afinidade(ab_j,ag'_i) $\geq \delta_{AF}$ e $\mathbf{l}'_i[q] = 1$;
- $VN_{y_{jq}}$ (Verdadeiro Negativo): número de exemplos com Afinidade(ab_j,ag'_i) $< \delta_{AF}$ e $\mathbf{l}'_i[q] = 0$;
- $FP_{y_{jq}}$ (Falso Positivo): número de exemplos com Afinidade(ab_j,ag'_i) $\geq \delta_{AF}$ e $\mathbf{l}'_i[q] = 0$; e
- $FN_{y_{jq}}$ (Falso Negativo): número de exemplos com Afinidade(ab_j,ag'_i) $< \delta_{AF}$ e $\mathbf{l}'_i[q] = 1$.

Tabela 5: Matriz de confusão para uma classe $y_{jq} \in Y_j$ em um anticorpo ab_j.

		Classes Reais	
		y_{jq}	$\neg y_{jq}$
Classes Preditas	y_{jq}	$VP_{y_{jq}}$	$FP_{y_{jq}}$
	$\neg y_{jq}$	$FN_{y_{jq}}$	$VN_{y_{jq}}$

Para ilustrar a construção e atualização de uma matriz de confusão, seja o ag_i apresentado como exemplo na Seção 3.2.2 e ilustrado na Figura 20, onde:

- o conjunto de termos anotados para $L_i = \{17016, 4518\}$;
- o conjunto de termos ancestrais de ab_i é dado por $T_{A_i} = \{166, 5488, 16788, 16787, 3824\}$, de acordo com a Equação 12;
- $T_{H_i} = \{5488, 3824, 166, 16787, 17016, 16788, 4518\}$, obtido pela Equação 13;
- $m = 11$ indicando a dimensão do vetor binário de termos $\mathbf{l}'_i$, onde $m = |T_H|$. A construção de T_H é detalhada na Seção 3.2.2;
- $\mathbf{l}'_i = [1_1, 1_2, 1_3, 0_4, 1_5, 0_6, 1_7, 0_8, 0_9, 1_{10}, 1_{11}]$, de acordo com a Equação 14; e
- Seja ag'_i o antígeno ag_i após a exploração da estrutura hierárquica do DAG $\mathscr{H}$.

Considerando que os valores dos atributos previsores $\mathbf{x}_i$ do antígeno ag'_i tenham sido cobertos pelo antecedente $\mathbf{z}_1$ da regra ab_1, portanto Afinidade(ab_1,ag'_i) $\geq \delta_{AF}$. As classes previstas pela regra ab_1 são as mesmas ilustradas na Figura 19 pelos nodos em cor cinza e preto. Neste cenário, o exemplo seria contabilizado nas matrizes de confusão de y_{1q} para a regra codificada em ab_1 da seguinte forma:

- VP em $MatConf_{y_{1q}}$ para os termos $y_{1q} = \{5488, 3824, 166, 16787, 17016, 16788, 4518\}$; e
- FP em $MatConf_{y_{1q}}$ para os termos $y_{1q} = \{3676, 16740, 3677, 3723\}$.

Considere agora que, $\mathbf{x}_i$ da mesma proteína ag'_i não tenha sido coberto pelo antecedente $\mathbf{z}_2$ da uma segunda regra ab_2, portanto Afinidade(ab_2,ag'_i) $< \delta_{AF}$. Neste outro cenário, o exemplo seria contabilizado nas matrizes de confusão de y_{2q} de ab_2 da seguinte forma:

- FN em $MatConf_{y_{2q}}$ para os termos $y_{2q} = \{5488, 3824, 166, 16787, 17016, 16788, 4518\}$; e
- VN em $MatConf_{y_{2q}}$ para para os termos $y_{2q} = \{3676, 16740, 3677, 3723\}$.

Após serem contabilizados erros e acertos de todo ag_i em toda $MatConf_{y_{jq}}$ de $Y_j \in ab_j$, o valor de *fitness* de cada termo y_{jq} é calculado. Nas próximas subseções serão descritas as funções de *fitness* utilizadas nesta tese.

3.2.5.1 FUNÇÃO *FITNESS* COM BASE NA MEDIDA SENSIBILIDADE VS. ESPECIFICIDADE

A função Sensibilidade vs. Especificidade é frequentemente utilizada em algoritmos de classifição (BOJARCZUK; LOPES; FREITAS, 2004; ALVES et al., 2004a, 2004b) sendo definida para cada classe pela seguinte equação:

$$\text{FitY}(y_{jq}) = \frac{VP_{y_{jq}}}{VP_{y_{jq}} + FN_{y_{jq}}} \times \frac{VN_{y_{jq}}}{VN_{y_{jq}} + FP_{y_{jq}}}; \tag{16}$$

O primeiro termo da Equação 16 representa a *sensibilidade* e o segundo a *especificidade*. A sensibilidade determina a porcentagem (cobertura) das classes positivas preditas, enquanto que a especificidade especifica a porcentagem das classes negativas corretamente não preditas.

3.2.5.2 FUNÇÃO *FITNESS* COM BASE NA MEDIDA *F-MEASURE*

A função *f-measure* também tem tido amplamente usada na tarefa de classificação (VAN RIJSBERGEN, 1979; KIRITCHENKO, 2005; EISNER et al., 2005) sendo definida para cada classe representada na regra da seguinte forma:

$$\text{FitY}(y_{jq}) = \frac{(\beta_{FT}^2 + 1) \times P \times R}{\beta_{FT}^2 \times P + R}, \beta_{FT} \in [0,\infty]; \tag{17}$$

$$P = \frac{VP_{y_{jq}}}{VP_{y_{jq}} + FP_{y_{jq}}}; \tag{18}$$

$$R = \frac{VP_{y_{jq}}}{VP_{y_{jq}} + FN_{y_{jq}}}; \tag{19}$$

Os termos P e R da Equação 17 são chamados de *precision* e *recall*, respectivamente. O termo *precision* representa o percentual de previsões corretas; enquanto que, *recall* determina o percentual (cobertura) de exemplos positivos corretamente classificados. Note que o *recall* equivale ao termo sensibilidade da Equação 16. A função *f-measure* equivale a uma média harmônica, onde através do valor do parâmetro β_{FT} atribui-se maior peso para P ou R. Para $\beta_{FT} = 1$ ambos os termos P e R têm o mesmo peso na Equação 17. Se $\beta_{FT} < 1$, *precision* tem maior peso; de forma análoga, se $\beta_{FT} > 1$ *recall* tem peso maior sobre *precision*. A função $\text{FitY}(y_{jq})$ determina a qualidade (confiabilidade) de predição da regra ab_j para o termo y_{jq}.

3.2.5.3 CONSISTÊNCIA HIERÁQUICA DA FUNÇÃO DE *FITNESS*

Neste algoritmo a consistência hierárquica das classes preditas é considerada durante o treinamento. Este procedimento evita que ao prever uma classe y_{jq}, classes ancestrais deixem de ser preditas.

Neste sentido, para manter a consistência hierárquica dos termos representados em Y_j; caso o *fitness* de algum termo $y_{jq'} \in \text{Ancestrais}(y_{jq})$ seja menor que o *fitness* de y_{jq}, então o *fitness* de y_{jq} é replicado para $y_{jq'}$. Assim:

$$\text{FitY}(y_{jq'}) = \max[\text{FitY}(y_{jq}),\text{FitY}(y_{jq'})], \forall y_{jq'} \in \text{Ancestrais}(y_{jq}); \tag{20}$$

Esta consistência é tratada, indepente da função de *fitness* utilizada, devido ao fato que termos ancestrais não preditos pelo classificador são considerados como um erro de predição durante a avaliação de desempenho do mesmo .

3.2.5.4 CÁLCULO DO *FITNESS* DE UMA REGRA

Após o cálculo do *fitness* de cada classe y_{jq}, o *fitness* global de ab_j é calculado de acordo com a seguinte equação:

$$\text{Fitness}(ab_j) = \frac{1}{nt}\sum_{q}\text{FitY}(y_{jq}) \geq \delta_{FT}; \tag{21}$$

onde *nt* especifica o número de termos y_{jq}, cujo valor de *fitness* $\mathrm{FitY}(y_{jq}) > \delta_{FT}$. O valor $\mathrm{Fitness}(ab_j)$ representa o *fitness* médio dos termos y_{jq} acima do limiar $\delta_{FT} \in [0,1]^{\Re}$. O parâmetro δ_{FT} foi introduzido para evitar a subestimação do *fitness* das regras ab_j, devido o elevado número de classes do domínio de classes a serem preditas em problemas de classificação hierárquica e *multi-label*.

3.2.6 PODA DAS REGRAS

Esta técnica consiste em remover de possíveis soluções do problema partes irrevelantes tornando-as inativas (MINGERS, 1989; ISHIBUCHI; NAKASHIMA, 1999). O objetivo neste procedimento é tornar as soluções mais simples ou compreensíveis possível, sem comprometer sua qualidade. Em muitos casos, a aplicação deste procedimento melhora a qualidade das soluções, bem como reduz um ajuste demasiado (*overfitting*)[19] das regras sobre os dados de treinamento. Em mineraçao de dados, esta técnica é chamada de "poda" e tem sido frequentemente utilizada em algoritmos que constróem árvores de decisão (QUINLAN, 1993), removendo nodos da estrutura hierárquica; e em algoritmos que constróem regras (FREITAS, 2002), removendo condições irrelevantes para a regra. É oportuno lembrar que, neste trabalho o classificador construído é formado por regras, portanto o mecanismo de poda tem como objetivo remover das regras condições irrelevantes. Conforme já mencionado, o procedimento de poda tem como característica tornar as soluções mais simples, motivando assim, seu uso no trabalho proposto que tem como um de seus objetivos disponilizar para o usuário conhecimento compreensível para auxiliá-lo no seu processo de tomada de decisão.

O esquema de funcionamento do mecanismo utilizado neste trabalho é apresentado na Figura 21. De forma geral, seleciona-se da população $AB_{t'}$ o anticorpo de melhor *fitness* para remoção de condições irrelevantes. É importante lembrar que o elemento B_d^j da tripla de ab_j (Equação 4) determina se a d-ésima condição está ativa ou não na j-ésima regra. A condição está ativa na regra ab_j se $B_d^j = 0$, e inativa se $B_d^j = 1$. O anticorpo selecionado deverá ter pelo menos 2 condições ativas, haja visto não fazer sentido remover condições de regras com apenas 1 condição ativa. O número de condições ativas para qualquer ab_j é definido pela seguinte Equação:

$$\#CondAtiv_j = \sum_{\forall d \in \mathscr{D}} B_d^j; \tag{22}$$

[19] *Overfitting* ocorre, em geral, em bases de treinamento com poucos exemplos ou ruídos. Um classificador com *overfitting* sobre os dados de treinamento tende a apresentar baixa correção preditiva sobre os dados de teste (MITCHELL, 1997).

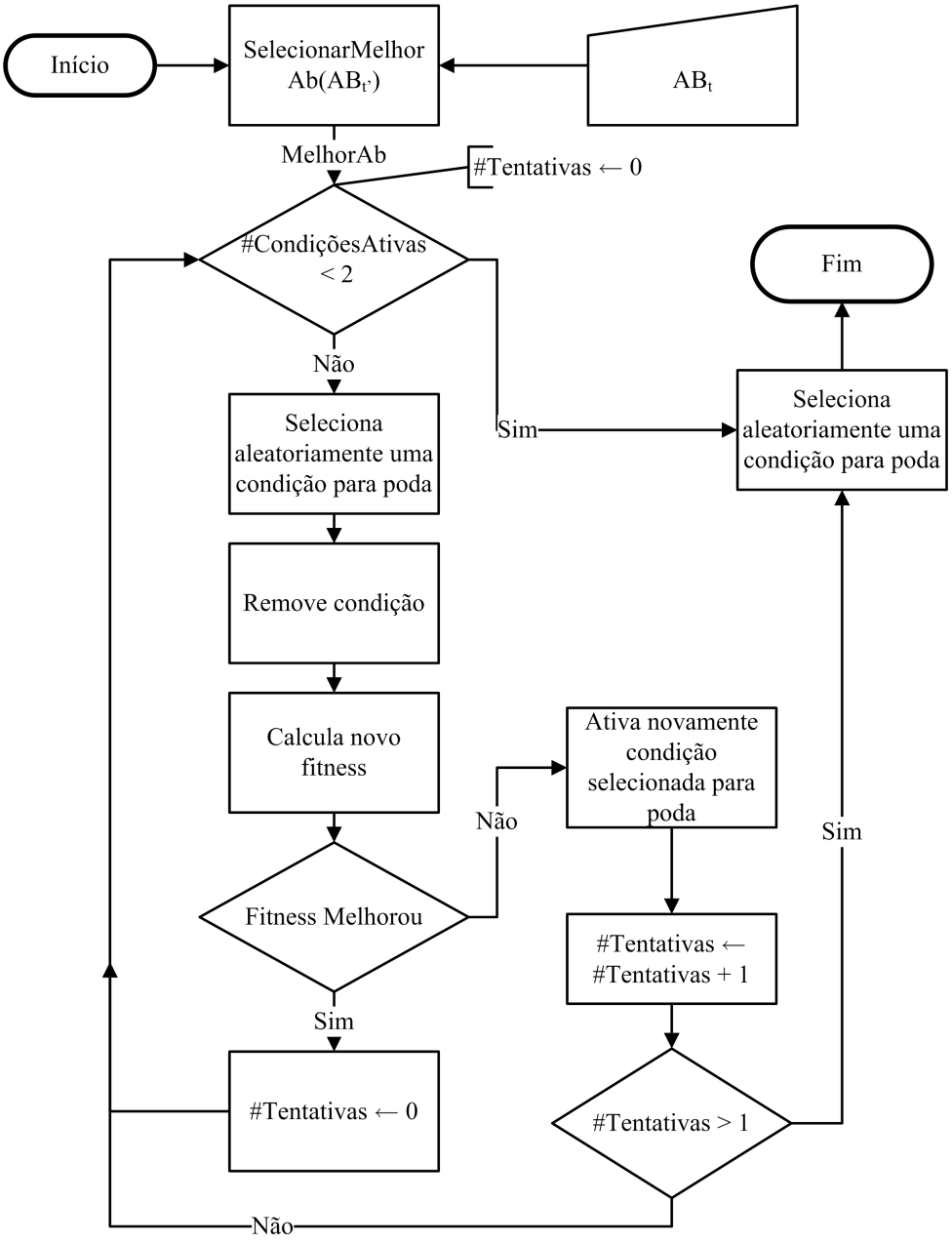


Figura 21: Fluxograma do procedimento de poda de condições irrelevantes.

Satisfeita a restrição de número de condições ativas, seleciona-se aleatoriamente uma condição ativa de ab_j para poda. O *fitness* da regra é recalculado; se o novo valor de *fitness* for maior ou igual ao *fitness* anterior, então a condição é efetivamente removida especificando assim $B_d^j = 0$. O ciclo de escolha de condições ativas para poda se repete enquanto o processo de remoção não comprometer (baixar) o valor de *fitness* de ab_j ou $\#CondAtiv_j > 1$.

Em caso de piora do valor de *fitness*, outra condição ativa é selecionada (aleatoriamente) para poda. Se o *fitness* não melhorar após duas (valor definido para evitar desperdício computacional) escolhas consecutivas de condições ativas o processo de poda é finalizado. É importante ressaltar que uma condição é removida apenas se o *fitness* da regra não piorar. Uma condição desativada só poderá tornar-se novamente ativa através do mecanismo de hipermutação somática que será descrito na Subseção 3.2.10.

3.2.7 BUSCA LOCAL

A busca local utilizada neste trabalho tem como objetivo geral realizar um "ajuste fino" no antecedente da regra para melhorar seu *fitness*. Este ajuste permite que uma solução representando um ótimo local (comum em buscas heurísticas) se torne, eventualmente, um ótimo global, através de pequenas perturbações ou mudanças de estado (HOOS; STüTZLE, 2005). Seja tomado como exemplo uma regra formada por n condições, onde uma das condições é representada por $idade \geq 18$. Eventualmente, modificar esta condição através de uma busca local, para $idade \geq 20$ poderia melhorar a qualidade (*fitness*) da regra. Deste modo, pequenas mudanças nos valores das condições de uma regra podem resultar na melhora de seu *fitness*.

O procedimento de poda descrito na seção anterior realiza um tipo específico de busca local, pois ao remover condições do antecedente simplifica a regra que passa agora representar uma possível solução em um espaço de busca reduzido (apenas para esta regra). Diferentemente, a busca local proposta neste trabalho executa pequenas mudanças (alterações) de valores, escolhidos de forma aleatória, nas condições ativas que formam o antecedente da regra.

Este trabalho apresenta uma busca local semelhante ao algoritmo de busca de subida de encosta padrão (*hill climbing*). Na técnica de subida de encosta, sucessivas mudanças de estado são realizadas em um espaço de busca, enquanto melhorar a qualidade da solução, sem armazenar em uma "memória" os estados anteriomente visitados (RUSSELL; NORVIG, 2003).

O fluxograma apresentado na Figura 22 ilustra o esquema geral de funcionamento da busca local realizada. Inicialmente, seleciona-se o melhor anticorpo *ab* da população corrente. Posteriormente, uma condição ativa de *ab* é escolhida, de forma aleatória, para que a busca local

seja realizada sobre esta condição. Em seguida, é selecionado um valor possível, no universo do atributo d que compõe a condição, também aleatoriamente. Este novo valor substituirá o valor anterior da condição escolhida. O *fitness* de ab é recalculado; enquanto o valor de *fitness* melhorar, novos valores possíveis do atributo d são escolhidos de acordo com o passo anteriormente descrito. Caso, o valor de *fitness* da regra não melhore após duas mudanças consecutivas no valor da condição, o processo de busca local para esta condição é finalizado e uma nova condição é selecionada para exploração do seu espaço de busca. Se após duas condições escolhidas, de forma consecutiva, o valor de *fitness* não apresentar melhora a busca local sobre ab é finalizada. O valor 2 (dois) utilizado tanto para o máximo de condições a serem alteradas como para o máximo de tentativas de melhora no *fitness* foi definido como base no equilíbrio (melhoria vs. custo computacional). Isto porque valores maiores, embora pudessem permitir melhores ganhos em termos de correção preditiva, trariam um custo computacional maior.

Conforme mencionado anteriormente, o procedimento proposto é baseado na técnica de subida de encosta padrão (*hill climbing*) em dois aspectos. No primeiro, por mudar valores nas condições do antecedente enquanto o *fitness* estiver melhorando. O segundo aspecto a ser considerado, deve-se ao fato da técnica proposta não armazenar, em uma espécie de memória, valores ou condições selecionados em passos anteriores. A motivação para a escolha deste procedimento de busca local em detrimento a outro se deve ao fato da simplicidade de implementação e custo computacional.

3.2.8 SUPRESSÃO

O mecanismo de Supressão é responsável por manter a diversidade da população AB que representa uma rede imunológica. Esta diversidade é alcançada excluindo da rede anticorpos similares. A similaridade entre anticorpos é calculada da seguinte forma:

$$\text{Similaridade}(ab_j, ab_{j'}) = \frac{\#CondIg}{\#MaxCondAt_\alpha}, \text{para } ab_j \neq ab_{j'}; \tag{23}$$

para:

$$\#MaxCondAt_\alpha = \max\left[\sum_{\forall d \in \mathscr{D}} B_d^j, \sum_{\forall d \in \mathscr{D}} B_d^{j'}\right]; \tag{24}$$

onde $\#MaxCondAt_\alpha$ indica a maior quantidade de condições ativas entre ab_j e $ab_{j'}$ e $\#CondIg$ representa o número de condições ativas iguais entre ab_j e $ab_{j'}$.

Caso $\text{Similaridade}(ab_j, ab_{j'}) \geq \delta_{SIM} \in [0,1]^{\Re}$, então ab_j ou $ab_{j'}$ deve ser suprimido da

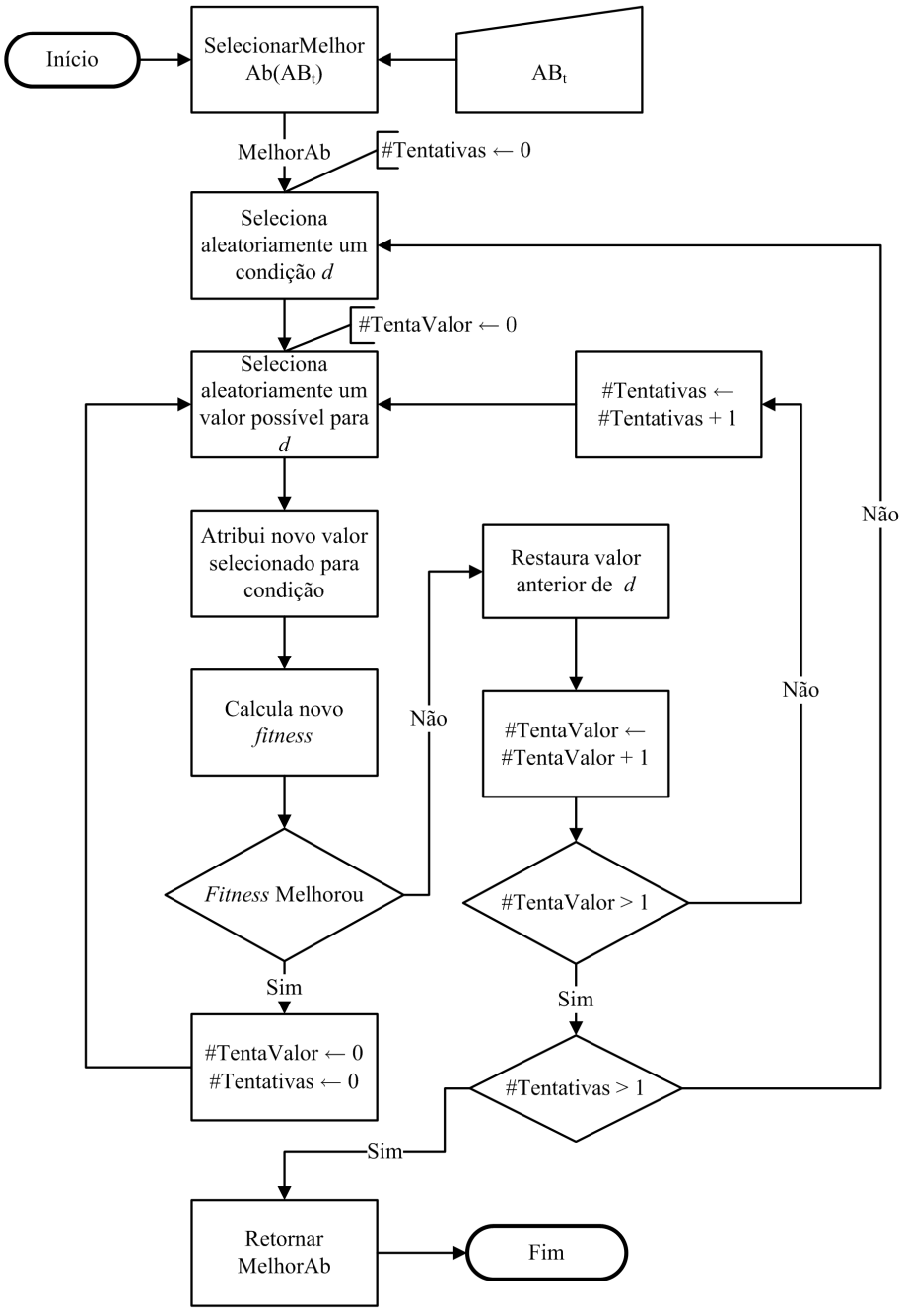


Figura 22: Fluxograma do procedimento de Busca Local.

população, onde δ_{SIM} representa um limiar para a similaridade entre anticorpos. Esta escolha é baseada no *fitness*, excluindo aquele *ab* com menor fitness.

3.2.9 EXPANSÃO CLONAL

O procedimento de Expansão Clonal, baseado no princípio da Seleção Clonal (Seção 2.6.1), tem como objetivo gerar clones de ab_j. Portanto, $\forall ab_j \in AB$ é gerada uma quantidade de clones, onde $cl \equiv ab$. A quantidade de clones de ab_j a ser gerada é diretamente proporcional ao seu *fitness*. O cálculo do número de clones é obtido por:

$$\#Cl_j = \quad \text{Inteiro}(\text{Fitness}(ab_j) \times \#MaxCl \times ClTx) \mid \#Cl_j \geq 1; \tag{25}$$

onde o valor do parâmetro $\#MaxCl \in \mathbb{N}^*$ representa o número máximo de clones que pode ser gerado de ab_j e $ClTx$ é uma variável cujo valor é calculado pelo algoritmo a cada iteração com o objetivo de controlar o tamanho da população AB, estimulando ou não a geração de clones. O valor de $ClTx$ é calculado da seguinte maneira:

$$ClTx = \begin{cases} HiperClTx & \text{se } |AB| < TPI \\ 0 & \text{se } |AB| > TPMax \\ 1 - \left(\dfrac{|AB| - TPI}{TPMax - TPI} \right) & \text{caso contrário} \end{cases}; \tag{26}$$

onde $HiperClTx$, TPI e $TPMax$ indicam respectivamente, taxa de hiperexpansão clonal, tamanho da população inicial e tamanho máximo da população de anticorpos. Os valores destes parâmetros são especificados no início da execução do algoritmo. É importante ressaltar que o parâmetro $TPMax$ não representa o tamanho máximo que a população AB pode assumir durante a evolução, todavia indica que, caso o tamanho de AB ultrapasse o valor daquele parâmetro, a geração de clones proporcional ao *fitness* é desestimulada.

3.2.10 HIPERMUTAÇÃO SOMÁTICA

Após o procedimento de Expansão Clonal, todo $cl \in CL$ é submetido ao procedimento de Hipermutação Somática, onde CL representa a população de clones gerados. A taxa de mutação a qual cl deve ser submetido é inversamente proporcional ao *fitness* do *ab* a partir do qual foi gerado. A taxa de mutação é determinada pela seguinte equação:

$$TxMutacao_{cl} = MutMin + (MutMax - MutMin) \times (1 - \text{Fitness}(cl)); \tag{27}$$

onde os parâmetros *MutMin* e *MutMax* indicam, respectivamente, a taxa de mutação mínima e máxima que um *cl* pode ser submetido; a função Fitness(cl) é definida na Equação 21.

A taxa de mutação representa a probabilidade com que cada condição no antecedente do clone *cl* sofre mutação em sua estrutura. O procedimento da mutação é aplicado sobre cada um dos 3 elementos de cada condição: valor, operador relacional e marcador que indica se a condição está ou não ativa na regra (Seção 3.1.2). Os três elementos da condição têm a mesma probabilidade de sofrerem mutação. Todo clone *cl* que sofre mutação é inserido no conjunto CL'. Ao final, CL' é adicionado na população $AB_{t'+1}$ de anticorpos.

3.2.11 ELITISMO

O procedimento de Elitismo adotado neste trabalho é semelhante aos utilizados em algoritmos evolutivos (GOLDBERG, 1989), onde o objetivo é preservar para a população $AB_{t'+1}$ aquele ab_j com o maior valor de *fitness* (Equação 28) da população corrente. O uso deste mecanismo possibilita garantir que o ab_j melhor avaliado entre todas as iterações seja preservado ao final do período evolutivo.

$$\text{Elistimo}(AB) = \arg \max_{ab \in AB} \text{Fitness}(ab); \tag{28}$$

3.2.12 ATUALIZAÇÃO DA POPULAÇÃO DE ANTICORPOS

O período evolutivo tem como objetivo melhorar de forma iterativa a população de possíveis soluções do problema. A cada iteração t' a população $AB_{t'}$ é submetida a todos os procedimentos anteriormente descritos, gerando um nova população $AB_{t'+1}$. Esta nova população é formalizada da seguinte forma:

$$AB_{t'+1} = \text{Supressao}(AB_{t'}) \cup \text{Elistimo}(AB_{t'}) \cup CL'_{t'}; \tag{29}$$

Este ciclo é repetido até que seja alcançado o número máximo de iterações $\#MaxIter$ (parâmetro configurado pelo usuário).

3.2.13 RETIRADA DOS EXEMPLOS DA BASE

Em um processo de classificação *single-label* baseado no procedimento de Extração Sequencial de Regras (WITTEN; FRANK, 2005) a retirada dos exemplos da base de treinamento

é extremamente simples. Esta retirada é realizada da seguinte maneira: se um exemplo é classificado (*matching* parcial ou total) pela melhor regra descoberta na iteração t e a classe associada ao exemplo é a mesma representada no consequente da regra, então o exemplo é retirado da base de treinamento. Em classificação *multi-label*, baseado no procedimento de Extração Sequencial de Regras, o processo é mais complicado, pois um regra descoberta pode cobrir parcialmente as classes associadas a um exemplo, impossibilitando assim a retirada deste exemplo. O trabalho de Chan e Freitas (2006) propõe um algoritmo de indução de regras baseado em colonia de formigas usando o procedimento de extração sequencial de regras para classificação *multi-label*, porém não-hierárquica. Deste modo, o MHCAIS pode ser considerado o primeiro algoritmo para classificação hierárquica e *multi-label* para indução de regras que usa este procedimento. Na continuidade da seção é descrito o mecanismo de retirada de exemplos da base de treinamento proposto neste trabalho.

Na versão global, todo $ag'_i \in AG'$ está associado a um vetor $\mathbf{q}_i$ de classes previstas pelo conjunto de regras CR até a iteração t corrente. O vetor $\mathbf{q}_i[q] = 1$ se o termo l_q referente à posição q não foi ainda previsto por CR, caso contrário $\mathbf{q}_i[q] = 0$, onde $l_q \in T_H$ (Equação 9). O vetor $\mathbf{q}_i$ possui o mesmo número m de elementos que $\mathbf{l}'_i$ e cada posição q corresponde ao mesmo termo $l_q \in T_H$ em ambos os vetores.

Inicialmente, em $t = 0$, os valores dos componentes de $\mathbf{l}'_i$ (Equação 14) são atribuídos a $\mathbf{q}_i$, pois nenhuma classe foi ainda prevista para ag'_i. Esta notação foi adotada para facilitar a implementação do procedimento, bem como a irrelevância de valores $\mathbf{l}'_i[q] = 0$ (classes não associadas a ag_i) para a retirada de exemplos da base. Assim:

$$\mathbf{q}_i^{t_0}[q] = \mathbf{l}'_i[q], q = 1, \ldots, m; \tag{30}$$

A cada regra descoberta em t, o vetor $\mathbf{q}_i$ é atualizado para iteração seguinte $t+1$. Esta atualização é realizada apenas para os ag'_i corretamente classificados por *MelhorRegra*. Em mineração de dados, um exemplo é corretamente classificado se o antecedente da regra classifica os atributos previsores do exemplo e este está associado à(s) mesma(s) classes(s) prevista(s) pela regra. Esta atualização é realizada da seguinte maneira:

$$\mathbf{q}_i^{t+1}[q] = \begin{cases} 0 & \text{se FitY}(y_q) > \delta_{FT} \wedge \mathbf{l}'_i[q] = 1 \\ \mathbf{q}_i^t[q] & \text{caso contrário} \end{cases}, \text{ para Afinidade}(ab, ag'_i) \geq \delta_{AF}; \tag{31}$$

onde $ab = MelhorRegra$.

Atualizado $\mathbf{q}_i$, os exemplos que tiveram todas as classes cobertas por *CR* são retirados de AG'. Estes exemplos são identificados de acordo com a seguinte equação:

$$\text{NcobClass}(ag'_i) = \sum_{q=1}^{m} \mathbf{q}_i^{t+1}[q]; \tag{32}$$

Se $\text{NcobClass}(ag'_i) = 0$, todas as classes de ag'_i foram cobertas por *CR*, e portanto deve ser retirado de AG'.

Os exemplos que devem permanecer na base de treinamento AG' para a próxima iteração t são obtidos da seguinte forma:

$$AG'_{t+1} = \{ag'_i \in AG'_t \mid \text{NcobClass}(ag'_i) > 0\}; \tag{33}$$

onde $\text{NcobClass}(ag'_i) > 0$ indica que há classes ainda não cobertas pelo classificador.

3.2.14 RECÁLCULO DO *FITNESS* DAS REGRAS DESCOBERTAS

Após o término do processo de construção do classificador, o *fitness* de todas as regras descobertas de *CR* é recalculado. Conforme detalhado na seção anterior, à medida que as regras são iterativamente descobertas, exemplos são removidos da base de dados. Deste modo, regras são descobertas sobre um conjunto cada vez menor de dados. Intuitivamente, à medida que o número de exemplos do conjunto de treinamento diminui, o suporte estatístico para cálculo do *fitness* das regras também diminui. Dessa forma, diminui o grau de confiabilidade nas regras do classificador sobre os dados. Portanto, em uma tentativa de computar a qualidade das regras de uma forma mais robusta em termos estatísticos, o *fitness* de todas as regras descobertas é recalculado conforme descrito na Subseção 3.2.5 sobre o conjunto de treinamento original com todos os exemplos da base.

Após o recálculo do *fitness* o classificador está preparado para ser usado sobre os dados do conjunto de teste, conforme será descrito na Seção 3.4.

3.3 DESCRIÇÃO DO MHCAIS-LOCAL

Diferentemente da versão global, o MHCAIS-local constrói para cada classe l do domínio da aplicação um classificador para discriminar se um exemplo pertence ou não à classe l predita. Todavia, algumas características e procedimentos são comuns entre as duas versões. Desta forma, nesta seção são detalhadas as diferenças entre as versões e mencionadas as similaridades.

Tal qual a versão global, o MHCAIS-local possui os procedimentos de Extração e Evolução de Regras. O funcionamento geral do procedimento de Evolução de Regras é idêntico para as versões (Figura 18). Por outro lado, o mecanismo de Extração de Regras apresenta algumas diferenças, conforme ilustra a Figura 23.

O fluxograma ilustrado na Figura 23 mostra que toda classe l do domínio de classes a serem preditas é selecionada para construção de um classificador. O MHCAIS-local recebe como entrada a base de dados de treinamento e a estrutura hierárquica a ser explorada. Os exemplos da base de dados são reclassificados e passam a estar associados a uma classe positiva ($l'_i = 1$) ou negativa ($l'_i = 0$), de acordo com a estrutura hierárquica de classes (Subseção 3.3.1). O conjunto de regras descobertas (CR^{l_q}) para prever a classe l_q é inicialmente vazio. O conjunto de regras CR^{l_q} é construído sequencialmente, tal qual na versão global, porém é dividido em duas etapas. Na primeira etapa, são descobertas regras para prever a classe positiva ($y^{l_q} = 1$); enquanto a segunda etapa descobre regras para prever a classe negativa ($y^{l_q} = 0$). No início de cada uma das duas etapas, o conjunto de treinamento está completo com todos os exemplos. Os dados são submetidos ao procedimento de Evolução de Regras que retornará a $MelhorRegra_y^{l_q}$ descoberta pelo SIA na iteração corrente para ser adicionada ao conjunto de regras descobertas (CR^{l_q}) da classe l_q em questão. Em seguida, os exemplos corretamente classificados por $MelhorRegra_y^{l_q}$ são removidos da base de treinamento. O ciclo de descoberta de regras para prever a classe l_q termina quando o MHCAIS-local descobrir quantas regras forem necessárias para cobrir os exemplos positivos e negativos da base de treinamento. Por fim, o *fitness* de todas as regras descobertas em CR^l é recalculado sobre a base de treinamento completa. Este processo se repete até que sejam construídos classificadores para todas as classes do domínio.

3.3.1 APLICAÇÃO DA HIERARQUIA DE CLASSES DA GO

Da mesma forma que na versão global, a estrutura hirárquica $\mathscr{H}$ dos termos (classes) da GO é fornecida como entrada para o algoritmo. O MHCAIS-local contrói um classificador para cada termo (classe) l_q que ocorre na base de dados e seus ancestrais (Ancestrais(l)). O procedimento para o levantamento das classes para as quais os classificadores serão construídos é detalhado na Seção 3.2.2 e formalizado pelas Equação 9, 10 e 11.

Usualmente, para construir classificadores locais, problemas de classificação hieráquica e *multi-label* são transformados em classificação plana e *single-label* (FREITAS; CARVALHO, 2007). Desse modo, cada exemplo da base de dados está associado a apenas uma classe, porém a hierarquia deve ser de algum modo considerada. Nesta tese a hierarquia de classes é representada da seguinte forma nos dados:

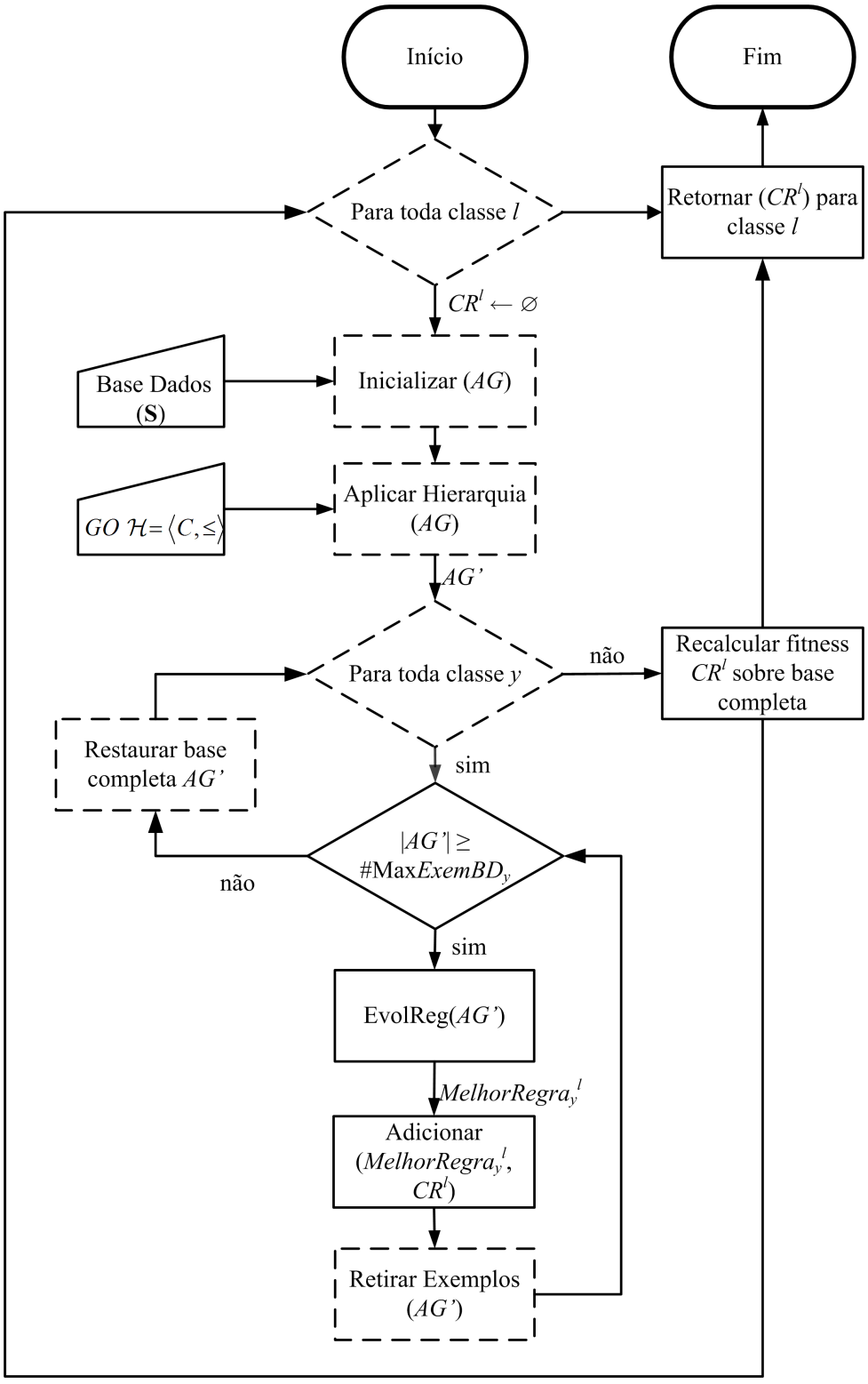


Figura 23: Fluxograma geral de funcionamento do procedimento de Extração de Regras para o MHCAIS-local. A linha não-contínua representa que o procedimento difere da versão global

$$AG' = \{ag'_i = \langle \mathbf{x}_i, l'_i \rangle \mid \exists ag_i \in AG\}_{i=1}^{|AG'|}, |AG'| = |AG|; \quad (34)$$

para:

$$l'_i = \begin{cases} 1 & \text{se } l_{ik} \equiv l \vee l_{ik} \in \text{Descendentes}(l_q) \\ 0 & \text{caso contrário} \end{cases}, \text{ para: } l_{ik} \in L_i; \quad (35)$$

onde $l_q \equiv c$ representa a classe para a qual o classificador será criado para prever e l_{ik} é a k-ésima classe anotada ao i-ésimo exemplo da base de dados. Portanto, na construção de um classificador para prever uma classe qualquer l, exemplos positivos são aqueles anotados com a classe l_q ou seus descendentes; caso contrário representarão exemplos negativos.

Um exemplo de estrutura hierárquica considerada pelo MHCAIS-local é apresentado na Figura 24. Nesta figura são ilustrados:

- Termos anotados diretamente para proteínas $T = \{17016, 3677, 3723, 4518, 16740\}$ nodos na cor preta;
- Termos ancestrais de T (nodos em cinza) $T_A = \{5488, 3824, 166, 3676, 16787, 16788\}$;
- $T_H = T \bigcup T_A$

Seja $l = 3676$ a classe para qual um classificador será treinado para a prever. Considerando as Equações 34 e 35, os exemplos positivos ($l'_i = 1$) no conjunto de treinamento são todos aqueles associados aos termos $l = 3676$ ou Descendentes$(l = 3676) = \{3677, 3723\}$, conforme ilustra a Figura 24 com linhas pontilhadas. Exemplos associados aos termos circulados com linha contínua representarão classes negativas e portanto exemplos negativos.

Diversos esquemas para definição de classes positivas e negativas são abordados na literatura, conforme é destacado por Eisner et al. (2005) e descrito brevemente na Subseção 2.7.4. Neste trabalho foi adotado o esquema de representar como classes positivas as proteínas anotadas à classe l_q e seus descendentes. Termos descendentes de uma classe podem ser considerados classes positivas pelos tipos de relacionamentos hierárquicos empregados na GO "é um" e "parte de" (The Gene Ontology Consortium, 2004). Com o uso de termos descendentes como classe positiva, aumenta-se o número de exemplos positivos, e assim espera-se aumento da correção preditiva do classificador.

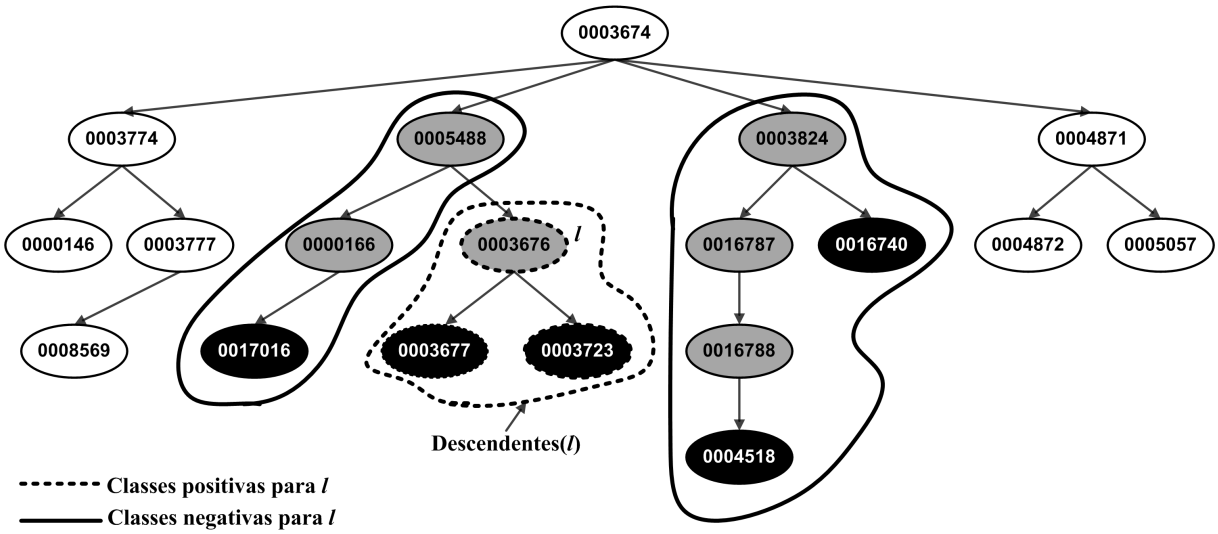


Figura 24: Exemplo de exploração hierárquica pelo MHCAIS-Local. Os nodos em cor preta representam termos diretamente anotados para as proteínas da base de dados. Nodos em cinza são resultado das exploração hierárquica e representam nodos antecedentes dos termos anotados. O nodo $l = 3676$ representa a classe considerada para construção do classificador. Os nodos circulados com linhas pontilhadas são descendentes de l e representam classes positivas para o classificador de l, enquanto que os nodos circulados com linha contínua representam classes negativa.

3.3.2 CRIAÇÃO DA POPULAÇÃO INICIAL DE ANTICORPOS

Na geração da população inicial de anticorpos, as condições que formam o antecedente de $\forall ab_j^{l_q} \in AB^{l_q}$ são geradas de forma aleatória, onde j representa o j-ésimo anticorpo (regra) da população para prever a classe l_q. A formalização do consequente é apresentada na Equação 5. O tamanho da população inicial AB^l (TPI) é determinado pelo usuário.

3.3.3 DEFINIÇÃO DAS FUNÇÕES DE *FITNESS*

Na versão local, como apenas uma classe é representada no consequente, as funções de *fitness* devem ser reformuladas para determinar a qualidade da regras. Deste modo, a regra $ab_j^{l_q}$ possui uma matriz de confusão única para contabilizar erros e acertos referentes às previsões positivas e negativas para a classe l_q. Assim, tem-se

- $VP_{y_j^{l_q}}$ (Verdadeiro Positivo): número de exemplos com Afinidade($ab_j^{l_q}$,ag_i') $\geq \delta_{AF}$ e $y_j^{l_q} = l_i'$;
- $VN_{y_j^{l_q}}$ (Verdadeiro Negativo): número de exemplos com Afinidade($ab_j^{l_q}$,ag_i') $< \delta_{AF}$ e $y_j^{l_q} \neq l_i'$;
- $FP_{y_j^{l_q}}$ (Falso Positivo): número de exemplos com Afinidade($ab_j^{l_q}$,ag_i') $\geq \delta_{AF}$ e $y_j^{l_q} \neq l_i'$; e
- $FN_{y_j^{l_q}}$ (Falso Negativo): número de exemplos com Afinidade($ab_j^{l_q}$,ag_i') $< \delta_{AF}$ e $y_j^{l_q} = l_i'$.

Contabilizados erros e acertos a função de *fitness* baseada em *f-measure* é reformulada para a seguinte equação:

$$\text{Fitness}(ab_j^{l_q}) = \frac{(\beta_{FT}^2+1)\times P \times R}{\beta_{FT}^2 \times P + R}, \beta_{FT} \in [0,\infty]; \tag{36}$$

$$P = \frac{VP_{y_j^{l_q}}}{VP_{y_j^{l_q}} + FP_{y_j^{l_q}}}; \tag{37}$$

$$R = \frac{VP_{y_j^{l_q}}}{VP_{y_j^{l_q}} + FN_{y_j^{l_q}}}; \tag{38}$$

Os termos P e R representam *precision* e *recall*, respectivamente. Para o cálculo do *fitness* tendo como base a função de Sensibilidade vs. Especificidade, a função de *fitness* é definida como:

$$\text{Fitness}(ab_j^{l_q}) = \frac{VP_{y_j^{l_q}}}{VP_{y_j^{l_q}} + FN_{y_j^{l_q}}} \times \frac{VN_{y_j^{l_q}}}{VN_{y_j^{l_q}} + FP_{y_j^{l_q}}}; \quad (39)$$

No MHCAIS-global se faz necessário a consistência hierárquica do valores de *fitness* haja vista que no consequente da regra é representada uma ou mais classes (Subseção 3.2.5.3), diferentemente do MHCAIS-local que prevê apenas uma classe por vez.

3.3.4 RETIRADA DOS EXEMPLOS DA BASE

Na versão local, o mecanismo de remoção de exemplos da base de treinamento à medida que regras são descobertas segue o modelo de padrão descrito na literatura (WITTEN; FRANK, 2005). Esta técnica é trivial, pois remove sequencialmente exemplos cobertos e corretamente classificados pela regra descoberta ($MelhorRegra^{l_q}$) na iteração t corrente. Formalmente, o conjunto de exemplos que não serão removidos e deverão permanecer para a próxima iteração é definido como:

$$AG'_{t+1} = \{ag'_i \in AG'_t \mid l'_i \neq y^{l_q} \vee \text{Afinidade}(MelhorRegra^{l_q}, ag'_i) < \delta_{AF}\}; \quad (40)$$

onde y^{l_q} representa a previsão da regra $MelhorRegra^{l_q}$.

3.4 CLASSIFICAÇÃO DE EXEMPLOS DE TESTE E AVALIAÇÃO DO CLASSIFICADOR

Finalizada a etapa de treinamento, o conjunto de regras descobertas ou os conjuntos (no caso de classificadores locais) são utilizados para classificar os exemplos da base de teste. Os exemplos que compõem o conjunto de teste não fazem parte da base de treinamento, portanto representam exemplos não vistos durante a construção do classificador. A classificação dos exemplos contidos no conjunto de teste tem como objetivo calcular a correção preditiva do conjunto de regras descobertas. A correção preditiva é considerada uma medida para avaliação de desempenho de algoritmos de mineração de dados (FREITAS, 2002).

A notação utilizada na etapa de teste é a seguinte para o MHCAIS-Global:

- conjunto de regras $CR = \{r_j\}_{j=1}^{|CR|}$, onde a regra $r_j \equiv ab_j$; e
- base de teste $\mathbf{T} = \{t_i\}_{i=1}^{|\mathbf{T}|}$, onde o exemplo de teste $t_i \equiv ag_i$.

Nesta etapa, inicialmente, para cada exemplo t_i é(são) identificada(s) a(s) regra(s) r_j que classifica(m) o mesmo. Lembrando que, um exemplo é considerado classificado se os valores de seus atributos previsores satisfazem (total ou parcialmente) o antecedente da regra. As classes previstas pelo classificador global são aquelas representadas no consequente das regras que classificam o exemplo testado e são definidas da seguinte forma:

$$H_{t_i}^{CR} = \bigcup y_{jq}, \forall r_j \in CR \mid \text{Afinidade}(r_j,t_i) \geq \delta_{AF} \wedge \text{FitY}(y_{jq}) \geq \delta_{CL} \in [0,1]^{\Re}; \quad (41)$$

onde y_{jq} representa a q-ésima classe representada no consequente da j-ésima regra; a restrição Afinidade$(r_j,t_i) \geq \delta_{AF}$ é definida pela Equação 15 e δ_{CL} especifica um limiar de classificação. A função FitY(y_{jq}) é definida na Equação 17 para *fitness* com base em *f-measure* ou Equação 16 para *fitness* com base na função sensibilidade vs. especificidade. Caso o exemplo t_i não seja classificado por regra alguma descoberta, o classificador prevê para este exemplo as classes presentes na regra padrão (*default*), que é normalmente usada em algoritmos de indução de regras. A regra padrão não possui antecedente e prevê, em geral, a classe mais frequente na base de dados. Todavia, o MHCAIS-global constrói regras *multi-label* e dessa forma tem representado no consequente da regra padrão todas as classes do domínio, onde o fator de confiança da previsão da classe y_{jq} é determinado pela sua frequência na base de dados ao invés do *fitness*.

A notação utilizada na etapa de teste é a seguinte para o MHCAIS-Local:

- conjunto de regras $CR^{l_q} = \{r_j^{l_q}\}_{j=1}^{|CR^{l_q}|}$, onde a regra $r_j^{l_q} \equiv ab_j$; e
- base de teste $\mathbf{T} = \{t_i\}_{i=1}^{|\mathbf{T}|}$, onde o exemplo de teste $t_i \equiv ag_i$.

onde CR^{l_q} representa o classificador construído para prever a q-ésima classe l do domínio.

A agregração de todas as previsões de todos os classificadores l_q treinados é definida por:

$$H_{t_i}^{CR} = \bigcup_{l_q \in T_H} l_q \mid y_j^{l_q} = 1 \quad (42)$$

onde:

$$j = \arg \max_{r_j^{l_q} \in CR^{l_q}} \text{Fitness}(r_j^{l_q}) \mid \text{Afinidade}(r_j^{l_q},t_i) \geq \delta_{AF} \quad (43)$$

A função Afinidade$(r_j^{l_q},t_i) \geq \delta_{AF}$ é definida pela Equação 15 e T_H é o conjunto de classes do domínio (Subseção 3.3.1). A Equação 42 mostra que para cada classificador (CR^{l_q}) construído

para prever uma classe $l_q \in T_H$, o MHCAIS-local identifica a regra com o maior *fitness* que classifique o exemplo t_i. Em classificação *single-label*, classificadores devem possuir alguma estratégia para resolução de conflitos quando regras prevendo classes distintas classificam o mesmo exemplo t_i. Para evitar esta ambiguidade, o MHCAIS-local usa o *fitness* obtido pela regra na etapa de treinamento como critério de "desempate". É importante ressaltar que cada conjunto de regras CR^{l_q} possui regras para prever se uma classe l_q está ou não associada ao exemplo t_i. Caso o exemplo t_i não seja classificado por regra alguma descoberta, o classificador prevê para este exemplo a classe presente na regra padrão (*default*) que prevê a classe mais frequente na base de dados.

3.4.1 AVALIAÇÃO DE DESEMPENHO COM BASE NA MEDIDA *F-MEASURE*

Uma vez determinadas as classes previstas, a qualidade da previsão $H_{t_i}^{CR}$ de CR para o exemplo t_i é avaliada. Para o cálculo da correção preditiva sobre os dados do conjunto de teste foi adotada a medida *F-Measure*, conforme já descrita nas Subseções 3.2.5.2 e 3.3.3. Desta forma, as funções *precision*(P), *recall* (R) e *F-Measure* são reformuladas da seguinte maneira:

$$P(t_i,H_{t_i}^{CR}) = \frac{|H_{t_i}^{CR} \cap L_{t_i}|}{|H_{t_i}^{CR}|}; \tag{44}$$

onde, L_{t_i} representa o conjunto de classes associadas ao exemplo de teste t_i.

$$R(t_i,H_{t_i}^{CR}) = \frac{|H_{t_i}^{CR} \cap L_{t_i}|}{|L_{t_i}|}; \tag{45}$$

$$\text{F-Measure}(t_i,H_{t_i}^{CR}) = \frac{(\beta^2+1) \times P(t_i,H_{t_i}^{CR}) \times R(t_i,H_{t_i}^{CR})}{\beta^2 \times P(t_i,H_{t_i}^{CR}) + R(t_i,H_{t_i}^{CR})}, \beta \in [0,\infty]; \tag{46}$$

O parâmetro β é especificado com o valor 1 indicando que *precision* e *recall* têm o mesmo peso no cálculo da correção preditiva.

Calculada a qualidade da previsão para cada exemplo t_i, o desempenho geral do classificador é obtido pela seguinte equação:

$$\text{Prec}(\mathbf{T}) = \frac{1}{|\mathbf{T}|} \sum_{t_i \in \mathbf{T}} \text{F-Measure}(t_i,H_{t_i}^{CR}); \tag{47}$$

Neste trabalho, o algoritmo é avaliado através da medida *F-Measure*, entretanto é facilmente adaptado para o uso de outras funções de avaliação de desempenho.

3.4.2 AVALIAÇÃO DE DESEMPENHO COM ÁREA DA CURVA *PRECISION-RECALL*

É desejável que algoritmos de classificação em mineração de dados, além de prever a(s) classe(s) para um exemplo qualquer, informem ao usuário o fator (valor) de confiança da previsão feita. Em alguns paradigmas usados em mineração de dados, esse valor de confiabilidade é inerente, como por exemplo, as redes bayesianas que fornecem uma probabilidade da previsão feita pelo classificador estar correta (mais informações no Apêndice B.4).

Algoritmos de classificação baseados em regras mostram para o usuário apenas as classes previstas, que em geral são nominais. Todavia, nada impede que exista um fator de confiabilidade para a classe nominal prevista. No MHCAIS foi adotado como fator de confiabilidade o *fitness* da regra obtido durante a fase treinamento do classificador. É importante ressaltar que no MHCAIS-Global todas as classes domínio são representadas no consequente da regra, e portanto, cada classe tem seu próprio *fitness*.

O uso de um fator de confiança na previsão realizada pelo classificador permite que seja utilizado um limiar de confiabilidade ou classificação (δ_{CL}). Dessa forma, o usuário pode especificar o grau de confiabilidade das classes previstas, desejando apenas as previsões com fator de confiabilidade maior ou igual ao limiar δ_{CL}.

Algoritmos de classificação são frequentemente avaliados, na literatura, quanto à sua sensibilidade ao valor de δ_{CL}. Essa análise pode ser feita através da geração de uma curva em um plano cartesiano. Para que se possa construir essa curva, é necessário gerar pontos com coordenadas (x,y). Dessa forma, cada valor δ_{CL} testado gera um ponto $p(x,y)$, onde x e y representam alguma medida de avaliação. A análise de desempenho de algoritmos através de curvas mais conhecida é a curva ROC (*receiver operating characteristic*) (LASKO et al., 2005). Cada ponto de uma curva ROC é obtido pelas seguintes equações:

$$x = TFP = \frac{FP}{VN+FP}; \tag{48}$$

$$y = TVP = \frac{VP}{VP+FN}; \tag{49}$$

onde:

- VP - número de exemplos categorizados como verdadeiro positivo
- VN - número de exemplos categorizados como verdadeiro negativo
- FP - número de exemplos categorizados como falso positivo

- FN - número de exemplos categorizados como falso negativo
- TVP - Taxa de Verdadeiro Positivo
- TFP - Taxa de Falso Positivo

É importante lembrar que para cada δ_{CL} um ponto $p(TFP,TVP)$ é gerado e todos os pontos encontrados devem gerar uma curva no plano cartesiano. Este trabalho não tem por obtivo discutir o uso e caracaterísticas da curva ROC, todavia é possivel notar que o ponto considerado ótimo é $p(0,1)$, ou seja 0% de classificações de falsos positivos e 100% de classificações positivas corretas.

Neste trabalho, a curva de desempenho que analisa a sensibilidade do classificador ao valor de δ_{CL} é a curva de *precision-recall*. Cada ponto p nesse tipo de curva é obtido pelas medidas de *precision* e *recall* formalizadas da seguinte maneira:

$$x = rec = \frac{VP}{VP+FN}; \tag{50}$$

$$y = prec = \frac{VP}{VP+FP}; \tag{51}$$

O valor de limiar δ_{CL}, em geral varia de 0 até 1; e à medida que este valor sobe, o número de classes positivas preditas diminue causando redução do valor de *recall*, enquanto o valor de *precision* tende a aumentar, porém nada impede que diminua (MANNING; RAGHAVAN; SCHüTZE, 2008).

De acordo com Vens et al. (2008) a curva de *precision-recall* é mais adequada para problemas de classificação *multi-label*, embora a curva ROC seja mais conhecida e utilizada. O argumento é justificado pelo fato que em classifição *multi-label*, geralmente classes do domínio da aplicação têm poucos exemplos positivos e por consequência aumento significativo de exemplos negativos. Esse desbalanceamento de classes é frequentemente encontrado em bases de dados biológicos. Geralmente, em bioinformática, o usuário está interessado em identificar as classes positivas de um exemplo, ao invés de reconhecer as classes que não estão associadas a esse exemplo. A identificação de classes corretamente não associadas a exemplos representa uma das preocupações da curva ROC através da taxa de falso positivo (TFP). O trabalho de Davis e Goadrich (2006) demonstra que, quando não se está interessado na taxa de falso positivo, a curva ROC pode apresentar resultados demasiadamente superestimados.

Muito embora a curva de *precision-recall* permita entender o comportamento e a sensibilidade do classificador a diferentes valores de limiar de classificação δ_{CL}, a comparação entre

classificadores pode se tornar difícil apenas pela representação gráfica. Desta maneira, é mais útil e simples comparar desempenho de algoritmos quando se pode extrair dessas curvas algum valor quantitativo. Este valor é frequentemente obtido através do cálculo da área da curva de *precision-recall* (AUPRC - *area under the precision-recall curve*). A área da curva de *precision-recall* varia de 0 a 1, quanto mais próximo de 1, melhor é o modelo de classificação.

Geralmente, para construir uma curva são obtidos alguns pontos pelo teste de alguns valores de δ_{CL}. Entretanto, para um cálculo mais real possível da área da curva, o uso de algum método de interpolação deve ser utilizado. Este trabalho usa um método de interpolação não-linear, haja vista que *precision* é não-linear no número de verdadeiro positivos e falso positvos (VENS et al., 2008). O método de interpolação não-linear utilizado neste trabalho e a demonstração da não-linearidade do *precision* são discutidos em detalhes no trabalho de Davis e Goadrich (2006).

Conforme mencionado no início da subseção as classes nominais devem ser transformadas para um fator de confiança, onde neste trabalho usa-se o *fitness* da classe obtido no treinamento.

No MHCAIS-Global o vetor de valores de confiança para cada classe q é obtido de seguinte forma.

$$\mathbf{h}_{t_i}^{CR}[q] = \begin{cases} \arg\max \mathrm{FitY}(y_{jq}) & \text{se Afinidade}(r_j,t_i) \geq \delta_{AF} \\ \mathrm{Freq}(y_q) & \text{senão, ativa regra padrão} \end{cases} ; \tag{52}$$

Se existir uma ou mais regras j que classifique(m) o exemplo t_i (Afinidade$(r_j,t_i) \geq \delta_{AF}$), o fator de confiança da q-ésima classe é o maior *fitness* representado nos consequentes das regras ativas. Todavia, se nenhuma regra for ativada, o exemplo é classificado pela regra padrão, cujo valor de confiança é determinado pela frequência da classe y_q (Freq(y_q)) na base de dados.

No MHCAIS-local o vetor de valores de confiança para cada classe q é obtido de seguinte forma.

$$\mathbf{h'}_{t_i}^{CR}[q] = \begin{cases} \arg\max \mathrm{Fitness}(r_j^{l_q}) & \text{se Afinidade}(r_j^{l_q},t_i) \geq \delta_{AF} \\ \arg\max \mathrm{Freq}(y^{l_q}) & \text{senão, ativa regra padrão} \end{cases} ; \tag{53}$$

A Equação 53 mostra que, se existir mais de uma regra $r_j^{l_q}$ classificando o mesmo exemplo, a resposta do classificador é a classe representada na regra de maior *fitness*; e por consequencia o fator de confiança é determinado por seu *fitness*. Caso contrário, o exemplo é classificado pela regra padrão, e portanto, a confiança é determinada pela frequência da classe mais frequente de y^{l_q} (positva ou negativa) – $\arg\max \mathrm{Freq}(y)$. É importante lembrar que para cada classe l_q do

domínio, um classificador local é construído para distinguir entre positivo ($y = 1$) ou negativo ($y = 0$).

Em classificadores locais, nos casos onde a previsão da classe l_q é negativa, deve-se usar no cálculo da confiança o complementar. Por exemplo, se para um exemplo qualquer o classificador prevê uma classe negativa com 0,7 de confiança, pela complementariedade, esse mesmo exemplo tem um fator de confiança de 0,3 associado à classe positiva, em uma escala de 0 a 1. Dessa forma, a Equação 53 é reformulada da seguinte maneira:

$$\mathbf{h}_{t_i}^{CR}[q] = \begin{cases} 1 - \mathbf{h}'^{CR}_{t_i}[q] & \text{se } y^{l_q} = 0 \\ \mathbf{h}'^{CR}_{t_i}[q] & \text{caso contrário} \end{cases}; \tag{54}$$

Desta forma, independente da versão do MHCAIS, toda classe do domínio tem um fator de confiança $\mathbf{h}_{t_i}^{CR}[q]$ da q-ésima classe ser positiva.

As classes associadas ao exemplo t_i (considerando a estrutura hierárquica de classes) são representadas em um vetor binário de classes $\mathbf{l}'_i$, onde a q-ésima posição possui valor 1 se l_q está associada ao exemplo t_i ou 0, caso contrário. Deste modo obtêm-se:

- *VP*: $\mathbf{l}'_i[q] = 1$ e $\mathbf{h}_{t_i}^{CR}[q] \geq \delta_{CL}$;
- *VN*: $\mathbf{l}'_i[q] = 0$ e $\mathbf{h}_{t_i}^{CR}[q] < \delta_{CL}$;
- *FP*: $\mathbf{l}'_i[q] = 0$ e $\mathbf{h}_{t_i}^{CR}[q] \geq \delta_{CL}$;
- *FN*: $\mathbf{l}'_i[q] = 1$ e $\mathbf{h}_{t_i}^{CR}[q] < \delta_{CL}$;

Há diversas maneiras de gerar uma curva e por fim obter sua área, como por exemplo: a média das áreas de cada classe l; média dos ponto encontrados e geração das curvas a partir dos pontos médios; e até mesmo usar ponderações para a área de cada curva. Neste trabalho optou-se por gerar a curva de *precision-recall* pelo desempenho global do classificador, ao invés de classes individuais. A obtenção de cada ponto $(rec, prec)$ é formalizada da seguinte maneira:

$$rec = \frac{\sum_q VP_q}{\sum_q VP_q + \sum_q FN_q}; \tag{55}$$

$$prec = \frac{\sum_q VP_q}{\sum_q VP_q + \sum_q FP_q}; \tag{56}$$

onde q representa a q-ésima classe do domínio. Esta forma de cálculo de *precision/recall* é frequentemente chamada na literatura de micro-média de *precision/recall* (KIRITCHENKO,

2005). Neste contexto o *recall* representa a porcentagem de exemplos de classes positivas que foram corretamente previstos como tendo classe positiva, enquanto que *precision* representa a porcentagem das previsões realizadas pelo classificador que estão corretas.

3.5 ABORDAGENS RELACIONADAS BASEADAS EM SISTEMAS IMUNOLÓGICOS ARTIFICIAIS

Esta seção descreve brevemente alguns trabalhos baseados em SIA aplicados em mineração de dados. A Tabela 6 mostra as características gerais dos algoritmos apresentados nesta seção, como a tarefa de mineração de dados, paradigma de treinamento, representação do conhecimento, classificadores *single* ou *multi-label*, aspectos imunológicos e tratamento de classes organizadas hierarquicamente.

O primeiro algoritmo baseado em SIA para classificação foi proposto por WATKINS; BOGGESS em 2002 chamado de AIRS (*artificial immune recognition system*). O AIRS é um algoritmo de classificação *single-label* e não-hierárquico cujo aprendizado é baseado em instância (*instance-based learning*) ou k-vizinho(s) mais próximo(s) (*nearest neighbor*), onde se mede a distância entre um exemplo e todos os exemplos da base de treinamento (WATKINS; BOGGESS, 2002). Em seguida, o algoritmo seleciona os k vizinhos de treinamento mais próximos do exemplo de teste e especifica como a classe predita a classe da maioria dentre os k vizinhos selecionados. Diferentemente do AIRS, o MHCAIS é um algoritmo baseado em indução de regras. No que diz respeito ao conhecimento extraído dos dados durante o treinamento, o AIRS não gera nenhuma forma explícita de representação do conhecimento obtido, pois o classificador construído é do tipo caixa-preta, enquanto o MHCAIS gera um modelo de predição representado na forma de regras SE-ENTÃO. Enquanto o MHCAIS lida tanto com atributos contínuos quanto categóricos, o AIRS considera apenas atributos contínuos. Dentre as similaridades, o AIRS e o MHCAIS apresentam os comum os princípios imunológicos da seleção clonal, hipermutação somática e rede imunógica. O AIRS passou por várias melhorias, onde mais informações podem ser encontradas em Watkins, Timmis e Boggess (2004)

Embora o AIRS seja considerado o primeiro algoritmo para classificação baseado em SIA, o aiNet (*Artificial Immune Network*) proposto por DE CASTRO e VON ZUBEN (2001) foi uma das primeiras abordagens usadas em mineração de dados. As similaridades entre aiNet e MHCAIS ficam restritas ao uso dos princípios da seleção clonal, hipermutação e rede imunológica. O aiNet foi aplicado na tarefa de agrupamento (*clustering*) em mineração de dados, enquanto o MHCAIS é focado em classificação.

O aiNet se tornou base para outros algoritmos, tal como o Copt-aiNet (*Artificial Immune*

Tabela 6: Algoritmos relacionados baseados em SIAs.

Algoritmo	Tarefa	Conhecimento	Treinamento	*Multi-label*	Hierárquico	SC[a]	HS [b]	RI [c]
MHCAIS	classificação	ER[d]	regras	sim	sim	sim	sim	sim
aiNET (DE CASTRO; VON ZUBEN, 2001)	agrupamento	BI[e]	-	não	não	sim	sim	sim
AIRS (WATKINS; TIMMIS; BOGGESS, 2004)	classificação	BI	caixa-preta	não	não	sim	sim	sim
IFRAIS (ALVES et al., 2004a)	classificação	ER	regras nebulosas	não	não	sim	sim	não
Copt-aiNET (CASTRO et al., 2005)	classificação	NR[f]	regras nebulosas	não	não	sim	sim	sim
Alatas e Akin (2005)	classificação	ER/*Boosting*	regras nebulosas	não	não	sim	sim	não
IFRAIS-T2F (MENOLASCINA et al., 2008)	classificação	ER	regras nebulosas	não	não	sim	sim	não
IFRAIS-B (MEZYK; UNOLD, 2009)	classificação	ER	regras nebulosas	não	não	sim	sim	não

[a]Seleção Clonal
[b]Hipermutação Somática
[c]Rede Imunológica
[d]Extração Sequêncial de Regras
[e]Baseado em Instâncias
[f]as n melhores regras da população representam o conjunto de regras descobertas

Network for Combinatorial Optimization) (CASTRO et al., 2005). Em comparação ao MHC-AIS, o Copt-aiNet apresenta as mesmas similaridades do aiNet, porém foi aplicado na tarefa de classificação, cujo conhecimento extraído é representado na forma de regras SE-ENTÃO. As regras geradas pelo Copt-aiNet possuem em seus consequentes apenas uma classe prevista, caracterizando-o como um algoritmo de classificação *single-label*, enquanto o MHCAIS é *multi-label*. Além disso, o Copt-aiNet não foi desenvolvido para classificação hierárquica. Outra diferença importante está relacionada ao esquema de treinamento, Copt-aiNet gera um conjunto de regras até atingir o número máximo de iterações, onde o número de regras do conjunto é um parâmetro do algoritmo. O treinamento no MHCAIS é baseado em um procedimento chamado neste trabalho de Extração Sequencial de Regras (*Sequential Covering*), onde o algoritmo descobre quantas regras forem necessárias para cobrir os exemplos da base de dados (WITTEN; FRANK, 2005). Por fim, o Copt-aiNet descobre regras nebulosas (*fuzzy*), enquanto o MHCAIS não.

Um outro algoritmo baseado em SIA foi proposto por Alatas e Akin (2005), onde as similaridades ficam restritas ao uso dos princípios da seleção clonal, hipermutação somática, aplicado em classificação e o conhecimento extraído é representado na forma de regras SE-ENTAO, entretanto as regras descobertas são nebulosa, tal como no Copt-aiNet. Por outro lado, entre as diferenças com o MHCAIS destaca-se que o modelo apresentado por Alatas e Akin (2005) não foi desenvolvido para classicação *multi-label* e classificação hierárquica. O esquema de treinamento também é diferente, pois em Alatas e Akin (2005) o treinamento é baseado em *boosting*, onde os exemplos da base de treinamentos estão associados a um peso, e durante o treinamento este peso vai decrescendo à medida que as regras são descobertas e classificando os exemplos da base. A construção do classificador termina quando todos os exemplos da base possuem seus pesos abaixo de um limiar. Por fim, o trabalho de Alatas e Akin (2005) não utiliza o princípio da rede imunológica.

Parte das características do MHCAIS foram originadas do algoritmo proposto em Alves et al. (2004b, 2004a), chamado de IFRAIS (*Induction Fuzzy Rules with an Artificial Immune System*). As características similares entre o MHCAIS e o IFRAIS são:

- inspirados nos princípios da seleção clonal e hipermutação somática;
- aplicados em classificação;
- conhecimento extraído é representado na forma de regras SE-ENTÃO; e
- baseados no mesmo esquema de treinamento, onde regras são sequencialmente descobertas, de tal forma que o treinamento termina quando os exemplos da bases são todos (ou

boa parte) cobertos pelo conjunto de regras.

Entretanto, MHCAIS apresenta diferenças em relação ao IFRAIS:

- o IFRAIS não foi desenvolvido para classificação *multi-label*, nem hierárquica;
- as regras descobertas pelo IFRAIS são nebulosas;
- o limiar de afinidade (δ_{AF}) é ajustado pelo IFRAIS de forma automática durante o treinamento, de maneira que é escolhido o valor para δ_{AF} que maximize o *fitness* da regra; e
- a teoria da rede imunológica não é empregada no IFRAIS.

O algoritmo IFRAIS foi melhorado por Mezyk e Unold (2009) apresentando os mesmo princípios imunológicos, paradigmas de mineração de dados e descoberta de regras nebulosas. A evolução do algoritmo original foi chamado de *buffered* IFRAIS (IFRAIS-B). Este nome foi atribuído tendo em vista o uso de tabelas *hash* para armazenar anticorpos repetidos. Com o uso deste tipo de tabela, o IFRAIS-B não recalcula o *fitness* de anticorpos repetidos durante o período evolutivo. Esse procedimento reduz significativamente o tempo de processamento do IFRAIS-B em comparação à sua versão original. Diferentemente do IFRAIS que cria a população inicial de anticorpos de forma aleatória, o IFRAIS-B usa o conceito de criação de população inicial uniforme (KARCI, 2004). Em uma população uniforme, para cada anticorpo (possível solução) gerado de forma aleatória, $2^r - 1$ anticorpos são criados usando o conceito de complementariedade, onde $r \in \mathbb{N}^*$. Para $r = 1$, cada anticorpo ab origina um $\overline{ab}$ com base no seu inverso (complementar). Se $r = 2$, para cada ab outros 3 anticorpos são gerados. Primeiramente, cria-se $\overline{ab}$. O segundo ab' será formado pelo inverso da primeira parte do vetor de características de ab, mais a segunda parte original. O terceiro ab'' é formado pela primeira parte original de ab, mais o inverso da segunda parte. O uso de criação de uma população uniforme apresentou pequena melhora de desempenho do IFRAIS-B, embora não significativa. Esse procedimento para criação de população inicial também foi usado no trabalho de Alatas e Akin (2005).

O algoritmo IFRAIS foi modificado por Menolascina et al. (2008) para contemplar o uso de conjuntos nebulosos do tipo-2 nas regras descobertas. Na versão original, o IFRAIS dicretiza atributos contínuos em termos linguísticos com funções de pertinência triangulares de acordo com a lógica nebulosa tradicional (PEDRYCZ; GOMIDE, 1998). Os conjuntos nebulosos do tipo-2 foram apresentados pela primeira vez por Zadeh (1975) como uma evolução da sua teoria dos conjuntos nebulosos tradicionais. Segundo a literatura, conjuntos nebulosos do tipo-2

atendem a dificuldade de determinar a função de pertinência adequada para conjuntos nebulosos tradicionais, e por consequência o nível exato de pertinência desses conjuntos (MENDEL; JOHN, 2002). De forma geral, conjuntos nebulosos tipo-2 criam uma terceira dimensão para tratar agora as incertezas envolvendo os conjuntos nebulosos tradicionais. Embora o trabalho de Menolascina et al. (2008) não detalhe o método proposto, nesta tese chamado de IFRAIS-T2F, resultados experimentais apresentam melhora de desempenho comparado à versão original do IFRAIS. Ademais, o IFRAIS-T2F apresenta os mesmos princípios imunológicos usados na versão antecessora.

Concluindo, embora haja vários outros SIAs para a tarefa de classificação em mineração de dados, esses outros trabalhos se limitam a classificação plana (não-hierárquica), e o MHCAIS descrito neste capitulo é o primeiro SIA projetado para construir classificadores hierárquicos e *multi-label*, o que caracteriza a originalidade desta tese.

4 METODOLOGIA DOS EXPERIMENTOS

Neste capítulo é descrita toda a metodologia de avaliação dos classificadores construídos, bases utilizadas nos experimentos computacionais e algoritmos usados para comparação.

Na Seção 4.2 são descritas as bases utilizadas nos experimentos computacionais e a etapa de pré-processamento realizada para que pudessem ser usadas.

As abordagens propostas nesta tese (MHCAIS global e local) são comparadas a outros dois algoritmos apresentados na literatura: PART e Clus. Estes algoritmos são brevemente descritos na Seção 4.3

A Seção 4.4 detalha a metodologia para avaliação de desempenho dos algoritmos comparados.

4.1 INTRODUÇÃO

O método proposto foi avaliado sobre a aplicação de predição de funções de proteínas que representa um problema de classificação hierárquica e *multi-label*. As classes a serem previstas representam funções biológicas de proteínas e são definidas na ontologia gênica (GO), mais especificamente a subontologia "função molecular". Deste modo, foi considerado para construção dos classificadores a organização hierárquica de classes na forma de um grafo acíclico direcionado (DAG). A capacidade do MHCAIS em construir classificadores globais e locais foi considerada e avaliada nos experimentos.

4.2 BASES DE DADOS

Nos experimentos computacionais foram utilizadas 10 bases de dados biológicos; destas uma base foi construída exclusivamente para os resultados apresentados nesta tese e as restantes são de domínio público. Ao longo desta seção, as bases são brevemente descritas e referenciadas. A Tabela 7 apresenta as características gerais das bases usadas, tais como: número de exemplos da base de dados, atributos previsores e domínio de classes a serem preditas.

Tabela 7: Principais características das bases de proteínas utilizadas.

Base	#Prot[a]	#APD[b]	#ATC[c]	#MTA[d]	#MTH[e]	#TDA[f]	#TCH[g]
ATP+DNAb	7877	38	2	6,25	25,62	109	214
CellCycle	1508	0	77	1,18	4,91	64	151
Church	1511	1	26	1,18	4,91	65	152
Derisi	1490	0	63	1,18	4,94	64	136
Eisen	1058	0	79	1,18	4,96	48	152
Gasch1	1515	0	173	1,18	4,90	65	152
Gasch2	1523	0	52	1,18	4,91	65	152
Pheno	417	69	0	1,14	4,72	23	61
Seq	1650	5	473	1,33	5,24	70	159
Spo	1482	3	77	1,18	4,92	64	151

[a]Número de Proteínas da Base de Dados
[b]Número de atributos previsores discretos (categóricos)
[c]Número de atributos previsores contínuos
[d]Média de termos diretamente anotados por proteína
[e]Média de termos anotados por proteína considerando a hierarquia da GO
[f]Total termos diretamente anotados nas proteínas da base dados
[g]Total termos anotados nas proteínas da base dados considerando a hierarquia da GO

A Tabela 7 mostra que ao considerar a estrutura hierárquica da GO, o número de classes a serem previstas pelo classificador aumenta significativamente. A base ATP+DNAb, por exemplo, tem seu o domínio de classes a serem preditas aumentado de 109 para 214 classes.

A base **ATP+DNAb** é formada por proteínas extraídas do repositório de dados biológicos Uniprot (Subseção 2.3.1.3) e foi usada primeiramente nos experimentos desta tese (ALVES et al., 2007; ALVES; DELGADO; FREITAS, 2008). As proteínas selecionadas pertencem às seguintes famílias (ALBERTS et al., 2002):

- *ligantes de DNA* (DNA *binding*) que auxiliam no processo de expressão gênica como ativadores de transcrição, dentre outras funções; e
- *ATPase* que são enzimas que catalisam a hidrólise do ATP (adenosina trifosfato) para originar ADP (adenosina difosfato) e fosfato inorgânico, liberando energia que é utilizada nas células.

A base ATP+DNAb é formada por 40 atributos previsores, sendo eles: padrões PROSITE (38 atributos), peso molecular e número de aminoácidos na sequência de proteínas. Concernente aos padrões PROSITE, estes são representados usando notação binária, indicando presença ou ausência do(s) padrão (padrões) em uma proteína.

Outras 9 bases de leveduras (*saccharomyces cerevisiae*) foram utilizadas nos experimentos, são elas: CellCycle (SPELLMAN et al., 1998), Church (ROTH et al., 1998), Derisi (DE

RISI; IYER; BROWN, 1997), Eisen (EISEN et al., 1998), Gasch1 (GASCH et al., 2000), Gasch2 (GASCH et al., 2001), Pheno (CLARE, 2003), Seq (CLARE, 2003) e Spo (CHU et al., 1998). No passado, essas bases foram usadas nos trabalhos de Clare (2003) e Vens et al. (2008). A bases originais podem ser encontradas em `http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets/`. Diferentes aspectos biológicos são abordados nestas bases:

Seq apresenta dados estatísticos da sequência de aminoácidos na construção dos atributos previsores. Alguns dos atributos previsores utilizados são: taxa de aminoácidos, tamanho da sequência, peso molecular, hidrofobicidade, propriedades extraídas usando ProtParam[20] e MIPS (MEWES et al., 1999).

Pheno contém informações fenotípicas de leveduras mutantes. Certos genes são desativados ou removidos para que se possa avaliar o comportamento do organismo a este procedimento. Segundo Vens et al. (2008), os dados foram extraídos de EUROFAN (OLIVER, 1996), MIPS (MEWES et al., 1999) and TRIPLES (KUMAR et al., 2000).

CellCycle, Church, Derisi, Eisen, Gasch1, Gasch2 e Spo são formadas por dados biológicos de expressão gênicas representados na forma de *microarray*.

4.2.1 PRÉ-PROCESSAMENDO DAS BASES DE DADOS

Para a realização dos experimentos, as bases anteriomente descritas passaram por uma etapa de pré-processamento. No caso específico da base ATP + DNBb, o pré-processamento se deu da seguinte maneira:

1. As proteínas armazenadas na base Uniprot são divididas em *revisadas* e *não-revisadas*. As proteínas revisadas foram manualmente curadas, enquanto que as não-revisadas ainda não, todavia serão curadas e validadas. Para a construção da base de dados foram consideradas apenas as proteínas revisadas.

2. Foram consideradas proteínas que tivessem ao menos um padrão PROSITE anotado. Este critério foi adotado por saber que regiões conservadas (padrões) em proteínas estão de alguma forma relacionadas à função biológica da mesma.

3. Os padrões (*motifs*) PROSITE são divididos em duas categorias: *expressão regular* e *perfil* (Seção 2.3.2). Estes padrões são utilizados para formar os atributos previsores de cada proteína. Apenas os padrões do tipo *expressão regular* foram considerados como

[20] http://www.expasy.org/tools/protparam.html

atributos previsores, pela facilidade de representá-los de forma binária. Neste tipo de representação, adotada nesta tese, cada *bit* indica presença ou ausência de um determinado padrão (expressão regular) em uma proteína qualquer.

4. Foram consideras proteínas que tivessem ao menos um termo da GO anotado, desde que este termo pertença ao domínio *função molecular* da GO (Seção 2.4), lembrando que os termos GO representam as classes previstas pelo algoritmo.

5. Foram considerados para representar os atributos previsores padrões PROSITE anotados a no mínimo 10 proteínas. Este procedimento teve como objetivo remover padrões muito específicos.

6. O conjunto de classes a serem preditas pelo classificador foi formado por termos GO com ao menos 20 proteínas anotadas. Este procedimento tem a finalidade de garantir um mínino de suporte estatístico ao treinamento do algoritmo.

Conforme mencionado anteriormente, a base ATP+DNAb foi criada exclusivamente para os experimentos computacionais desta tese, todavia resultados preliminares foram apresentados em Alves et al. (2007), Alves, Delgado e Freitas (2008).

O pré-processamento do restante das bases se deu de forma diferente, pois os exemplos já estavam preclassificados e com seus respectivos atributos previsores. Dessa forma, a etapa de processamento compreendeu apenas a exclusão dos termos da GO pouco frequentes na base de dados. A frequência mínima exigida foi de 10 proteínas anotadas para cada termo da GO para garantir um mínimo de suporte estatístico na construção dos classificadores. Diferente da base ATP+DNAb, na etapa de pré-processamento das bases anteriormente citadas, observou-se que o aumento da frequência mínima de classes anotadas para valores superiores a 10 resultava em um reduzido conjunto de classes a serem preditas. Assim, ao invés do valor 20, optou-se por utilizar a frequência mínima igual a 10. Além da frequência, foram excluídos também os termos da GO que não pertencessem ao domínio de termos que definem funções moleculares, tal qual a base de dados ATP+DNAb.

4.3 BREVE DESCRIÇÃO DOS ALGORITMOS USADAS PARA COMPARAÇÃO

Esta seção descreve sucintamente os algoritmos PART e Clus usados nos experimentos comparativos com as abordagens propostas nesta tese.

O algoritmo PART (*partial decision tree*) foi proposto por Frank e Witten (1998) e contrói classificadores formados por regras na forma SE <*antecedente*> ENTÃO <*consequente*>

a partir de árvores "parciais" de decisão. Tradicionalmente, em algoritmos de mineração de dados baseados em árvores de decisão, uma árvore contruída durante a etapa de treinamento pode ser transformada em várias regras na forma SE <*antecedente*> ENTÃO <*consequente*>. Diferentemente, o PART constrói várias sub-árvores, onde cada sub-árvore é transformada em apenas um regra (FRANK; WITTEN, 1998). O algoritmo PART constrói classificadores locais (binários) e portanto, para cada classe do domínio um classificador é gerado. O PART compõe o pacote de algoritmos para mineração de dados Weka e pode ser encontrado em `http://www.cs.waikato.ac.nz/ml/weka/`.

O algoritmo Clus constrói classificadores baseados em árvores de decisão e vem sendo investigado e melhorado desde 1997 (DE RAEDT; BLOCKEEL, 1997). A versão usada nos experimentos comparativos, gera classificadores globais para classificação hierárquica e *multi-label* (VENS et al., 2008). Deste modo, cada nodo folha prevê uma ou mais classes. Embora o Clus construa árvores de decisão, há um parâmetro do algoritmo que transforma as árvores geradas em regras na forma SE <*antecedente*> ENTÃO <*consequente*>. Atualmente o Clus encontra-se na versão 2.8 e está diponível em `http://www.cs.kuleuven.be/~dtai/clus/index.html`

O Apêndice B.1 descreve de forma sucinta o paradigma computacional de árvores de decisão usado em mineração de dados.

4.4 CRITÉRIOS DE AVALIAÇÃO

Os resultados são apresentados e analisados sob os seguintes critérios: correção preditiva com base na medida *f-measure* ou área da curva *precision-recall* (AUPRC) e simplicidade do conjunto de regras descobertas. Entretanto, é importante ressaltar que, ao longo dos experimentos realizados, a simplicidade das regras não foi considerada na função de *fitness* durante o treinamento dos classificadores.

A correção preditiva (taxa de acerto) foi calculada considerando-se a base de teste (contendo exemplos não vistos durante a fase de treinamento), conforme discutido na Seção 3.4. A técnica adotada para obtenção desta medida de precisão foi a validação cruzada (WEISS; KULIKOWSKI, 1991). A validação cruzada consiste, primeiramente, em dividir aleatoriamente a base de dados original em k partições iguais ou aproximadamente iguais. Selecionam-se $k-1$ partições para formar a base de treinamento e, em seguida, computa-se a correção preditiva, conforme descrito na Seção 3.4.1, para a partição que ficou de fora. Este passo se repete até que todas as k partições sejam usadas $k-1$ vezes como base de treinamento e uma vez como base de teste. Ao final, a correção preditiva é obtida pela média aritmética das taxas de acerto

obtidas (nos dados de teste) para as k partições. Neste trabalho a base original foi dividida em $k = 10$ partições.

Outra forma de comparar o desempenho entre algoritmos é calcular a área da curva *precision-recall*. Desta maneira, cada base utilizada nos experimentos foi dividada em dois subconjuntos disjuntos: treinamento (2/3 dos exemplos) e teste (1/3 dos exemplos). Conforme já mencionado, os exemplos do conjunto de teste não participam do treinamento do classificador construído. A área da curva *precision-recall* é calculada sobre o conjunto de teste e o valor desta área varia de 0 a 1. Quanto mais próximo de 1 o valor da área, melhor é considerado o resultado, conforme detelhado na Seção 3.4.2.

O desempenho com base na correção preditiva da medida *f-measure* e validação cruazada foi usado na análise de sensibilidade dos parâmetros e procedimentos do MHCAIS (Seção 5.1) e para comparação entre os algoritmos MHCAIS e PART (Seção 5.2). Devido a dificuldade de se obter a área da curva de *precision-recall* no algoritmo PART, esta medida não foi usada para comparar MHCAIS vs. PART.

A divisão da base de dados em treinamento (2/3) e teste (1/3) foi adotada para comparar os resultados entre MHCAIS e Clus. Esta estratégia foi adotada considerando que as 9 bases de domínio pública, descritas anteriormente, já se encontrarem divididas no repositório `http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets/`. Como medida de análise de desempenho a área da curva *precision-recall* foi adotada para comparação entre MHCAIS e Clus, haja vista a diculdade de se obter o valor de *f-measure* de cada previsão no algoritmo Clus. Estes experimentos computacionais são apresentados na Seção 5.3.

Com relação à avaliação da simplicidade das regras descobertas, os resultados foram analisados considerando-se o número de regras e condições presentes no conjunto de regras, onde quanto menor estes números, mais simples é o conhecimento descoberto.

5 EXPERIMENTOS COMPUTACIONAIS E RESULTADOS

Neste capítulo são apresentados, analisados e discutidos os resultados obtidos durantes os diversos experimentos computacionais realizados.

A Seção 5.1 apresenta resultados computacionais que visam compreender o comportamento do MHCAIS quando os valores de certos parâmetros e procedimentos são variados.

As análises comparativas de desempenho entre o MHCAIS e os algoritmos PART e Clus são apresentadas e discutidas na Seções 5.2 e 5.3

5.1 INFLUÊNCIA DA VARIAÇÃO DE VALORES DE PARÂMETROS E PROCEDIMENTOS NO DESEMPENHO DO MHCAIS

Nesta seção são apresentados e analisados os resultados que permitem entender a influência e a sensibilidade do MHCAIS global e local à variação dos valores de alguns parâmetros; o uso de procedimentos de busca local e diferentes funções de *fitness*. A base usada nos experimentos foi a ATP+DNAb, descrita na Tabela 7. Os critérios de desempenho considerados foram correção preditiva com base na medida *f-measure* e a simplicidade do conjunto de regras descobertas com validação cruzada de tamanho $k = 10$, devidamente detalhados na Seção 4.4. Tendo em vista a divisão da ATP+DNAb em 10 partições para uso da validação cruzada, nesta seção esta base será chamada de ATP+DNAb-1. A sobreposição do desvio-padrão foi adotada como critério de análise estatística.

A Tabela 8 apresenta os valores utilizados nos experimentos para cada parâmetro do MHCAIS nas versões global e local. Nesta tabela alguns valores de parâmetros estão representados na forma de intervalo, todavia em cada subseção o valor de cada parâmetros será devidamente especificado.

Tabela 8: Valores utilizados para os parâmetros do MHCAIS nos experimentos.

Parâmetro	Definição	Global	Local
δ_{AF}	limiar de *matching*	$\{0{,}8; 0{,}9; 1{,}0\}$	
δ_{FT}	limiar de *fitness*	$[0{,}5; 0{,}9]$	-
δ_{CL}	limiar de classificação	0,9	-
δ_{SIM}	limiar de similaridade ente *ab*'s	0,7	0,7
β_{FT}	utilizado no função fitness (*f-measure*)	$[0{,}05; 80]$	1
$MaxExemBD$	número de exemplos não cobertos na base de treinamento	10	10
$HiperClTx$	taxa de hiper-expansão clonal	2,0	2,0
$MutMin$	taxa mínima de mutação	0,01	0,01
$MutMax$	taxa máxima de mutação	0,5	0,5
$\#MaxCl$	número máximo de clones	10	10
$\#MaxIter$	número iterações do período evolutivo	50	50
TPI	tamanho da população inicial	100	100
$TPMax$	tamanho máximo da população	500	500

5.1.1 INFLUÊNCIA DO NÚMERO DE ITERAÇÕES DO PERÍODO EVOLUTIVO

Nesta subseção são apresentados resultados de experimentos que permitem compreender a influência do parâmetro que controla o número de iterações por período evolutivo ($\#MaxIter$) no desempenho do MHCAIS. É importante lembrar que, em cada período evolutivo uma regra é encontrada e adicionada ao conjunto de regras descobertas. Portanto, o MCHAIS executa quantos períodos evolutivos forem necessários para construção do classificador.

Os experimentos realizados ficaram restritos apenas à versão global do MHCAIS e os valores dos parâmetros do MHCAIS são especificados na Tabela 8. Outros valores de parâmetros são:

- $\delta_{AF} = \{0{,}8; 0{,}9; 1{,}0\}$;
- $\delta_{FT} = 0{,}9$; e
- $\beta_{FT} = 1{,}0$

A Tabela 9 apresenta os resultados obtidos para os critérios de correção preditiva e simplicidade com seus respetivos desvio padrões ($\pm$). Os resultados são comparados entre mesmos valores de δ_{AF}, onde para cada medida de desempenho o melhor resultado é destacado em negrito.

De acordo com a Tabela 9, é possível observar que o aumento do número de iterações ($\#MaxIter$) por período de evolutivo de 50 para 100 não garantiu ao MHCAIS a melhora de

Tabela 9: Resultados para valores do parâmetro número de iterações #*MaxIter*.

		Correção Preditiva			Simplicidade	
#MaxIter	δ_{AF}	*Precision*	*Recall*	*F-Measure*	#Regras	#TTCond
50	0,8	89,76 ± 0,6	**70,68** ± 1,1	75,70 ± 0,9	**17,1** ± 0,8	**110,2** ± 6,8
	0,9	**88,98** ± 0,3	62,43 ± 1,8	69,41 ± 1,4	**13,3** ± 0,3	60,1 ± 6,7
	1,0	**89,04** ± 0,4	**58,12** ± 0,9	**66,38** ± 0,6	**11,8** ± 0,5	25,4 ± 3,8
100	0,8	**90,78** ± 0,6	70,51 ± 1,5	**75,90** ± 1,0	18,4 ± 0,4	124,4 ± 7,1
	0,9	88,84 ± 0,4	**64,90** ± 1,6	**71,29** ± 1,1	13,9 ± 0,3	**55,8** ± 6,5
	1,0	89,02 ± 0,5	57,96 ± 0,9	66,28 ± 0,6	12,2 ± 0,4	**22,5** ± 2,7

desempenho em todas medidas de desempenho consideradas. Embora o MHCAIS tenha obtido melhores resultados de *f-measure* para $\delta_{AF} = \{0{,}8; 0{,}9\}$, esta melhora não foi significativa, considerando o desvio padrão, para justificar o uso de duas vezes mais iterações por período evolutivo, já que o custo computacional tende a dobrar.

5.1.2 INFLUÊNCIA DA FUNÇÃO DE FITNESS

Nesta subseção são apresentados resultados que têm como objetivo comparar a influência da função de *fitness* (usada durante a fase de treinamento) no desempenho preditivo do MHCAIS. Nas Subseções 3.2.5 e 3.3.3 são definidas duas funções de *fitness*: sensibilidade vs. especificidade e *f-measure*. A função sensibilidade vs. especificidade considera o produto do percentual da cobertura das classes positivas corretamente preditas pela cobertura das classes negativas corretamente não preditas. A função *f-measure* representa a média harmônica entre o percentual das predições corretas e cobertura das classes positivas preditas. A função sensibilidade vs. especificidade tem sido aplamente usada em trabalhos de mineração de dados e por isso foi escolhida. A função *f-measure* tem sido igualmente usada, todavia a maior motivação para avaliá-la como função de *fitness* se deve ao fato de representar a mesma função usada para o cálculo de desempenho de correção preditiva sobre os dados do conjunto de teste.

Os parâmetros utilizados nos experimentos foram os mesmos apresentados na Tabela 8 para *matching* parcial e total, exceção feita ao parâmetro $\delta_{FT} = 0{,}5$ (global) e β_{FT} especificado com o valor 1,0. É importante lembrar que o parâmetro β_{FT} está presente apenas na função *f-measure*. A Tabela 10 apresenta os resultados obtidos para correção preditiva: *precision*, *recall* e *f-measure*. Conforme outros experimentos em subções anteriores o símbolo ± representa o desvio-padrão obtido e os melhores resultados, no comparativo entre as funções de *fitness*, são apresentados em destaque com negrito.

No que diz respeito ao MHCAIS-global, observa-se na Tabela 10 que o uso da função sensibilidade/especificidade (Sensib/Espec) obtém melhores resultados para a medida *recall*. Por

Tabela 10: Resultados comparativos entre funções de *fitness* Sensib/Espec vs. *F-Measure*.

MHCAIS-Global						
	Sensib/Espec			*F-Measure*		
δ_{AF}	*precision*	*recall*	*f-measure*	*precision*	*recall*	*f-measure*
0,8	$37{,}74 \pm 1{,}9$	**98,17** $\pm 0{,}3$	$50{,}63 \pm 1{,}8$	**51,01** $\pm 1{,}3$	$97{,}47 \pm 0{,}2$	**64,13** $\pm 1{,}2$
0,9	$41{,}57 \pm 1{,}6$	**97,85** $\pm 1{,}0$	$53{,}22 \pm 1{,}2$	**54,96** $\pm 2{,}8$	$95{,}75 \pm 0{,}4$	**66,05** $\pm 2{,}0$
1,0	$40{,}27 \pm 1{,}4$	**99,57** $\pm 0{,}3$	$52{,}72 \pm 1{,}2$	**52,35** $\pm 2{,}1$	$95{,}42 \pm 0{,}4$	**63,94** $\pm 1{,}5$
MHCAIS-Local						
0,8	$80{,}58 \pm 1{,}0$	$44{,}65 \pm 1{,}5$	$55{,}65 \pm 1{,}4$	**90,91** $\pm 0{,}2$	**87,32** $\pm 0{,}3$	**87,96** $\pm 0{,}2$
0,9	$75{,}61 \pm 1{,}1$	$52{,}57 \pm 2{,}3$	$59{,}75 \pm 1{,}7$	**89,69** $\pm 0{,}3$	**87,13** $\pm 0{,}4$	**87,27** $\pm 0{,}3$
1,0	$58{,}56 \pm 1{,}0$	$69{,}91 \pm 1{,}1$	$61{,}37 \pm 0{,}8$	**84,64** $\pm 0{,}2$	**87,09** $\pm 0{,}4$	**84,63** $\pm 0{,}5$

outro lado, a utilização da função *f-measure* para cálculo do *fitness* resulta significativa melhora no valor de *precision*. Considerando a medida *f-measure* sobre os dados do conjunto de teste, com a função de *fitness f-measure* o MHCAIS-global obteve resultados significativamente superiores do que com o uso da função sensibilidade/especificidade, alavancado pela diferença de *precision* entre as duas funções de *fitness*. A medida de *recall* para o MHCAIS-global não apresentou melhora significativa, tendo em vista característica implícita desta versão em favorecer o *recall* por representar no consequente todas as classes do domínio. A diferença de desempenho para a medida *precision* entre as duas medidas de *fitness* avaliadas, pode ser explicada porque a função sensibilidade/especificidade tende a privilegiar a cobertura correta das classes positivas e negativas, enquanto a função *f-measure* tem a pré-disposição nas previções corretas e cobertura das classes positivas.

O MHCAIS-local apresentou significativa diferença das três medias de desempenho entre as funções de *fitness* sensibilidade/especificidade e *f-measure*. Essa diferença pode ser novamente explicada pelo compromisso da função de *fitness f-measure* com as previsões corretas e a cobertura das classes positivas.

5.1.3 SENSIBILIDADE DO PARÂMETRO δ_{FT} NO MHCAIS-GLOBAL

O conjunto de experimentos descritos e apresentados nesta subseção tem como objetivo investigar a sensibilidade no desempenho do MHCAIS-Global quando utilizados diferentes valores para o parâmetro limiar de *fitness* δ_{FT}. É importante lembrar que o parâmetro δ_{FT} está presente apenas na versão global do MHCAIS (conforme Seção 3.2), e por isso os experimentos ficaram restritos a esta versão. Os valores dos parâmetros utilizados nestes experimentos são os mesmos apresentados na Tabela 8, exceção feita ao parâmetro β_{FT} usado na função de *fitness*, cujo valor especificado foi de 1,0. Lembrando que o valor β_{FT} determina o peso da medidas *precision* e *recall* no *fitness* da regra; para $\beta_{FT} = 1{,}0$ ambas têm o mesmo peso. Os valores

testados para o parâmetro analisado nesta subseção são: $\delta_{FT} = \{0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9\}$.

As Tabelas 11 e 12 apresentam os resultados obtidos para os critérios de correção preditiva e simplicidade do conhecimento descoberto. Estas tabelas trazem os valores de *precision*, *recall*, *f-measure*, número de regras (#Regras) e total de condições do conjunto de regras (#TTCond). Novamente os experimentos foram realizados usando *matching* parcial e total (δ_{AF}). O símbolo $\pm$ indica o desvio padrão, enquanto $\oplus(\ominus)$ representa o melhor (pior) resultado obtido para cada medida de desempenho. Valores com destaque em negrito indicam o melhor resultado para cada valor δ_{AF} entre os valores de δ_{FT} considerados. Na Tabela 12 é apresentado também o número de classes preditas para cada exemplo do conjunto de teste (#Pred/Exemp). Esta nova informação permitirá entender o comportamento das medidas de desempenho utilizadas.

Tabela 11: Precisão obtida pelo MHCAIS para valores do parâmetro δ_{FT}.

δ_{AF}	δ_{FT}	*Precision*	*Recall*	*F-Measure*
0,8	0,5	$\ominus$60,23 $\pm$ 0,6	$\oplus$**93,40** $\pm$ 0,3	69,56 $\pm$ 0,4
	0,6	69,56 $\pm$ 0,8	89,62 $\pm$ 0,6	74,23 $\pm$ 0,5
	0,7	75,21 $\pm$ 1,1	86,99 $\pm$ 0,3	77,40 $\pm$ 0,9
	0,8	86,90 $\pm$ 0,7	76,53 $\pm$ 1,0	$\oplus$**77,60** $\pm$ 0,4
	0,9	$\oplus$**89,76** $\pm$ 0,6	70,68 $\pm$ 1,1	75,79 $\pm$ 0,9
0,9	0,5	63,66 $\pm$ 0,6	**92,38** $\pm$ 0,2	71,52 $\pm$ 0,4
	0,6	70,74 $\pm$ 0,9	88,58 $\pm$ 0,3	74,63 $\pm$ 0,5
	0,7	75,55 $\pm$ 1,1	82,60 $\pm$ 0,8	74,13 $\pm$ 0,6
	0,8	87,75 $\pm$ 0,3	71,20 $\pm$ 0,4	**75,10** $\pm$ 0,3
	0,9	**88,98** $\pm$ 0,3	62,43 $\pm$ 1,8	69,41 $\pm$ 1,4
1,0	0,5	64,36 $\pm$ 0,4	**91,31** $\pm$ 0,3	71,58 $\pm$ 0,2
	0,6	69,61 $\pm$ 0,4	89,66 $\pm$ 0,1	73,91 $\pm$ 0,2
	0,7	80,79 $\pm$ 0,7	77,63 $\pm$ 0,9	**75,35** $\pm$ 0,6
	0,8	87,70 $\pm$ 0,2	69,40 $\pm$ 1,0	74,03 $\pm$ 0,7
	0,9	**89,04** $\pm$ 0,4	$\ominus$58,12 $\pm$ 0,9	$\ominus$66,38 $\pm$ 0,6

A Tabela 11 mostra que os melhores valores de *precision*, para cada valor de δ_{AF}, são obtidos com valores mais altos do limiar de *fitness* δ_{FT}; o contrário é visto para a medida *recall*, onde os melhores valores são observados para os valores mais baixo de δ_{FT}. No critério simplicidade do conhecimento descoberto (Tabela 12), valores altos de δ_{FT} resultam em menos regras e condições, enquanto para os valores mais baixos δ_{FT} estes números aumentam.

No que diz respeito à correção preditiva, o grande desafio é encontrar o valor δ_{FT} que apresente um melhor compromisso entre *precision* e *recall* com a medida *f-measure*. É importante lembrar que a medida *f-measure* representa a média harmônica entre *precision* e *recall*. Neste sentido, os melhores valores de *f-measure* para *matching* parcial ($\delta_{AF} = \{0{,}8; 0{,}9\}$) foram obtidos para $\delta_{FT} = 0{,}8$, enquanto para o *matching* total ($\delta_{AF} = 1{,}0$) o melhor resultado foi alcançado com $\delta_{FT} = 0{,}7$.

Tabela 12: Simplicidade das regras descobertas para valores de δ_{FT}.

δ_{AF}	δ_{FT}	#Regras	#TTCond	#Pred/Exemp
0,8	0,5	$\ominus$53,8 $\pm$ 1,6	$\ominus$466,8 $\pm$ 23,0	40,2 $\pm$ 0,5
	0,6	36,7 $\pm$ 0,8	337,6 $\pm$ 14,9	33,3 $\pm$ 0,7
	0,7	32,3 $\pm$ 0,8	248,1 $\pm$ 8,7	29,0 $\pm$ 0,7
	0,8	24,9 $\pm$ 0,4	162,9 $\pm$ 7,4	21,4 $\pm$ 0,4
	0,9	**17,1** $\pm$ 0,8	**110,2** $\pm$ 6,8	18,8 $\pm$ 0,4
0,9	0,5	44,3 $\pm$ 1,6	257,0 $\pm$ 16,0	37,5 $\pm$ 0,5
	0,6	32,4 $\pm$ 0,6	173,5 $\pm$ 5,6	32,2 $\pm$ 0,7
	0,7	28,7 $\pm$ 1,1	143,8 $\pm$ 9,0	28,4 $\pm$ 1,0
	0,8	18,2 $\pm$ 0,5	78,4 $\pm$ 7,0	19,4 $\pm$ 0,1
	0,9	**13,3** $\pm$ 0,3	**60,1** $\pm$ 6,7	16,1 $\pm$ 0,6
1,0	0,5	33,4 $\pm$ 1,8	71,5 $\pm$ 8,1	36,7 $\pm$ 0,2
	0,6	32,2 $\pm$ 1,1	59,2 $\pm$ 4,9	33,8 $\pm$ 0,2
	0,7	22,8 $\pm$ 0,7	42,8 $\pm$ 5,3	23,7 $\pm$ 0,7
	0,8	15,6 $\pm$ 0,5	34,8 $\pm$ 5,1	18,8 $\pm$ 0,3
	0,9	$\oplus$**11,8** $\pm$ 0,5	$\oplus$**25,4** $\pm$ 3,8	14,7 $\pm$ 0,3

Os melhores resultados de *precision* para valores mais altos de δ_{FT} eram esperados, pois ao forçar a previsão de classes com alto fator de confiança (*fitness* da classes), diminui a probabilidade de uma previsão errada. Na Tabela 12 observa-se que, quanto menor o valor δ_{FT}, mais classes são preditas por exemplo (#Pred/Exemp). Possivelmente, isto explique o melhor (pior) desempenho da medida *recall* e baixo (alto) desempenho da medida *precision* para alguns valores de δ_{FT}. Em classificação *multi-label* um exemplo está associada à uma ou mais classes, onde este número é $n \geq 1$. Deste modo, para #Pred/Exemp $> n$ o valor de *recall* tende a aumentar e o *precision* diminuir. De forma análoga, para #Pred/Exemp $< n$ o comportamento contrário acontece. É possivel notar este comportamento com o dado mostrado na Tabela 7, onde ATP+DNAb está associado em média a 25,62 classe. De acordo com a Tabela 12, considerando $\delta_{AF} = 0{,}8$ e $\delta_{FT} = 0{,}5$, o número de classes previstas para cada exemplo é de 40,2; o que resulta um erro esperado médio de $\approx 36\%$ $(1 - 25{,}62/40{,}2)$ na medida *precision*. Para $\delta_{AF} = 1{,}0$ e $\delta_{FT} = 0{,}9$ com 14,7 classes preditas para cada exemplo, o erro esperado é de $\approx 42\%$ $(1 - 14{,}6/25{,}62)$, para a medida *recall*.

5.1.4 SENSIBILIDADE DO PARÂMETRO β_{FT} NO MHCAIS GLOBAL

A Subseção 5.1.2 trouxe resultados experimentais demonstrando que o MHCAIS apresenta melhor desempenho quando usada a medida *f-measure* como função de *fitness*. Nesta subseção são apresentados resultados de experimentos computacionais que permitem analisar a sensibilidade ou influência do parâmetro β_{FT} no desempenho do MHCAIS-global quando usada a medida *f-measure* para calcular o *fitness* das regras descobertas. Conforme detalhado na Sub-

seção 3.2.5, o valor de β_{FT} varia de 0 até ∞ e tem como objetivo especificar um peso maior para *precision* ou *recall* na função *f-measure*. Assim, para:

- $\beta_{FT} = 1$: *precision* e *recall* têm mesmo peso;
- $0 \leq \beta_{FT} < 1$: *precision* tem maior peso;
- $1 < \beta_{FT}$: *recall* tem maior peso;

As Tabelas 13 e 14 apresentam os resultados obtidos para os valores de β_{FT} testados. São apresentados os valores de *precision*, *recall*, *f-measure*, número de regras (#Regras), total de condições (#TTCond) e o número médio de classes preditas por exemplo (#PredExemp). Os melhores resultados entre os valores de β_{FT} testados são destacados em negrito. O desvio padrão é representado por $\pm$ e o melhor (pior) resultado de *f-measure* obtido por $\oplus$ ($\ominus$).

Os valores analisados do parâmetro usado na função *f-measure* são: $\beta_{FT} = \{0{,}05; 0{,}2; 0{,}4; 0{,}6; 1{,}0; 80\}$. Os experimentos foram executados para $\delta_{AF} = \{0{,}8; 1{,}0\}$. Como valores de parâmetros foram usados os mesmos apresentados na Tabela 8. Para $\delta_{AF} = 0{,}8$ foram testados os valores de $\delta_{FT} = \{0{,}8; 0{,}9\}$; esta escolha foi feita pelo fato que nos resultados apresentados na Tabela 11 a combinação de valores $\delta_{AF} = 0{,}8$ e $\delta_{FT} = 0{,}9$ obteve o melhor valor de *precision* dentre todos os resultados, enquanto que $\delta_{AF} = 0{,}8$ e $\delta_{FT} = 0{,}8$ apresentou o melhor valor de *f-measure*. A combinação de valores de parâmetros $\delta_{AF} = 1{,}0$ e $\delta_{FT} = 0{,}9$ também foi avaliada. Embora esta combinação tenha apresentado o pior resultado, conforme Tabela 11, o objetivo foi analisar se o parâmetro β_{FT} tem significativa influência no comportamento do algoritmo para melhorar sensivelmente um resultado considerado ruim.

No critério predição preditiva (Tabela 13), observa-se que para valores de $\beta_{FT} < 1$ e próximo a 0, o MHCAIS obtém significativa melhora para as medidas de *precision* e *recall* e por consequencia essa melhora se reflete na medida *f-measure*. Para o único valor de $\beta_{FT} > 1$ experimentado, o MHCAIS apresenta significativa melhora de *recall*, porém uma acentuada queda da medida *precision*, resultando nos piores desempenhos de *f-measure* mostrados na Tabela 13.

Com relação ao critério simplicidade das regras (Tabela 14), nota-se que para valores de $\beta_{FT} < 1$, em geral, mais regras são descobertas e por consequência mais condições aparecem no conjunto de regras. Este aumento do número de regras, possivelmente, é explicado haja vista que o MHCIAS para descobrir regras mais precisas diminui a cobertura dos exemplos classificados no treinamento. Lembrando que, para cada regra descoberta no treinamento, exemplos corretamente classificados por esta regra são removidos da base de treinamento. Para $\beta_{FT} = 80$, privilegiando a medida *recall* no treinamento, observa-se um conjunto de regras mais simplificado, todavia comprometendo significantemente a correção preditiva do classificador. Isto se

Tabela 13: Precisão obtida para diferentes valores do parâmetro β_{FT}.

δ_{AF}	δ_{FT}	β_{FT}	*Precision*	*Recall*	*F-Measure*
0,8	0,8	0,05	87,51 ± 0,5	82,67 ± 0,2	**80,51** ± 0,4
		0,2	**88,49** ± 0,6	79,66 ± 0,6	79,92 ± 0,6
		0,4	88,41 ± 0,3	76,29 ± 0,5	78,42 ± 0,3
		0,6	86,83 ± 0,5	77,58 ± 0,6	78,21 ± 0,4
		1,0	86,90 ± 0,7	76,53 ± 1,0	77,60 ± 0,4
		80	13,70 ± 0,5	**99,92** ± 0,0	23,68 ± 0,7
	0,9	0,05	**95,36** ± 0,35	79,89 ± 0,4	⊕**83,93** ± 0,4
		0,2	94,25 ± 0,4	72,07 ± 1,0	78,71 ± 0,6
		0,4	93,95 ± 0,4	71,83 ± 1,2	78,47 ± 0,7
		0,6	92,91 ± 0,2	71,10 ± 1,0	77,70 ± 0,7
		1,0	89,76 ± 0,6	70,68 ± 1,1	75,79 ± 0,9
		80	14,55 ± 0,7	**99,83** ± 0,0	24,80 ± 1,0
1,0	0,9	0,05	**96,17** ± 0,2	77,44 ± 0,2	**82,92** ± 0,1
		0,2	93,87 ± 0,2	68,61 ± 0,2	76,37 ± 0,1
		0,4	92,60 ± 0,3	66,65 ± 0,4	74,67 ± 0,3
		0,6	91,92 ± 0,4	57,34 ± 1,2	66,62 ± 0,9
		1,0	89,04 ± 0,4	58,12 ± 0,9	66,38 ± 0,6
		80	13,21 ± 0,9	**96,45** ± 3,4	⊖22,55 ± 0,9

Tabela 14: Simplicidade das regras para valores de β_{FT}.

δ_{AF}	δ_{FT}	β_{FT}	#Regras	#TTCond	#PredExemp
0,8	0,8	0,05	138,9 ± 3,4	1381,0 ± 40,6	23,6 ± 0,2
		0,2	49,4 ± 1,8	408,3 ± 15,6	22,0 ± 0,2
		0,4	35,6 ± 1,0	265,4 ± 14,0	20,8 ± 0,2
		0,6	29,2 ± 0,8	204,0 ± 6,4	21,7 ± 0,3
		1,0	24,9 ± 0,4	**162,9** ± 7,4	**21,4** ± 0,4
		80	**21,0** ± 1,3	223,8 ± 15,8	188,7 ± 7,1
	0,9	0,05	104,3 ± 3,6	909,9 ± 24,2	20,5 ± 0,1
		0,2	34,5 ± 1,5	244,2 ± 12,3	18,5 ± 0,3
		0,4	26,1 ± 0,9	152,7 ± 10,2	18,4 ± 0,4
		0,6	20,9 ± 0,5	120,4 ± 5,2	**18,3** ± 0,4
		1,0	**17,1** ± 0,8	**110,2** ± 6,8	18,8 ± 0,4
		80	24,9 ± 1,2	243,8 ± 18,5	183,4 ± 7,8
1,0	0,9	0,05	46,9 ± 1,0	122,6 ± 5,4	19,7 ± 0,1
		0,2	28,9 ± 0,7	68,7 ± 9,3	17,5 ± 0,0
		0,4	22,9 ± 0,6	43,9 ± 4,6	17,1 ± 0,1
		0,6	15,5 ± 0,7	22,6 ± 1,9	14,9 ± 0,4
		1,0	11,8 ± 0,5	25,4 ± 3,8	**14,7** ± 0,3
		80	**5,8** ± 0,5	**11,8** ± 2,2	196,3 ± 12,7

deve ao fato que o classificador construído tem maior cobertura das classes preditas para cada exemplo (*#PredExemp*). A Tabela 14 que o mostra número de classes preditas para cada exemplo de teste é significatemente maior para $\beta_{FT} = 80$. Conforme discutido em seções anteriores, maior número de classes preditas tende a melhorar o *recall* porém piora o *precision*. É possivel observar este comportamento para o conjunto de parâmetros $\beta_{FT} = 80$, $\delta_{FT} = 0{,}9$ e $\delta_{AF} = 1{,}0$, onde o classificador construído tem um número médio de 196,3 classes predita por exemplo, onde o universo total de classes do domínio da base utilizada nos experimentos contém 214 classes e uma média de 25 classes por exemplo pré-classificado, é fácil notar que o valor de *recall* será alto e o de *precision* baixo.

5.1.5 INFLUÊNCIA DA BUSCA LOCAL E PODA NO MHCAIS

Os procedimentos de busca local e poda são mecanismos implementados para melhorar a qualidade e a simplicidade das regras descobertas e são detalhados nas Subseções 3.2.6 e 3.2.7. Embora ambos os procedimento sejam considerados técnicas de busca local, cada um atua com um objetivo distinto. O mecanismo de poda tem como função tornar as regras descobertas mais simples pela remoção de condições irrelevantes enquanto o *fitness* estiver sendo melhorado. O procedimento chamado efetivamente de busca local atua especificamente alterando valores das condições do antecedente da regra em busca de novos valores que melhorem o *fitness* da regra.

Nesta subseção são apresentados os resultados de experimentos computacionais que permitem avaliar e justificar o uso dos mecanismos de poda e busca local no algoritmo proposto nesta tese. Os experimentos foram realizados para as seguintes versões:

- MHCAIS-$<$*Versão*$>_{padrao}$: sem busca local e poda;
- MHCAIS-$<$*Versão*$>_{busloc}$: apenas busca local;
- MHCAIS-$<$*Versão*$>_{poda}$: apenas poda; e
- MHCAIS-$<$*Versão*$>$: com busca local e poda.

onde $<$*Versão*$>$ especifica o algoritmo MHCAIS-global ou MHCAIS-local.

A Tabela 15 mostra os resultados de correção preditiva e simplicidade do conhecimento descoberto, acompanhados dos respectivos desvio-padrão, para as seguintes versões: MHCAIS-Global$_{padrao}$, MHCAIS-Global$_{busloc}$, MHCAIS-Global$_{poda}$, MHCAIS-Global. Os melhores resultados são destacados em negrito. O símbolo $\oplus$ indica que MHCAIS-Global com Poda e Busca local é estatisticamente superior ao MHCAIS-Global padrão (sem os dois procedimentos), considerando a sobreposição dos desvios-padrão. Nos experimentos foram considerados

matching parcial e total (δ_{AF}). Os valores dos parâmetros são os mesmos apresentados na Tabela 8 com as seguintes particularidades: $\delta_{FT} = 0{,}9$ e $\beta_{FT} = 0{,}05$.

Tabela 15: Desempenho do MHCAIS-global com uso de busca local e poda.

MHCAIS-Global$_{padrao}$					
	Correção Preditiva			Simplicidade	
δ_{AF}	*Precision*	*Recall*	*F-Measure*	#Regras	#TTCond
0,8	$92{,}60 \pm 0{,}5$	$79{,}02 \pm 0{,}4$	$81{,}75 \pm 0{,}4$	$111{,}2 \pm 5{,}6$	$2012{,}3 \pm 106{,}2$
0,9	$95{,}10 \pm 0{,}4$	$76{,}80 \pm 0{,}4$	$81{,}89 \pm 0{,}2$	$63{,}1 \pm 1{,}1$	$1031{,}6 \pm 19{,}4$
1,0	$77{,}07 \pm 8{,}8$	$49{,}19 \pm 6{,}0$	$56{,}70 \pm 6{,}7$	**21,2** $\pm 1{,}9$	$294{,}1 \pm 26{,}8$
MHCAIS-Global$_{busloc}$					
0,8	$94{,}25 \pm 0{,}3$	$79{,}09 \pm 0{,}3$	$82{,}88 \pm 0{,}3$	$112{,}8 \pm 4{,}0$	$2039{,}8 \pm 75{,}9$
0,9	$95{,}08 \pm 0{,}5$	$76{,}95 \pm 0{,}5$	$82{,}01 \pm 0{,}4$	$61{,}6 \pm 2{,}0$	$1018{,}3 \pm 35{,}2$
1,0	$83{,}92 \pm 7{,}9$	$55{,}30 \pm 5{,}9$	$62{,}54 \pm 6{,}4$	$25{,}1 \pm 1{,}9$	$358{,}0 \pm 27{,}4$
MHCAIS-Global$_{poda}$					
0,8	$94{,}62 \pm 0{,}4$	$78{,}91 \pm 0{,}9$	$82{,}73 \pm 0{,}6$	$107{,}1 \pm 2{,}4$	$960{,}9 \pm 29{,}1$
0,9	$95{,}29 \pm 0{,}4$	**78,05** $\pm 0{,}3$	$82{,}71 \pm 0{,}3$	**61,3** $\pm 1{,}9$	$410{,}3 \pm 16{,}7$
1,0	**95,79** $\pm 0{,}2$	**77,62** $\pm 0{,}3$	$82{,}86 \pm 0{,}1$	$47{,}3 \pm 1{,}2$	**114,7** $\pm 9{,}0$
MHCAIS-Global					
0,8	$95{,}36 \pm 0{,}3$	**79,89** $\pm 0{,}4$	$\oplus$**83,93** $\pm 0{,}4$	**104,3** $\pm 3{,}6$	$\oplus$**909,9** $\pm 24{,}2$
0,9	**96,58** $\pm 0{,}4$	$77{,}86 \pm 0{,}3$	$\oplus$**83,41** $\pm 0{,}3$	$64{,}7 \pm 2{,}2$	$\oplus$**409,6** $\pm 17{,}6$
1,0	**96,12** $\pm 0{,}1$	$77{,}39 \pm 0{,}3$	$\oplus$**82,87** $\pm 0{,}1$	$47{,}3 \pm 0{,}6$	$\oplus 115{,}5 \pm 6{,}6$

Conforme os resultados apresentados na Tabela 15, nota-se uma esperada melhora de desempenho, no critério de correção preditiva, com o uso da versão MHCAIS-Global$_{busloc}$ comparado ao MHCAIS-Global$_{padrao}$, porém sem simplificação do conhecimento descoberto. Conforme mencionado anteriormente, o procedimento de busca local encontra diferentes valores para condições da regra que melhore seu *fitness*, sem o compromisso torná-la mais simples. Neste sentido, o procedimento de poda (MHCAIS-Global$_{poda}$) descobriu regras signifivamente mais simples comparadas ao MHCAIS-Global$_{padrao}$, embora para $\delta_{AF} = 1{,}0$ mais regras tenham sido descobertas com o uso deste procedimento. É importante observar que além de regras mais simples (objetivo principal deste procedimento), o MHCAIS-Global$_{poda}$ apresentou significativa melhora da correção preditiva. Agregando os mecanismos de poda e busca local, o MHCAIS apresentou resultados ainda mais significativos comparados ao MHCAIS-Global$_{padrao}$. A Tabela 15 mostra também que as melhoras mais expressivas foram obtidas para $\delta_{AF} = 1{,}0$, haja vista a dificuldade apresentada pelo MHCAIS-Global$_{padrao}$ no critério de correção preditiva.

A Tabela 16 mostra os resultados de correção preditiva e simplicidade do conhecimento descoberto, acompanhados dos respectivos desvio-padrão, para as seguintes versões: MHCAIS-Local$_{padrao}$ e MHCAIS-Local. Os mecanismos de poda e busca local não foram testados isoladamente, tendo em vista que a dinâmica de construção das regras no MHCAIS-local, durante

o período evolutivo, é similar ao MHCAIS-global. Os melhores resultados são destacados em negrito. Nos experimentos foram considerados *matching* parcial e total (δ_{AF}). Os valores dos parâmetros são os mesmos apresentados na Tabela 8. O símbolo $\oplus$ indica que MHCAIS-Local com Poda e Busca local é estatisticamente superior ao MHCAIS-Local padrão (sem os dois procedimentos), considerando a sobreposição dos desvios-padrão.

Tabela 16: Desempenho do MHCAIS-local com uso de busca local e poda.

MHCAIS-Local$_{padrao}$					
	Correção Preditiva			Simplicidade	
δ_{AF}	*Precision*	*Recall*	*F-Measure*	#Regras	#TTCond
0,8	$90{,}97 \pm 0{,}2$	$86{,}97 \pm 0{,}2$	$87{,}82 \pm 0{,}2$	**725,3** $\pm 5{,}6$	$12910{,}0 \pm 106{,}3$
0,9	$84{,}93 \pm 0{,}5$	$81{,}44 \pm 0{,}4$	$81{,}74 \pm 0{,}3$	$793{,}8 \pm 7{,}5$	$12731{,}4 \pm 138{,}0$
1,0	$73{,}64 \pm 0{,}7$	$78{,}29 \pm 0{,}4$	$73{,}63 \pm 0{,}5$	$1582{,}4 \pm 25{,}7$	$20942{,}6 \pm 376{,}8$
MHCAIS-Local					
0,8	$90{,}91 \pm 0{,}2$	$87{,}32 \pm 0{,}3$	**87,96** $\pm 0{,}2$	$738{,}3 \pm 4{,}4$	$\oplus$**8394,7** $\pm 59{,}8$
0,9	$89{,}69 \pm 0{,}3$	$87{,}13 \pm 0{,}4$	$\oplus$**87,27** $\pm 0{,}3$	$\oplus$**752,5** $\pm 4{,}4$	$\oplus$**7165,3** $\pm 82{,}9$
1,0	$84{,}64 \pm 0{,}2$	$87{,}09 \pm 0{,}4$	$\oplus$**84,63** $\pm 0{,}5$	$\oplus$**734,9** $\pm 7{,}3$	$\oplus$**4610,3** $\pm 51{,}2$

A versão local do MHCAIS seguiu a mesma melhora apresentada pela versão global. Conforme a Tabela 16, o MHCAIS-Local apresentou significativa melhora de desempenho preditivo para $\delta_{AF} = \{0{,}9; 1{,}0\}$ com o uso de busca local e poda, comparado ao MHCAIS-Local$_{padrao}$. Embora MHCAIS-Local tenha obtido melhor *f-measure* para $\delta_{AF} = 0{,}8$, considerando o desvio-padrão, a diferenção não é estatisticamente superior ao MHCAIS-Local$_{padrao}$. No critério simplicidade do conhecimento descoberto, novamente a utilização de poda e busca local simplificou consideravelmente o número de condições do conjunto e número de regras descobertas. A única exceção ficou por conta do maior número de regras para $\delta_{AF} = 0{,}8$ do MHCAIS-Local em comparação MHCAIS-Local$_{padrao}$, embora o número de condiçõe tenha sido menor.

5.1.6 COMPARANDO O DESEMPENHO DO MHCAIS COM *MATCHING* PARCIAL E TOTAL

O algoritmo MHCAIS em suas versões global e local apresenta como uma de suas características a possilidade do usuário optar pela construção de um modelo de predição usando *matching* parcial ou total (conforme discutido na Subseção 3.2.4). Os valores utilizados nos experimentos envolvendo *matching* parcial são: $\delta_{AF} = \{0{,}8; 0{,}9\}$. O valor $\delta_{AF} = 1{,}0$ representa *matching* total.

A Tabela 17 apresenta os resultados obtidos pelo algoritmo MHCAIS nas versões global e local para os valores de *matching* (δ_{AF}) anteriormente citados. Nesta tabela são mostrados os valores de *precision*, *recall* e *f-measure* obtidos, bem como o número de regras descobertas

(#Regras) e o total de condições do conjunto de regras descobertas(#TTCond). O símbolo $\pm$ representa o desvio padrão dos resultados obtidos nos experimentos. Os valores dos parâmetros são apresentados na Tabela 8 com as seguintes particularidades: $\beta_{FT} = 0{,}05$ e $\delta_{FT} = 0{,}9$ na versão global; e $\beta_{FT} = 1$ na versão local. Os melhores resultados entre os valores de δ_{AF} são destacados em negrito

Tabela 17: Resultados obtidos pelo MHCAIS para o parâmetro δ_{AF}.

Global					
	Correção Preditiva			Simplicidade	
δ_{AF}	*Precision*	*Recall*	*F-measure*	#Regras	#TTCond
0,8	**95,36** $\pm$ 0,3	**79,89** $\pm$ 0,4	**83,93** $\pm$ 0,4	104,3 $\pm$ 3,6	909,9 $\pm$ 24,2
0,9	96,58 $\pm$ 0,4	77,86 $\pm$ 0,3	83,41 $\pm$ 0,3	64,7 $\pm$ 2,2	409,6 $\pm$ 17,6
1,0	96,17 $\pm$ 0,2	77,44 $\pm$ 0,2	82,92 $\pm$ 0,1	**46,9** $\pm$ 1,0	**122,6** $\pm$ 5,4
Local					
0,8	**90,91** $\pm$ 0,2	**87,32** $\pm$ 0,3	**87,96** $\pm$ 0,2	738,3 $\pm$ 4,4	8394,7 $\pm$ 59,8
0,9	89,69 $\pm$ 0,3	87,13 $\pm$ 0,4	87,27 $\pm$ 0,3	752,5 $\pm$ 4,4	7165,3 $\pm$ 82,9
1,0	84,64 $\pm$ 0,2	87,09 $\pm$ 0,4	84,63 $\pm$ 0,5	**734,9** $\pm$ 7,3	**4610,3** $\pm$ 51,2

A Tabela 17 mostra que para ambas as versões do MHCAIS, com o relaxamento do valor de δ_{AF} os resultados de correção preditiva, considerando a medida *f-measure*, são ligeiramente melhores do que com o uso de *matching* total (δ_{AF}=1,0), embora estatisticamente a diferença não seja significativa, exceção feita ao resultado apresentado pelo MHCAIS-local para $\delta_{AF=1,0}$.

No que diz respeito à simplicidade do conhecimento descoberto, para a versão global o uso do *matching* total resultou em conjunto de regras e condições significativamente menor do que a utilização do *matching* parcial. Na versão local, a diferença mais significativa ficou por conta do menor número de condições do conjunto de regras obtido com o uso do *matching* total comparado ao *matching* parcial. A diferença na simplicidade das regras descobertas com o uso do *matching* total era esperada, pois para algoritmo evoluir e descobrir regras durante o treintamento é necessário que todas as condições presentes no antecedente da regra satisfaçam os valores dos atributos previsores dos exemplos, o que torna mais complexo este processo quando regras possuem muitas condições. Sendo assim, o algoritmo é estimulado a descobrir regras o mais "simples" possível durante o treinamento para conseguir classificar os exemplos da base.

Conforme esperado, para a versão global, à medida que se torna mais restritivo o *matching*, o valor de *precision* melhora, enquanto que o valor de *recall* diminui. Embora o *precision* ($\delta_{AF} = 0{,}9$) seja maior que o valor de *precision* ($\delta_{AF} = 1{,}0$), essa diferença não é significativa considerando o desvio padrão. Todavia, este esperado comportamento de *precision* e *recall* não se repete na versão local, pois com o aumento do valor de δ_{AF}, o valor de *precision* diminui,

embora o valor de *recall* permaneça praticamente constante. Possivelmente, a queda do valor de *precision* na versão local, seja explicado pelo menor número de condições do conjunto de regras.

Embora o objetivo dos resultados apresentados na Tabela 17 não seja a comparação entre as versões propostas do MHCAIS, é importante notar que a versão local obtém melhor desempenho de correção preditiva, enquanto que a versão global obtém como conhecimento um resultado significativamente mais simples, segundo os critérios de avaliação adotados neste conjunto de experimentos.

5.1.7 CONSIDERAÇÕES

Nesta seção foram apresentados resultados para análise da influência e sensibilidade de desempenho do MHCAIS local e global à variação dos valores de seus parâmetros e procedimentos.

A influência do número de iterações do período evolutivo ($\#MaxIter$) foi analisada na Subseção 5.1.1. Os valores analisados foram $\#MaxIter = \{50; 100\}$. Os resultados mostraram que o ganho de desempenho para 100 iterações não foi significativo comparado à 50 iterações.

A sensibilidade do MHCAIS quanto a função de *fitness* usada no treinamento do classificador foi analisada na Subseção 5.1.2. Foram avaliadas duas funções de *fitness*: sensibilidade/especificicade e *f-measure*. Nos experimentos realizados o uso da função de *fitness f-measure* obteve significativo melhor desempenho preditivo para a versão local e global do MHCAIS.

A Subseção 5.1.3 analisou a influência do parâmetro δ_{FT} (presente apenas no MHCAIS global). O valores testados foram $\delta_{FT} = \{0{,}5; 0{,}6; 0{,}7; 0{,}8; 0{,}9\}$. No critério de desempenho preditivo, os valores mais altos de δ_{FT} apresentam melhores resultados para *precision* e piores para *recall*, para valores mais baixo de δ_{FT} o contrário acontece. O valor de δ_{FT} que apresentou melhor compromisso com a medida *f-measure* foi $\delta_{FT} = 0{,}8$

Na Subseção 5.1.4 foi avaliado o parâmetro β_{FT} utilizado na função de *fitness f-measure*. Os valores testados foram $\beta_{FT} = \{0{,}05; 0{,}2; 0{,}4; 0{,}6; 1{,}0; 80\}$, onde $\beta_{FT} < 1{,}0$ mais peso no *fitness* tem a media *precision*; e $\beta_{FT} > 1{,}0$ o maior peso fica por conta do *recall*. Os resultados mostraram que os melhores resultados de *f-measure* sobre os dados de teste foram obtidos para $\beta_{FT} = 0{,}05$, e os piores para $\beta_{FT} = 80$. Além disso, o melhor resultado obtido foi para $\beta_{FT} = 0{,}05$ e $\delta_{FT} = 0{,}9$.

A sensibilidade do MHCAIS global e local ao uso dos procedimentos de buscal local e poda

foram analisados na Subseção 5.1.5. Nos experimentos foi observado que o uso conjunto dos dois procedimentos melhoram o desempenho preditivo das duas versões do MHCAIS, porém a melhora significativa ficou por conta da simplicidade do conhecimento descoberto, onde o número de regras e condições do conjunto de regras foi significativamente reduzido.

A Susbseção 5.1.6 apresentou resultados para análise da influência do parâmetro δ_{AF} que específica o nível de afinidade ou *matching* da regra com o exemplo. Os valores testados foram $\delta_{AF} = \{0{,}8; 0{,}9; 1{,}0\}$. O valor $\delta_{AF} = 1{,}0$ representa *matching* total, enquanto $\delta_{AF} < 1{,}0$ indica *matching* parcial. Os resultados mostraram que para o critério de simplicidade das regras $\delta_{AF} = 1{,}0$ obteve os melhores resultados, enquanto $\delta_{AF} = 0{,}8$ apresentou melhor desempenho preditivo para a medida *f-measure*.

5.2 COMPARATIVO ENTRE OS ALGORITMOS MHCAIS E PART

Nesta seção resultados comparativos de desempenho nos experimentos computacionais entre o MHCAIS e PART são apresentados. Conforme mencionado anteriormente, o PART é um algoritmo que constrói classificadores locais, tal qual o MHCAIS-local, baseados em regras obtidas a partir de árvores de decisão (detalhes sobre AD em B.1). Para obtenção dos resultados foi usada nos experimetos a validação cruzada (Seção 4.4) e a base de dados ATP+DNAb (Seção 4.2), chamada nesta seção de ATP+DNAb-1 (mesma base usada na seção anterior) pela divisão em 10 subconjuntos para uso na validação cruzada. Os valores dos parâmetros utilizados para execução das versões global e local do MHCAIS são os mesmos apresentados na Tabela 8 com $\delta_{FT} = 0{,}9$ e $\beta_{FT} = 0{,}05$. Os parâmetros do algoritmo PART respeitam os valores padrões disponíveis na ferramenta Weka.

A Tabela 18 apresenta os resultados obtidos dos algoritmos comparados. Nela são mostrados os valores de *precision*, *recall*, *f-measure*, número de regras descobertas (#Regras) e total de condições do conjunto de regras (#TTCond). São apresentados resultados do MHCAIS para *matching* parcial e total (δ_{AF}). O algoritmo PART não permite *matching* parcial, apenas total. Os símbolos $\pm$ representa o desvio padrão dos resultados, $\oplus(\ominus)$ indica que o MHCAIS (independentemente da versão) obteve melhor (pior) resultado que o PART. Os melhores (piores) resultados, considerando o critério estatístico de sobreposição do desvio-padrão, são representados pelo símbolo $\blacktriangle(\triangledown)$.

Na Tabela 18 pode-se observar que o MHCAIS-local foi estatisticamente superior ao PART, considerando o desvio padrão, em praticamente todas as medidas de desempenho consideradas, exceção ao número total de condições do conjunto de regras descobertas. De forma geral, o MHCAIS-local obteve melhores resultados com menos regras. O menor número total de

Tabela 18: Resultados comparativos entre MHCAIS e PART.

Global					
	Correção Preditiva			Simplicidade	
δ_{AF}	*Precision*	*Recall*	*F-Measure*	#Regras	#TTCond
0,8	▲95,36 ± 0,3	▽79,89 ± 0,4	▲83,93 ± 0,4	▲104,3 ± 3,6	▲909,9 ± 24,2
0,9	▲96,58 ± 0,4	▽77,86 ± 0,3	⊕83,41 ± 0,3	▲64,7 ± 2,2	▲409,6 ± 17,6
1,0	▲96,17 ± 0,2	▽77,44 ± 0,2	⊕82,92 ± 0,1	▲46,9 ± 1,0	▲122,60 ± 5,46
Local					
0,8	▲90,91 ± 0,2	▲87,32 ± 0,3	▲87,96 ± 0,2	▲738,3 ± 4,4	▽8394,7 ± 59,8
0,9	▲89,69 ± 0,3	▲87,13 ± 0,4	▲87,27 ± 0,3	▲752,5 ± 4,4	▽7165,3 ± 82,9
1,0	▲84,64 ± 0,2	▲87,09 ± 0,4	▲84,63 ± 0,5	▲734,9 ± 7,3	▽4610,3 ± 51,2
PART - Weka					
	83,08 ± 0,6	81,85 ± 0,6	82,78 ± 0,5	4759,3 ± 12,6	1820,6 ± 6,8

condições do conjunto de regras obtidos pelo PART se deve ao fato que, para muitas classes o algoritmo encontrou apenas a regra padrão (*default*), isso também explica porque o número de regras foi maior que o número total de condições do conjunto de regras descobertas, conforme apresenta a última linha da Tabela 18. A regra padrão representa uma regra sem antecedente prevendo a classe mais frequente na base de dados.

Por outra lado, o algoritmo MHCAIS-global foi estatiscamente superior ao PART nas seguintes medidas de desempenho: *precision*, número de regras descobertas e total de condições do conjunto de regras para todos os experimentos variando o valor de δ_{AF}; e *f-measure* para $\delta_{AF} = 0{,}8$. Para os outros valores δ_{AF}, observando a medida *f-measure*, o MHCAIS-global obteve melhor resultado, porém não pode ser considerado estatisticamente superior. No tocante à medida *recall*, o PART foi estatisticamente melhor que o MHCAIS-global em todos os experimentos. De forma geral, em comparação ao PART, nota-se que o MHCAIS global erra menos previsões (*precision*), porém deixa de prever classes que deveriam ser preditas.

5.2.1 ANÁLISE ESTATÍSTICA CONSIDERANDO TESTE DE WILCOXON

Embora a Seção 5.2 compare os algoritmos MHCAIS Global/Local com o PART considerando o desvio-padrão, testes estatísticos de hipóteses permitem uma análise mais aprofundada e confiável quando se deseja apontar o algoritmo com melhor desempenho. Neste sentido, esta subseção apresenta análise estatística usando o teste de hipóteses de Wilcoxon, descrito na Seção A.1 do Apêndice A, para comparar o desempenho dos algoritmos acima mencionados, considerando apenas o critério de correção preditiva (*f-measure*).

A Tabela 19 mostra os resultados estatísticos obtidos. Neste tabela, os algoritmos são comparados dois a dois, haja vista que o teste de Wilcoxon é sugerido para comparação entre dois

algoritmos. A Tabela 19 apresenta os valores de *p-values*, obtidos com auxílio do *software* Matlab 7.1, e a confirmação (F) ou rejeição da hipótese nula (V), com 95% de nível de significância. É importante lembrar, que a rejeição da hipótese nula, indica que a diferença de desempenho entre algoritmos comparados é significativa.

Tabela 19: Teste de Hipóteses de Wilcoxon com postos de sinais comparando MHCAIS-Global/Local vs PART e Local vs Global.

	Global vs PART		Local vs PART		Local vs Global	
δ_{AF}	*p-value*	h	*p-value*	h	*p-value*	h
0,8	0,0195	V	0,0020	V	0,0020	V
0,9	0,1055	F	0,0020	V	0,0020	V
1,0	0,3223	F	0,0059	V	0,0137	V

Os resultados do teste de hipóteses de Wilcoxon apresentados na Tabela 19 confirmam os resultados e análises realizadas, considerando o desvio padrão na Seção 5.2, onde o MHCAIS-Global é significativamente superior ao PART com o uso de $\delta_{AF} = 0{,}8$ e MHCAIS-Local é estatisticamente superior ao PART com todos os valores de δ_{AF} avaliados. O teste de hipótese confirmou a análise usando o desvio-padrão para a não diferença estatística de desempenho entre o MHCAIS-Global $\delta_{AF} = 0{,}9$ vs PART e MHCAIS-Global $\delta_{AF} = 1{,}0$ vs PART.

A Tabela 19 compara através do teste de Wilcoxon o desempenho de ambas as versões do MHCAIS, onde é mostrado que o MHCAIS-Local é estatisticamente superior ao MHCAIS-global no conjunto de experimentos realizados, considerando o critério de correção preditiva.

5.3 COMPARATIVO ENTRE OS ALGORITMOS MHCAIS E CLUS

Nesta seção são apresentados resultados comparativos entre os algoritmos MHCAIS global, local e o Clus. Os critérios considerados para comparação são: área da curva de *precision-recall*, número de regras descobertas e número total de condições presentes no conjunto de regras. Estes critérios são descritos na Seção 4.4. Os testes de Friedman e Wilcoxon são usados para análise estatística dos resultados e são descritos nas Seções A.2 e A.1, respectivamente, do Apêndice A. Cada base de dados avaliada foi dividida em 2/3 para etapa de treinamento e outros 1/3 para etapa teste. As bases utilizadas nos experimentos são descritas na Seção 4.2, totalizando 10 conjuntos de dados. Nesta seção a base ATP+DNAb foi dividida, aleatoriamente, em 2 partições sendo 2/3 para treinamento e 1/3 para teste, e portanto é ligeiramente diferente da base usada na Seção 5.2. Deste modo, a base ATP+DNAb será chamada nesta seção de ATP+DNAb-2

5.3.1 ÁREAS DA CURVA *PRECISION-RECALL* OBTIDAS NOS EXPERIMENTOS

Nesta subseção são apresentadas e analisadas as áreas das curvas com base nas medidas *precision-recall* (AUPRC) obtidas pelos algoritmos comparados. A Tabela 20 mostra as áreas de curva obtidas pelos algoritmos MHCAIS global/local e o Clus sobre as bases avaliadas, cujos melhores valores de AUPRC (para cada base) é destacado em negrito. Para os algoritmos MHC-AIS global e local foram considerados os valores de *matching* $\delta_{AF} = \{0{,}8; 0{,}9; 1{,}0\}$. As curvas de *precision-recall* obtidas nos experimentos são apresentadas graficamente no Apêndie E.

Tabela 20: Resultados de AUPRC obtidos pelos algoritmos MHCAIS-Global/Local e Clus.

	MHCAIS-Global			MHCAIS-Local			Clus
Base de Dados	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	
ATP+DNAb-2	0,9096	0,9115	0,9097	0,8627	0,8406	0,8293	**0,9821**
CellCycle	0,1884	0,1879	0,1843	0,1695	0,1580	0,1764	**0,1975**
Church	0,1855	0,1897	0,2005	0,1926	0,1767	0,1693	**0,2192**
Derisi	0,2034	0,2000	0,1933	0,1888	0,1769	0,1598	**0,2270**
Eisen	0,2127	0,2100	0,1972	0,2048	0,1945	0,2142	**0,2419**
Gasch1	0,2084	0,2084	0,2084	0,1587	0,1727	0,1840	**0,2408**
Gasch2	0,2033	0,2016	0,2058	0,1870	0,1670	0,1789	**0,2466**
Pheno	0,2351	0,2274	**0,2357**	0,2336	0,2051	0,2292	0,2350
Seq	0,1971	0,1971	0,1971	0,1357	0,1515	0,1570	**0,3309**
Spo	0,1988	0,1856	0,1835	0,1761	0,1744	0,1673	**0,2341**

Os resultados da Tabela 20 são analisados usando o teste de hipóteses de Friedman para verificar se há significância estatística entre as diferenças de desempenhos dos algoritmos. Para tal, os resultados da Tabela 20 são reapresentados na Tabela 21 considerando o posto (*ranking*) ordenado de cada algoritmo sobre cada base. Por fim, a Tabela 21 ainda mostra o posto médio dos algoritmos sobre as 10 bases avaliadas.

Tabela 21: Postos obtidos pelos algoritmos MHCAIS-Global/Local e Clus para AUPRC.

	MHCAIS-Global			MHCAIS-Local			Clus
Base de Dados	$\delta_{AF=0,8}$	$\delta_{AF=0,8}$	$\delta_{AF=1,0}$	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	
ATP+DNAb-2	4	2	3	5	6	7	1
CellCycle	2	3	4	6	7	5	1
Church	5	4	2	3	6	7	1
Derisi	2	3	4	5	6	7	1
Eisen	5	3	6	4	7	2	1
Gasch1	3	3	3	7	6	5	1
Gasch2	3	2	4	5	7	6	1
Pheno	2	6	1	4	7	5	3
Seq	3	3	3	7	6	5	1
Spo	2	3	4	5	6	7	1
Média dos postos	3,1	3,2	3,4	5,1	6,4	5,6	1,2

Aplicando os dados da Tabela 21 na Equação 61 para obtenção da estatística de Friedman, tem-se $\chi_F^2 = 41{,}1$. Em seguida, usa-se a estatística F_F para correção de χ_F^2, conforme Equação 62, chega-se ao valor de $F_F = 19{,}57$. Para os graus de liberdade $k-1$ e $(k-1)\times(n-1)$, onde $k = 7$ e $n = 10$, encontra-se o seguinte valor tabelado na distruição F de Snedecor, $F(6;54) = 2{,}19$. Logo, $F_F > F(6;54)$, e portanto a hipótese nula pode ser rejeitada, indicando que há diferença estatística entre os algoritmos comparados.

Rejeitada a hipótese nula, deve-se identificar quais algoritmos possuem diferenças significativas de desempenho considerando a distância crítica (CD), calculada através da Equação 63. Considerando o nível de significância de 95% obtem-se o seguinte valor $CD = 2{,}849$.

A Figura 25 ilustra graficamente os resultados obtidos de acordo com os testes de Friedman e Nemenyi. Neste gráfico os classificadores são posicionados num eixo horizontal de acordo com o valor do seu posto médio (Tabela 21), sendo que o melhor deles é posicionado à direita (Clus) e o pior à esquerda (MHCAIS-Local $\delta_{AF=0,9}$). Acima do eixo horizontal é mostrada a distância crítica (CD). Os classificadores cujos postos médios possuem distâncias menores que a distância crítica são conectados através de uma linha mais grossa formando grupos.

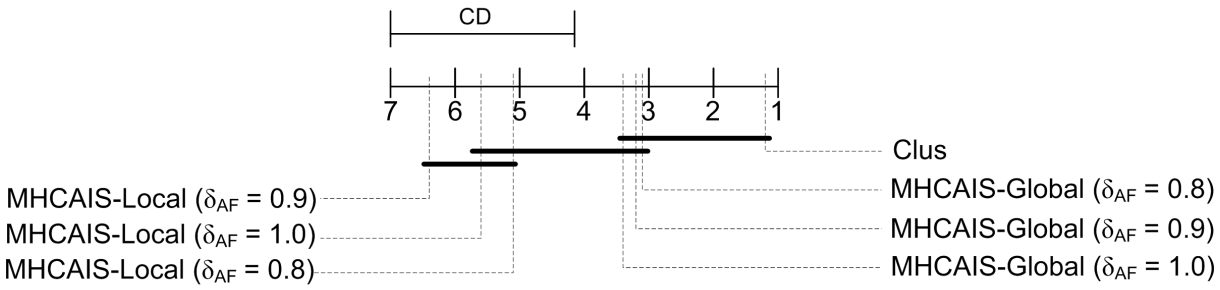


Figura 25: Classificadores ordenados pelos postos médios de acordo com resultados obtidos para AUPRC e agrupados pela distância crítica.

Na Figura 25 obsevam-se 3 grupos ligados por um linha grossa:

1. Clus, MHCAIS-Global $\delta_{AF} = 0{,}8$, $\delta_{AF} = 0{,}9$ e $\delta_{AF} = 1{,}0$.
2. MHCAIS-Global $\delta_{AF} = 0{,}8$, $\delta_{AF} = 0{,}9$ e $\delta_{AF} = 1{,}0$; MHCAIS-Local $\delta_{AF} = 0{,}8$ e $\delta_{AF} = 1{,}0$.
3. MHCAIS-Local $\delta_{AF} = 0{,}8$, $\delta_{AF} = 0{,}9$ e $\delta_{AF} = 1{,}0$.

De acordo com a representação gráfica da Figura 25 construída a partir dos testes de Friedman e Nemenyi pode-se afirmar que:

- o algoritmo Clus possui desempenho estatisticamente superior aos classificadores construídos pelo MHCAIS-Local para diferentes valores de δ_{AF};
- o MHCAIS-Global para todos os δ_{AF} é superior apenas ao MHCAIS-Local com $\delta_{AF} = 0{,}9$

Todavia, nenhuma outra afirmação pode ser feita, haja vista que alguns classifadores estão presentes em mais de um agrupamento, o que seria um contradição estatística, e portanto não pode determinar outras diferenças estatísticas além das mencionadas acima.

Conforme discutido anteriormente, teste de Friedman em conjunto com o teste Nemenyi mostram que não há diferença estatística entre o primeiro (Clus) e o segundo (MHCAIS-Global $\delta_{AF=0,8}$) colocados geral sobre as bases avaliadas. Porém, os dados apresentados nas Tabelas 20 e 21 chamam a atenção, haja vista que o Clus apresentou os melhores resultados de AUPRC e postos em 9 das 10 bases avaliadas. Portanto, é possível que os testes de Friedman e Nemenyi sejam sensíveis ao número de classificadores comparados ($k = 7$), muito embora o número de abordagens para construção dos classificadores seja diferente, neste caso, 3 algoritmos: MHCAIS-Global (3 classificadores), Local 3 (classificadores) e Clus (1 classificador). Deste modo, os algoritmos primeiro (Clus) e segundo (MHCAIS-Global $\delta_{AF=0,8}$) colocados, conforme ilustra a Figura 25, foram submetidos ao teste de hipóteses de Wilcoxon para nova análise estatística.

A Tabela 22 apresenta o resultado do teste de hipóteses de Wilcoxon comparando os algoritmos MHCAIS-Global $\delta_{AF=0,8}$ e o Clus. Na tabela é mostrado o *p-value* e a aceitação (F) ou não (V) da hipótese nula, para um nível de confiança de 95%.

Tabela 22: Teste de Hipóteses de Wilcoxon com postos de sinais comparando os resultados de AUPRC dos algoritmos MHCAIS-Global $\delta_{AF=0,8}$ (2^o) e Clus (1^o).

p-value	h
0,0039	V

O resultado da Tabela 22 mostra que, segundo o teste de Wilcoxon, há diferença estatística entre o desempenho dos algoritmos, apontando superioridade do Clus sobre o MHCAIS-Global $\delta_{AF=0,8}$. Com o resultado divergente entre os testes estatísticos analisados, aumentou-se a suspeita de que o teste de Friedman/Nemenyi seja sensível ao número de classificadores comparados. Deste modo, para analisar esta possível sensbilidade, nova análise estatística foi realizada.

Esta nova análise compara o desempenho do melhor classificador construído pelos algoritmos MHCAIS-global, local e Clus, de acordo com a Tabela 21.

A Tabela 23 apresenta o postos conseguidos pelos algoritmos em cada base e os postos médios.

Tabela 23: Postos obtidos pelos melhores classificadores construídos pelos algoritmos MHCAIS-Global, Local e Clus, de acordo com os dados da Tabela 21.

Base de Dados	MHCAIS-Global $\delta_{AF=0,8}$	MHCAIS-Local $\delta_{AF=0,9}$	Clus
ATP+DNAb-2	2	3	1
CellCycle	2	3	1
Church	3	2	1
Derisi	2	3	1
Eisen	3	2	1
Gasch1	2	3	1
Gasch2	2	3	1
Pheno	1	3	2
Seq	2	3	1
Spo	2	3	1
Média dos postos	2,1	2,8	1,1

Recalculando os valores para as equações utilizadas no teste de Frieman tem-se $\chi_F^2 = 14{,}6$ e $F_F = 24{,}33$. De acordo com o valor tabelado para os graus de liberdade 2 e 18 na distribuição F de Snedecor encontra-se o valor crítico de $F(2;18) = 3{,}55$. Deste modo a hipótese nula é rejeitada, $F_F > F(2;18)$. Com a hipótese nula rejeitada, calcula-se a distância crítica, assim $CD = 1{,}047$. A Figura 26 ilustra o resultado estatísco para o teste de Nemenyi para os algoritmos MHCAIS global, local e Clus.

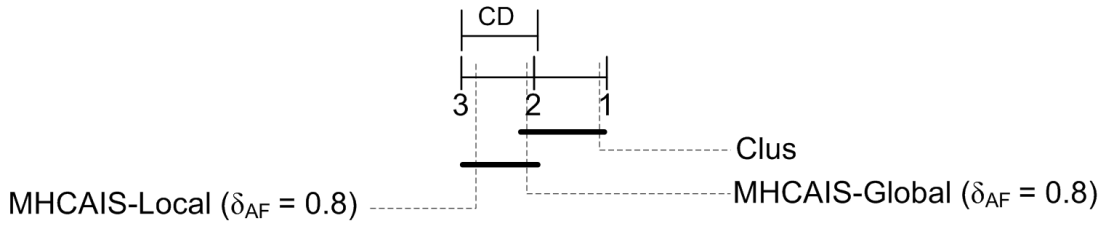


Figura 26: Melhores Classificadores do MHCAIS global, local e Clus (conforme Tabela 21) ordenados pelos postos médios de acordo com resultados obtidos para AUPRC e agrupados pela distância crítica.

A representação gráfica apresentada na Figura 26 confirma as informações e análises estatísticas realizadas com base na Figura 25. Conforme mostra a Figura 26, não há diferença sig-

nificativa de desempenho entre o Clus e o MHCAIS-Global com $\delta_{AF=0,8}$; da mesma forma essa diferança estatística não existe entre os algoritmos MHCAIS-Global com $\delta_{AF=0,8}$ e MHCAIS-Local com $\delta_{AF=0,8}$. Todavia, o algoritmo Clus apresenta desempenho estatisticamente superior ao MHCAIS-Local com $\delta_{AF=0,8}$. Deste modo, para o conjunto de experimentos apresentados nesta subseção, os testes de Frieadman e Nemenyi não se mostraram sensíveis ao número de algoritmos comparados, como se poderia supor.

No tocante às diferentes respostas retornadas pelos dois testes de hipóteses usados neste trabalho para um mesmo conjunto de experimentos, é importante esclarecer que o objetivo desta tese de maneira alguma é avaliar a confiabilidade ou indicar qualquer teste estatístico. Esta tese apenas usou testes validados e fortemente sugeridos pela literatura para de forma mais confiável analisar os resultados apresentados (DEMSAR, 2006).

Embora os testes estatísticos utilizados nesta tese divirjam quanto a significância ou não de parte dos resultados entre o MHCAIS e o Clus, deve-se destacar que este último apresentou melhores resultados comparado ao MHCAIS global/Local, considerando a AUPRC. Possivelmente, o Clus teve melhor desempenho, devido a possível dificuldade enfrentada pelo MHCAIS em lidar com atributos contínuos. É possível observar na Tabela 20 que a única base em que o MHCAIS (global) foi levemente superior ao Clus é a base Pheno, formada apenas por atributos discretos, conforme descreve a Tabela 7. De acordo com as informações mostradas na Tabela 7, as bases avaliadas são formadas, quase na sua totalidade, de atributos essencialmente contínuos, exceção feita às bases Pheno e ATP+DNAb-2. No que diz respeito a base ATP+DNAb-2, a Tabela 20 mostra que o Clus apresentou melhor AUPRC que os classificadores construídos pelo MHCAIS Global/Local, o que seria contraditório com a suspeita mencionada, já que esta base é formada basicamente por atributos categóricos, tendo apenas 2 atributos contínuos. Todavia, o Apêndice D apresenta parte das regras descobertas pelo Clus, onde aparentemente, os atributos contínuos foram mais relevantes, haja vista a frequência que aparecem no antecedente das regras, em detrimento dos atributos categóricos.

Reforçada a suspeita da dificuldade encontrada pelo MHCAIS em lidar com atributos contínuos, imagina-se que uma busca local mais efetiva contribuiria para a melhora do seu desempenho. No Apêndice D, é possível observar ainda, que o Clus permite que um atributo previsor ocorra mais de uma vez no antecedente da regra, beneficiado pela característica intrínseca de árvores de decisão que possibilita tal procedimento. Abaixo segue um exemplo de regra descoberta pelo Clus:

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE $>$ 41178,0 AND NUM_AA $<=$ 622 AND NUM_AA $>$ 388,0 AND MOL_WE $<=$ 52,019

Nota-se que os atributos MOL_WE e NUM_AA ocorrem duas vezes na mesma regra. Esta flexibilidade o MHCAIS não possui, pois cada atributo previsor aparece apenas uma única vez no antecedente, indicando que possívelmente este procedimento também possa contribuir para melhora de desempenho do MHCAIS. O Apêndice D mostra um conjunto parcial de regras descobertas pelo MHCAIS.

5.3.2 NÚMERO DE REGRAS DESCOBERTAS PELOS CLASSIFICADORES

Nesta subseção são apresentados e analisados os resultados referentes ao número de regras descobertas pelos classificadores construídos. A Tabela 24 mostra a quantidade de regras descobertas pelos algoritmos MHCAIS global/local e o Clus sobre as 10 bases avaliadas. Os melhores resultados são destacados em negrito para cada base. Para os algoritmos MHCAIS global e local foram considerados os valores de *matching* $\delta_{AF} = \{0{,}8; 0{,}9; 1{,}0\}$.

Tabela 24: Números de regras descobertas pelos algoritmos MHCAIS-Global/Local e Clus.

	MHCAIS-Global			MHCAIS-Local			Clus
Base de Dados	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	
ATP+DNAb-2	104	68	**41**	669	625	624	215
CellCycle	**2**	**2**	**2**	539	503	357	11
Church	6	6	6	523	519	500	**3**
Derisi	**4**	**4**	**4**	551	403	342	5
Eisen	**3**	**3**	**3**	377	333	281	51
Gasch1	**3**	**3**	**3**	423	355	321	27
Gasch2	4	3	4	512	577	420	41
Pheno	3	2	2	112	152	135	**1**
Seq	**2**	**2**	**2**	459	419	451	41
Spo	3	**2**	**2**	560	339	306	9

Os resultados da Tabela 24 são analisados usando o teste de hipóteses de Friedman para verificar se existe significância estatística entre as diferenças de desempenhos dos algoritmos. Conforme o esperado, os piores resultados são vistos para o MHCAIS-local, tendo em vista a característica intrínseca de classificadores locais em gerar mais regras. Deste modo a análise estatística fica restrita aos algoritmos MHCAIS-Global e Clus. Os resultados da Tabela 24 são transpostos para a Tabela 25 considerando o posto ordenado de cada classificador sobre cada base e o posto médio obtido sobre todas as bases.

Substituindo valores nas Equações 61 e 62, com base na Tabela 25, encontra-se $\chi^2_F = 12{,}63$ e $F_F = 3{,}04$. Considerando os graus de liberdade $k-1$ e $(k-1)\times(n-1)$, $k=4$ e $n=10$, tem-se o seguinte valor tabelado para a distribuição F de Snedecor, $F(3;27) = 2{,}96$. Logo, $F_F > F(3;27)$, e portanto a hipótese nula pode ser rejeitada, indicando que há diferença estatística entre os algoritmos comparados, considerando o número de regras descobertas.

Tabela 25: Postos obtidos pelos algoritmos MHCAIS-Global e Clus para o número de regras descobertas.

	MHCAIS-Global			Clus
Base de Dados	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	
ATP+DNAb-2	3	2	1	4
CellCycle	2	2	2	4
Church	3	3	3	1
Derisi	2	2	2	4
Eisen	2	2	2	4
Gasch1	2	2	2	4
Gasch2	2,5	1	2,5	4
Pheno	4	2,5	2,5	1
Seq	2	2	2	4
Spo	3	1,5	1,5	4
Média dos postos	2,55	2	2,06	3,4

Rejeitada a hipótese nula, calcula-se a distância crítica (*CD*) da através da Equação 63, com nível de significância de 95%, tem-se o seguinte valor $CD = 1{,}483$.

A Figura 27 é usada para facilitar o entendimento e a identificação das significâncias estatísticas entre os classificadores comparados. Tendo como base a estatística de Friedman, o gráfico ilustrado na Figura 27 posiciona ordenadamente, pelo posto médio, os classificadores num eixo horizontal. É representado também a distância crítica e os classificadores com distância menor que *CD* são agrupados por uma linha mais grossa.

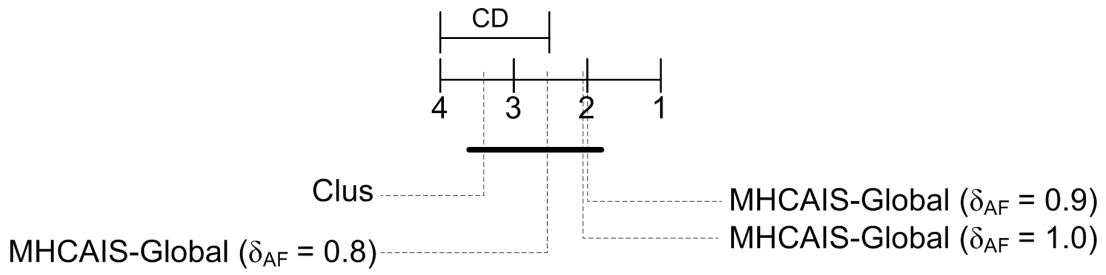


Figura 27: Classificadores ordenados pelos postos médios de acordo com resultados obtidos de número de regras e agrupados pela distância crítica.

De acordo com a Figura 27, todos os algoritmos comparados pertencem a um mesmo grupo. Deste modo, segundo o teste de Nemenyi, não há diferença significativa de desempenho entre os classificadores comparados.

Assim como na subseção anterior, chama a atenção a informação obtida pelo teste de Friedman onde o terceiro (MHCAIS-Global $\delta_{AF=0,8}$) colocado não possui desempenho significante

comparado ao quarto (Clus). A Figura 27 mostra visualmente que a distância entre este dois algoritmo parece próxima do limiar definido pela distância crítica calculada. Deste modo, foi aplicado o teste de Wilcoxon sobre o terceiro (3^{o}) e quarto (4^{o}) colocados para verificar se há divergência com o teste de Friedman.

A Tabela 26 mostra o resultado do teste de hipóteses de Wilcoxon comparando os algoritmos MHCAIS-Global $\delta_{AF=0,8}$ e Clus, levando em consideração o número de regras descobertas. Na tabela é mostrado o *p-value* e a aceitação (F) ou não (V) da hipótese nula, para um nível de confiança de 95%.

Tabela 26: Teste de Hipóteses de Wilcoxon com postos de sinais comparando os o número de regras descobertas pelos algoritmos MHCAIS-Global $\delta_{AF=0,8}$ (3^{o}) e Clus (4^{o}).

p-value	h
0,0195	V

Novamente, o teste de Wilcoxon apresenta resultado diferente do teste de Friedman; pois, conforme mostra a Tabela 26, a diferença de desempenho entre estes dois algoritmos é significativa. Desta forma, não é possível determinar com confiança se o MHCAIS-Global $\delta_{AF=0,8}$ é estatisticamente superior ou não ao Clus, no que diz respeito ao número de regras descobertas, haja vista que o teste de Wilcoxon e Friedman apresentam resultados distintos sobre os mesmos dados.

5.3.3 TOTAL DE CONDIÇÕES DO CONJUNTO DE REGRAS DESCOBERTAS

Nesta subseção são apresentados e analisados os resultados relativos ao total de condições do conjunto de regras descobertas pelos classificadores construídos. A Tabela 27 mostra o total de condições do conjunto de regras descobertas pelos algoritmos MHCAIS global/local e o Clus sobre as 10 bases avaliadas. Os melhores resultados são destacados em negrito para cada base. Para os algoritmos MHCAIS global e local foram considerados os valores de *matching* $\delta_{AF} = \{0,8; 0,9; 1,0\}$.

A Tabela 27 mostra que o MHCAIS-local independente do valor de δ_{AF} apresenta um número significativamente maior de número que condições que os algoritmo MHCAIS-Global e Clus. Este resultado era esperado, hava vista que o MHCAIS-Local constrói um grande número classificadores binário, e por consequência um número maior de condições. Isto posto, a análise estatística fica restrita aos algoritmos MHCAIS-Global e Clus, tendo em vista que pelo número de condições do MHCAIS-Local, fica evidente a diferença estatística.

A Tabela 28 mostra os postos otidos pelos classificadores comparados para cada base e os

Tabela 27: Total de condições do conjunto de regras descobertas pelos algoritmos MHCAIS-Global/Local e Clus.

	MHCAIS-Global			MHCAIS-Local			Clus
Base de Dados	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	
ATP+DNAb-2	786	202	**50**	3362	2867	1746	1603
CellCycle	6	**1**	**1**	6911	4696	357	30
Church	47	54	26	3154	2973	500	**2**
Derisi	21	5	**4**	5354	1232	342	6
Eisen	53	4	**2**	5343	1037	281	237
Gasch1	**2**	**2**	**2**	4857	1259	321	105
Gasch2	31	**4**	13	5919	5382	420	171
Pheno	16	4	1	616	357	135	**0**
Seq	**1**	**1**	**1**	505	816	451	188
Spo	10	**1**	3	2058	709	306	20

respectivos postos médios. O teste de Friedman é aplicado sobre os dados desta tabela.

Tabela 28: Postos obtidos pelos algoritmos MHCAIS-Global e Clus para o número total de condições do conjunto de regras descobertas.

	MHCAIS-Global			Clus
Base de Dados	$\delta_{AF=0,8}$	$\delta_{AF=0,9}$	$\delta_{AF=1,0}$	
ATP+DNAb-2	3	2	1	4
CellCycle	3	1,5	1,5	4
Church	3	4	2	1
Derisi	4	2	1	3
Eisen	3	2	1	4
Gasch1	2	2	2	4
Gasch2	3	1	2	4
Pheno	4	3	2	1
Seq	2	2	2	4
Spo	3	1	2	4
Média dos postos	3	2,05	1,65	3,3

Considerando as Equações 61 e 62, encontra-se $\chi_F^2 = 10{,}89$ e $F_F = 5{,}12$. Para os graus de liberdade $k-1$ e $(k-1)\times(n-1)$, $k=4$ e $n=10$, tem-se o seguinte valor tabelado para a distribuição F de Snedecor, $F(3;27) = 2{,}96$. Logo, $F_F > F(3;27)$, isto posto, rejeita-se a hipótese nula.

Rejeitada a hipótese nula, o valor de distância crítica (CD) encontrado através da Equação 63, com nível de significância de 95%, é de $CD = 1{,}483$.

A Figura 28 mostra os algoritmos ordenados pelo posto médio e agrupado pela diferença entre postos menor que a distância crítica calculada.

Os seguintes grupos são encontrados na Figura 28:

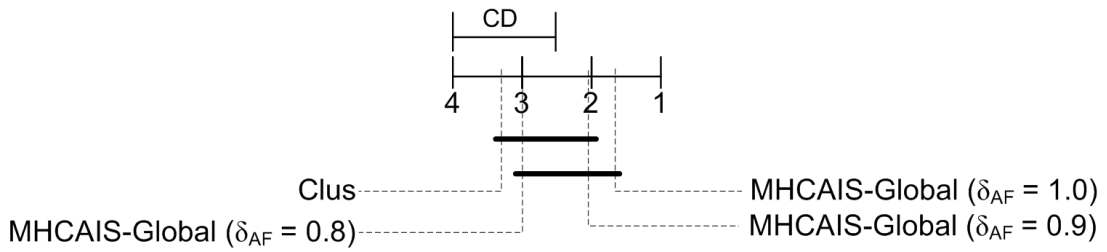


Figura 28: Classificadores ordenados pelos postos médios de acordo com resultados obtidos de número total de condições do conjunto de regras e agrupados pela distância crítica.

1. MHCAIS-Global $\delta_{AF} = 0{,}8$, $\delta_{AF} = 0{,}9$ e $\delta_{AF} = 1{,}0$.

2. Clus, MHCAIS-Global $\delta_{AF} = 0{,}8$ e $\delta_{AF} = 0{,}9$.

Deste modo, os algoritmos que fazem parte do mesmo grupo não têm diferença estatística de desempenho. A única diferença estatística observada no gráfico da Figura 28, fica restrita aos algoritmos MHCAIS-Global $\delta_{AF} = 1{,}0$ e Clus, onde o primeiro tem desempenho superior comparado ao último.

Seguindo o mesmo procedimento das subseções anteriores, aplicando o teste de Wilcoxon para comparar certos resultados do teste de Friedman. Os classificadores reanalisados são MHCAIS-Global $\delta_{AF=0{,}8}$ (3^{o} colocado) vs Clus (4^{o} colocado) e MHCAIS-Global $\delta_{AF=0{,}8}$ (3^{o}) vs $\delta_{AF=0{,}9}$ (2^{o}).

A Tabela 29 mostra o resultado do teste de hipóteses de Wilcoxon comparando os algoritmos MHCAIS-Global $\delta_{AF=\{0{,}8;0{,}9\}}$ e Clus, levando em consideração o total de condições do conjunto de regras descobertas. Na tabela é mostrado o *p-value* e a aceitação (F) ou não (V) da hipótese nula, para um nível de confiança de 95%.

Tabela 29: Teste de Hipóteses de Wilcoxon com postos de sinais comparando o total de condições do conjunto de regras descobertas pelos algoritmos MHCAIS-Global $\delta_{AF=\{0{,}8;0{,}9\}}$ e Clus. A colocação do algoritmo, segundo o teste de Friedman, encontra-se entre parênteses.

$\delta_{AF=0{,}8}$ (3^{o}) vs Clus (4^{o})		$\delta_{AF=0{,}8}$ (3^{o}) vs $\delta_{AF=0{,}9}$ (2^{o})	
p-value	h	*p-value*	h
0,084	F	0,0234	V

Conforme mostra a Tabela 29, o teste de Wilcoxon confirmou o teste de Friedman ao não atribuir diferença estatística entre o desempenho MHCAIS-Global $\delta_{AF=0{,}8}$ e Clus. Todavia, foi divergente do teste de Friedman na comparação entre MHCAIS-Global $\delta_{AF=0{,}8}$ vs $\delta_{AF=0{,}9}$

quando afirma diferença estatística entre os desempenhos dos 2 classificadores. Desta forma, não é possível determinar com confiança se o MHCAIS-Global $\delta_{AF=0,8}$ é estatisticamente superior ou não ao $\delta_{AF=0,9}$, no que diz respeito ao número de condições do conjunto de regras descobertas.

5.3.4 CUSTO COMPUTACIONAL DO MHCAIS GLOBAL E LOCAL

O custo computacional (em minutos) do MHCAIS global e local sobre cada uma das 10 bases avaliadas nos experimentos é mostrado na Tabela 30. Nesta tabela, o menor custo computacional para cada δ_{AF} considerado nos experimentos é destacado em negrito na comparação global vs. local.

Tabela 30: Custo computacional (em min.) do MHCAIS-Global/Local sobre as 10 bases avaliadas.

	MHCAIS-Global			MHCAIS-Local		
	$\delta_{AF=0.8}$	$\delta_{AF=0.9}$	$\delta_{AF=1.0}$	$\delta_{AF=0.8}$	$\delta_{AF=0.9}$	$\delta_{AF=1.0}$
ATP+DNAb-2	**109**	**82**	**58**	120	85	72
CellCycle	**6**	**5**	**5**	260	109	47
Church	**52**	**10**	**10**	114	76	49
Derisi	**18**	**9**	**7**	147	63	38
Eisen	**8**	**4**	**6**	94	51	33
Gasch1	**9**	**7**	**5**	327	143	75
Gasch2	**7**	**6**	**6**	182	146	44
Pheno	**4**	**1**	**2**	14	12	10
Seq	**20**	**13**	**8**	660	417	236
Spo	**12**	**5**	**4**	161	63	41

A Tabela 30 mostra que para ambas as versões do MHCAIS, à medida que o valor de δ_{AF} aumenta, o custo computacional para construção de classificadores diminui. Este comportamento se deve porque menos regras são descobertas para valores maiores de δ_{AF}.

No comparativo entre es versões para cada δ_{AF}, o MHCAIS global apresentou um custo computacional menor do que o local. Esta diferença era esperada haja vista que na construção de classificadores globais, um único classificador é construído para prever todas as classes do domínio de uma só vez, enquanto para classificadores locais, um classificadores é construído para cada classe do domínio.

6 CONCLUSÕES E PESQUISAS FUTURAS

A busca por técnicas computacionais que possibilitem a exploração, análise e extração de informações contidas em bases de dados biológicos é contínua e de fundamental importância para a bioinformática. A predição de funções de proteínas representa um dos problemas, dentre os vários, que a bioinformática tem se proposto a resolver.

De acordo com esta demanda, técnicas de mineração de dados, em especial a tarefa de classificação, têm sido sistematicamente usadas. Todavia, prever funções biológicas para proteínas representa um problema desafiador para a tarefa de classificação, haja vista que proteínas desempenham múltiplas funções, caracterizando assim um problema de classificação *multi-label*.

Ontologias têm sido criadas como um instrumento de padronização das informações biológicas, sendo utilizadas com frequência no processo de predição de funções de proteínas. Neste processo, os termos de uma ontologia são vistos, por exemplo, como um conjunto de definições que especificam a função de uma proteína. Um caso particular de ontologia frequentemente utilizada é a ontologia gênica (GO), onde parte dos termos definidos descrevem funções biológicas de proteínas. Neste trabalho, termos da GO representam as classes a serem previstas pelo algoritmo de classificação

Todavia, o uso da GO, sob alguns aspectos, representa um desafio para a construção de classificadores em mineração de dados. Primeiramente, o elevado número de termos definidos na GO (mais de 17.000 termos). Segundo, os termos são organizados hierarquicamente e algoritmos de classificação devem considerar esta estrutura para construção de classificadores. Finalmente, a estrutura hierárquica da GO representa um grafo acíclico direcionado, onde um nodo filho pode estar conectado a vários pais.

Deste modo, esta tese apresentou um novo método baseado em SIA para classificação hierárquica e *multi-label*, chamado de MHCAIS. Esta tese investiu em SIA para o problema abordado tendo em vista a capacidade deste tipo algoritmo em encontrar boas soluções para problemas complexos. O MHCAIS constrói classificadores para a predição de funções de proteínas. Ademais, a complexidade do método proposto aumenta significativamente, quando considerado

que as classes a serem preditas (termos da GO) são hierárquicamente organizadas na forma de um grafo acíclico direcionado.

Nesta tese foi investigada a capacidade do MHCAIS em construir classificadores globais e locais. O primeiro discrimina todas as classes do domínio em um único passo preditivo. O segundo, para cada classe do domínio, um classificador é construído. Em ambas as abordagens, o(s) classificador(es) construído(s) é(são) representado(s) na forma de regras SE-ENTÃO.

Além de desempenho preditivo, é de interesse do usuário, como por exemplo um biólogo, que algoritmos de classificação disponibilizem conhecimento interpretável para auxílio no processo decisório. Deste modo, além do aspecto preditivo, a capacidade de extrair dos dados conhecimento mais simples possível foi considerada pelo MHCAIS.

Experimentos computacionais contemplaram a investigação da influência de parâmetros e procedimentos no desempenho do MHCAIS global/local. A abordagem proposta foi ainda comparada com outros dois algoritmos descritos na literatura: PART e Clus. Ambos são baseados em árvores de decisão que podem ser transformadas em regras SE-ENTÃO. O PART constrói classificadores locais, enquanto a versão usada do Clus constrói classificadores globais.

Os resultados obtidos nos experimentos computacionais mostraram que a variação dos valores dos parâmetros e procedimentos testados influenciam no desempenho preditivo e simplicidade conhecimento descoberto sobre os dados de ambas as versões do MHCAIS.

Nos experimentos comparativos, o MHCAIS global e local foram superior ao algoritmo PART nos critérios de desempenho preditivo e simplicidade do conhecimento descoberto. Segundo os critérios estatísticos (desvio-padrão e teste de Wilcoxon) utilizados para análise, a diferença de desempenho foi significativa, principalmente no critério de simplicidade do conhecimento descoberto.

No comparativo com o algoritmo CLUS, as duas versões do MHCAIS foram superadas no critério de desempenho preditivo (área da curva *precision-recall*). Todavia, segundo o teste de hipóteses de Friedman, a diferença entre o CLUS e o MHCAIS-global não foi significativa. Por outro lado, de acordo com o teste hipóteses de Wilcoxon, a diferença entre MHCAIS e Clus foi estatisticamente significativa. Considerando o critério número de regras descobertas, o MHCAIS global não foi significativamente superior ao Clus segundo os testes de hipóteses de Friedman, porém o de acordo com teste de Wilcoxon existe diferença significativa de desempenho. Em termos de número de condições o MHCAIS-global $\delta_{AF} = 1{,}0$ apresentou resultados significativamente superiores ao CLUS, de acordo com o teste de Friedman e Wilcoxon. Deste modo, o MHCAIS global apresentou um perda de desempenho preditivo em detrimento da simplicidade do conhecimento extraído dos dados.

Concernente ao MHCAIS-local, este foi inferior à versão global na maioria dos experimentos, considerando o critério de correção preditiva. Todavia, o teste de Friedman não apresentou diferença significativa de desempenho nos experimentos realizados, exceção feita ao MHCAIS-local com $\delta_{AF} = 0{,}9$ onde essa diferença foi observada. Na comparação com o Clus, o desempenho do MHCAIS-local foi superado estatisticamente de acordo com o teste de Friedman em todos os experimentos realizados.

Nos experimentos comparativos entre o MHCAIS e Clus foi observado que possívelmente o primeiro é sensível aos atributos previsores contínuos. Esta hipótese é cogitada uma vez que a única base de dados (dentre 10 bases usadas) que o MHCAIS foi ligeiramente superior ao Clus é formada por atributos categóricos (discretos). Deste modo, acredita-se que o método de busca local usado nesta tese não foi eficiente, embora tenha obtidos melhores resultados do que a versão do MHCAIS sem busca local. Além disso, foi observado que o Clus lida melhor com atributos contínuos pela sua capacidade de permitir que um mesmo atributo previsor ocorra múltiplas vezes na árvore construída. O MHCAIS não permite tal flexibilidade, pois cada atributo previsor ocorre uma única vez na regra descoberta pelo SIA.

6.1 CONTRIBUIÇÕES

Levando em consideração os aspectos abordados, conclui-se que esta tese apresentou algumas contribuições para a comunidade acadêmica.

No aspecto da orginalidade, o MHCAIS é o primeiro algoritmo de indução de regras na forma SE *antecedente* ENTÃO *consequente* baseado em SIA para classificação hierárquica e *multi-label*, onde as classes a serem preditas são hierarquicamente organizadas em um grafo acíclico direcionado.

Nesta tese foi apresentada e formalizada a capacidade do MHCAIS na construção de classificadores globais e locais.

Os experimentos computacionais permitiram analisar e entender a influência dos valores de parâmetros no desempenho do MHCAIS global e local.

Nos experimentos foi observado que a função de *fitness* usada na fase de treinamento influencia significativamente no desempenho do classificador construído. Nesta tese foram estudadas duas funções de *fitness*: sensibilidade/especificidade e *f-measure*, onde a última apresentou os melhores resultados.

O algoritmo MHCAIS permite ao usuário o uso de *matching* parcial e total através do parâmetro $\delta_{AF} \in [0;1]^{\Re}$. Nos experimentos computacionais foi observado a influência do *mat-*

ching no desempenho do MHCAIS, onde o *matching* parcial ($\delta_{AF} = 0{,}8$) tende a maximizar a correção preditiva, enquanto o *matching* total ($\delta_{AF} = 1{,}0$) é mais eficiente na simplicidade do conhecimento extraído dos dados descobrindo menos regras e condições do conjunto de regras descobertas.

Uma nova base de proteínas foi construída e futuramente será disponibilizada como domínio público para uso irrestrito.

6.2 PESQUISAS FUTURAS

Embora os resultados tenham se mostrado promissores, considerando que o MHCAIS é o primeiro algoritmo de indução de regras baseado em SIA para classificação hierárquica, alguns aspectos das abordagens proposta merecem investigação futura.

De acordo com a limitação apresentada pelo MHCAIS em lidar com atributos contínuos, entende-se necessário o estudo de um método de busca local mais eficiente. Como alternativa pode-se aplicar conceitos de Enxame de Particulas (*Particle Swarm Optimization* - PSO) que é intrínsicamente um método para manipulação de atributos contínuos (KENNEDY; EBERHART; SHI, 2001) ou métodos de otimização, como o Quasi-Simplex (NELDER; MEAD, 1965).

Abordando ainda a dificuldade do MHCAIS na manipulação de atributos contínuos, a utilização de outras formas de representar este tipo de atributo pode melhorar o desempenho preditivo da abordagem proposta. O uso de conceitos de Sistemas Nebulosos (PEDRYCZ; GOMIDE, 1998) para discretizar os atributos contínuos em termos linguísticos parece uma alternativa a ser considerada. Intuitivamente, usar intervalos de valores para representar atributos contínuos nas condições das regras pode trazer ganho de desempenho, embora aumente o espaço de busca da regra. Estes intervalos podem ser representados, por exemplo, na seguinte forma: $limite\ inferior \leq valor \leq limite\ superior$.

Embora o MHCAIS tenha se mostrado eficiente na simplicidade do conhecimento descoberto, é importante validar a utilidade e o quão novo ou interessante este conhecimento é para o usuário. No caso específico do problema de predição de funções de proteínas abordado nesta tese, essa validação poderia ser feita por um biólogo.

Os experimentos computacionais mostraram que a função de *fitness* influencia no desempenho do MHCAIS. Deste modo, parece prudente avaliar o desempenho do MHCAIS com outras funções de *fitness*, diferentes das investigadas nesta tese. Neste sentido, pode-se ainda investigar o desempenho do MHCAIS considerando outros objetivos na função de *fitness*, como

por exemplo, o número de condições da regra e correção preditiva.

Alguns parâmetros do MHCAIS podem ser facilmente implementados para que automaticamente seja encontrado o valor que maximize o *fitness* da regra, embora este procedimento aumente o custo computacional. Exemplos de parâmetros que podem ser automatizados são: β_{FT}, δ_{AF} e δ_{FT}.

O algoritmo foi projetado para prever termos da GO que são organizados estruturalmente e hierarquicamente em um DAG. Entretanto, o modelo é facilmente adaptado para previsão de qualquer tipo de classes organizadas em uma estrutura hierárquica, mesmo que não seja um DAG. Por exemplo, os experimentos computacionais podem ser ampliados para avaliar o desempenho do algoritmo sobre o conjunto de proteínas enzimáticas classificadas com código EC (*Enzyme Commission*), onde as classes são organizadas na forma de uma árvore.

O procedimento de Extração Sequencial de Regras deve ser investigado e eventualmente melhorado, principalmente a retirada de exemplos da base de dados na construção de classificadores globais. No MHCAIS-global um exemplo é removido da base de treinamento apenas quando todas as suas classes são cobertas pelo conjunto de regras. Técnicas usadas para dar peso a exemplos da base de treinamento possivelmente melhoraria o desempenho do MHCAIS. Estas técnicas são frequentemente utilizadas em algoritmos baseados em *boosting*. No MHCAIS-global, estes pesos poderiam estimular a geração de regras prevendo classes que não foram ainda previstas e inibir a descoberta de regras prevendo classes já previstas.

Um mecanismo que encerre o período de evolução de regras quando não houver mais melhora do *fitness* das regras pode ser implementado. Esta melhoria possivelmente traria em uma diminuição do custo computacional.

O trabalho de Mezyk e Unold (2009) mostrou que o uso de tabela *hash* reduz o custo computacional, pois evita que haja recálculo sistemático do *fitness* de regras. Da mesma forma, este procedimento poderia ser implementado em ambas as versões do MHCAIS.

Nesta tese não foi contemplado nos experimentos medidas de desempenho específicas para classificação hierárquica. Deste modo, estas medidas devem ser futuramente estudadas, haja vista que esse procedimento pode apresentar uma melhora de desempenho dos classificadores construídos pelo MHCAIS.

Na construção de classificadores locais, o MHCAIS descobre iterativamente quantas regras forem necessárias para classificar os exemplos positivos e em seguida, repete o processo para que regras classificando os exemplos negativos sejam descobertas. Alterar este processo para que o MHCAIS local descobra apenas regras prevendo a classe positiva e assumir a classe ne-

gativa (regra *default*) como padrão apresentaria um conjunto de regras mais simples (menos regras e condições), redução do custo computacional, além de eventualmente melhorar a correção preditiva.

REFERÊNCIAS

ABBAS, A. K.; LITCHMAN, A. H. **Cellular and Molecular Immunology**. Chicago: W. B. Saunders Company, 2000.

ADRIAANS, P.; ZANTINGE, D. **Data Mining**. Harlow: Addison-Wesley, 1996.

ALATAS, B.; AKIN, E. Mining fuzzy classification rules using an artificial immune system with boosting. In: **Proceedings of the Conference on Advances in Databases and Information Systems (ADBIS-2005)**. [S.l.]: Springer-Verlag, 2005. (Lecture Notes in Computer Science, v. 3631), p. 283–293.

ALBERTS, B. et al. **Molecular Biology of the Cell**. 4o.. ed. New York: Garland Science, 2002.

ALTSCHUL, S. F. et al. Gapped BLAST and PSI-BLAST: A new generation of protein database search programs. **Nucleic Acids Res.**, v. 25, p. 3389–3402, 1996.

ALVES, R. T.; DELGADO, M. R.; FREITAS, A. A. Multi-label hierarchical classification of protein functions with artificial immune systems. In: BAZZAN, A. L. C.; CRAVEN, M.; MARTINS, N. F. (Ed.). **Proceedings of the 8th Brazilian Symposium on Bioinformatics (BSB)**. [S.l.]: Springer-Verlag, 2008. (Lecture Notes in Computer Science, v. 5167), p. 1–12.

ALVES, R. T. et al. Discovering multi-label hierarchical classification rules for protein function prediction. In: **Anais do III Workshop em Algoritmos e Aplicações de Mineração de Dados, 2007**. [S.l.]: Editora Universitária - UFPB, 2007. p. 87–90.

ALVES, R. T. et al. An artificial immune system for fuzzy-rule induction in data mining. In: YAO, X. et al. (Ed.). **Proceedings of the 8th International Conference on Parallel Problem Solving from Nature (PPSN)**. [S.l.]: Springer-Verlag, 2004. (Lecture Notes in Computer Science, v. 3242), p. 1011–1020.

ALVES, R. T. et al. Induction of fuzzy classification rules with an artificial immune system. In: **Proceedings of the 8th Brazilian Symposium on Neural Networks**. [S.l.]: IEEE Press, 2004.

BACKOFEN, R.; GILBERT, D. Bioinformatics and constraints. **Constraints**, v. 6, n. 2/3, p. 141–156, 2001.

BARAKAT, N.; BRADLEY, A. P. Rule extraction from support vector machines: Measuring the explanation capability using the area under the roc curve. In: **Proceedings of the 8th International Conference on Pattern Recognition (ICPR)**. [S.l.: s.n.], 2006. v. 2, p. 812–815.

BARAKAT, N.; DIEDERICH, J. Eclectic rule-extraction from support vector machines. **International Journal of Computational Intelligence**, v. 2, n. 1, p. 59–62, 2005.

BARKER, W. C. et al. The PIR - international protein sequence database. **Nucleic Acids Res.**, v. 26, n. 1, p. 27–32, 1998.

BARUTCUOGLU, Z.; SCHAPIRE, R. E.; TROYANSKAYA, O. G. Hierarchical multi-label prediction of gene function. **Bioinformatics**, v. 22, n. 7, p. 830–836, 2006.

BATEMAN, A. et al. The pfam protein families database. **Nucleic Acids Res.**, v. 32, p. D138–D141, 2004.

BäCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Evolutionary Computation 1: Basic Algorithms and Operators**. Bristol: Institute of Physiscs Publishing (IOP), 2000.

BäCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Evolutionary Computation 2: Advanced Algorithms and Operators**. Bristol: Institute of Physiscs Publishing (IOP), 2000.

BOECKMANN, B. et al. The swiss-prot protein knowledgebase and its supplement trembl in 2003. **Nucleic Acids Res.**, v. 31, n. 1, p. 365–370, 2003.

BOJARCZUK, C. C.; LOPES, H. S.; FREITAS, A. A. A constrained-syntax genetic programming system for discovering classification rules: Application to medical data sets. **Artificial Intelligence in Medicine**, v. 31, n. 1, p. 27–48, 2004.

BOUTELL, M. R. et al. Learning multi-label scene classification. **Pattern Recognition**, v. 37, n. 9, p. 1757–1771, 2004.

BRADLEY, D. W.; TYRRELL, A. M. Immunotronics: Hardware fault tolerance inspired by the immune system. In: **Proceedings of the 3rd International Conference on Evoluable Systems**. [S.l.]: Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1801), p. 11–20.

BRANDEN, C. I.; TOOZE, J. **Introduction to Protein Structure**. New York: Garland Publishing Inc, 1999.

BRU, C. et al. The ProDom database of protein domain families: More emphasis on 3D. **Nucleic Acids Res.**, v. 33, p. D212–D215, 2005.

BURNET, F. M. Clonal selection and after. In: BELL, G. I.; PERELSON, A. S.; Pimbley Jr, G. H. (Ed.). **Theoretical Immunology**. [S.l.]: Marcel Dekker Inc, 1978. v. 5, n. 3, p. 63–85.

CASTRO, P. A. D. et al. Designing ensembles of fuzzy classification systems: An immune-inspired approach. In: **Proceedings of the International Conference on Artificial Immune Systems (ICARIS)**. [S.l.]: Springer-Verlag, 2005. (Lecture Notes in Computer Science, v. 3627), p. 469–482.

CHAN, A.; FREITAS, A. A new ant colony algorithm for multi-label classification with applications in bioinformatics. In: **Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)**. [S.l.: s.n.], 2006. p. 27–34.

CHEN, M. S.; HAN, J.; YU, P. S. Data mining: an overview from a databases perspective. In: **Proceedings of the IEEE Transactions on Knowledge and Data Engineering**. [S.l.: s.n.], 1996. v. 8, p. 866–883.

CHU, S. et al. The transcriptional program of sporulation in budding yeast. **Science**, v. 282, p. 699–705, 1998.

CLARE, A. **Machine Learning and Data Mining for Yeast Functional Genomics**. Tese (Doutorado) — University of Wales, Aberystwyth, Wales, 2003.

CLARE, A.; KING, R. Knowledge discovery in multi-label phenotype data. In: **Proceedings of the 8th European Conference on Principles and Practice of Knowledge Discovery and Data Mining (PKDD-2001)**. [S.l.]: Springer-Verlag, 2001. (Lecture Notes in Computer Science, v. 2168), p. 42–53.

CRISTIANINI, N.; Shawe-Taylor, J. **An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods**. [S.l.]: Cambridge University Press, 2000.

CUTELLO, V. et al. Real coded clonal selection algorithm for global numerical optimization using a new inversely proportional hypermutation operator. In: **Proceedings of the 21st Annual ACM Symposium on Applied Computing**. [S.l.]: ACM Press, 2006. v. 2, p. 950–954.

DASGUPTA, D. **Artificial Immune Systems and Their Applications**. Berlin: Springer-Verlag, 1999.

DASGUPTA, D.; FORREST, S. Novelty detection in time series data using ideas from immunology. In: **5th International Conference on Intelligent Systems**. [S.l.: s.n.], 1996.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In: **Proceedings of the 23rd International Conference on Machine Learning**. [S.l.: s.n.], 2006. p. 233–240.

DE CASTRO, L. N. **Fundamentals of Natural Computing: Basic Concepts, Algorithms, and Applications**. Boca Raton: CRC Press, 2006.

DE CASTRO, L. N.; TIMMIS, J. **Artificial Immune Systems: A New Computational Intelligence Approach**. Berlin: Springer-Verlag, 2002.

DE CASTRO, L. N.; VON ZUBEN, F. J. ainet: An artificial immune network for data analysis. In: ____. [S.l.]: Idea Group Publishing, 2001. (DataMining: A Heuristic Approach), cap. XII, p. 231–259.

DE CASTRO, L. N.; VON ZUBEN, F. J. Learning and optimization using the clonal selection principle. **IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems**, v. 6, n. 3, p. 239–251, 2002.

DE RAEDT, L.; BLOCKEEL, H. Using logical decision trees for clustering. In: **Proceedings of the IJCAI Workshop on Frontiers of Inductive Logic Programming**. [S.l.: s.n.], 1997. p. 1–17.

DE RISI, J.; IYER, V.; BROWN, P. Exploring the metabolic and genetic control of gene expression on a genomic scale. **Science**, v. 278, p. 680–686, 1997.

DEMSAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine Learning Resarch**, v. 7, p. 1–30, 2006.

DUDA, R.; HART, P. **Pattern Classification and Scene Analysis**. [S.l.]: John Wiley & Sons, 1973.

EISEN, M. et al. Cluster analysis and display of genome-wide expression patterns. In: **Proceedings of the National Academy of Sciences**. [S.l.: s.n.], 1998. v. 95, p. 14863–14868.

EISNER, R. et al. Improving protein function prediction using the hierarchical structure of the gene ontology. In: **Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology**. [S.l.: s.n.], 2005.

ELIZONDO, D. A.; GONGORA, M. A. Current trends on knowledge extraction and neural networks. In: **Proceedings of the 15th International Conference On Artificial Neural Networks: Formal Models and Their Applications (ICANN)**. [S.l.]: Springer-Verlag, 2005. (Lecture Notes in Computer Science, v. 3697), p. 485–490.

FAUSETT, L. **Fundamentals of Neural Networks**. Upper Saddle River: Prentice Hall, 1994.

FAYYAD, U. M. et al. **Advances in Knowledge Discovery and Data Mining**. Cambridge: AAAI/MIT, 1996.

FISHER, R. A. **Statistical methods and scientific inference**. 2o.. ed. New York: Hafner Publishing Co., 1959.

FOGEL, G. B.; CORNE, D. W. **Evolutionary Computation in Bioinformatics**. San Franciso: Morgan Kaufmann Publishers, 2003.

FONSECA, J. S.; MARTINS, G. A. **Curso de Estatística**. São Paulo: Atlas S.A., 1996.

FRANK, E.; WITTEN, I. H. Generating accurate rule sets without global optimization. In: **Proceedings of the 15th International Conference on Machine Learning**. San Francisco: Morgan Kaufmann Publishers, 1998. p. 144–151.

FRAWLEY, W. J.; PIATETSKY-SHAPIRO, G.; MATHEUS, C. J. Knowledge discovery in databases: An overview. In: **Proceedings of the AAAI Workshop on Knowledge Discovery in Databases**. [S.l.: s.n.], 1991. p. 127.

FREITAS, A. A. **Data Mining and Knowledge Discovery with Evolutionary Algorithms**. Berlin: Springer-Verlag, 2002.

FREITAS, A. A.; CARVALHO, A. C. P. L. F. A tutorial on hierarchical classification with applications in bioinformatics. In: TANIAR, D. (Ed.). **Research and Trends in Data Mining Technologies and Applications**. [S.l.]: Idea Group, 2007. p. 175–208.

FREITAS, A. A.; LAVINGTON, S. H. **Mining Very Large Databases with Parallel Processing**. London: Kluwer, 1998.

FREITAS, A. A.; TIMMIS, J. Revisiting the foundations of artificial immune systems: A problem-oriented perspective. In: TIMMIS, J.; BENTLEY, P.; HART, E. (Ed.). **Proceedings of the 2nd International Conference on Artificial Immune Systems (ICARIS-2003)**. [S.l.]: Springer-Verlag, 2003. (Lecture Notes in Computer Science, v. 2787), p. 229–241.

FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. **Journal of the American Statistical Association**, v. 32, p. 675–701, 1937.

FRIEDMAN, M. A comparison of alternative tests of significance for the problem of m rankings. In: **Annals of Mathematical Statistics**. [S.l.: s.n.], 1940. v. 11, p. 86–92.

GASCH, A. et al. Genomic expression responses to dna-damaging agents and the regulatory role of the yeast atr homolog mec1p. **Mol. Biol. Cell**, v. 12, n. 10, p. 2987–3000, 2001.

GASCH, A. et al. Genomic expression program in the response of yeast cells to environmental changes. **Mol. Biol. Cell**, v. 12, p. 4241–4257, 2000.

GOLDBERG, D. E. **Genetic Algorithms in Search Optimization and Machine Learning**. MA: Addison-Wesley Reading, 1989.

GUARINO, N. Formal ontology and information systems. In: **International Conference On Formal Ontology In Information Systems**. [S.l.: s.n.], 1998. p. 3–15.

HAYKIN, S. **Neural Networks – A Comprehensive Foundation (2nd ed.)**. Upper Saddle River: Prentice Hall, 1999.

HOOS, H. H.; STüTZLE, T. **Stochastic Local Search: Foundations and Applications**. San Franciso: Morgan Kaufmann Publishers, 2005.

HUGHEY, R.; KARPLUS, K. Bioinformatics: a new field in engineering education. In: **Proceedings of the 31st Annual Frontiers in Education Conference**. [S.l.: s.n.], 2001. v. 2, p. 15–19.

HULO, N. et al. The PROSITE database. **Nucleic Acids Res.**, v. 34, p. D227–D230, 2006.

HUNT, J. E.; COOKE, D. E. Learning using an artificial immune system. **Journal of Network and Computer Applications**, v. 19, p. 189–212, 1996.

IMAN, R. L.; DAVENPORT, J. M. Approximations of the critical region of the friedman statistic. **Communications in Statistics**, p. 571–595, 1980.

ISHIBUCHI, H.; NAKASHIMA, T. Genetic-algorithm-based approach to linguistic approximation of nonlinear functions with many input variables. In: **Proceedings of the FUZZ-IEEE**. [S.l.]: IEEE Press, 1999. p. 779–784.

IUPAC-IUB. **Enzyme Nomenclature Recommendations**. New York: American Elsevier Pub. Co., 1972.

JENSEN, L. J. et al. Prediction of human protein function from post-translational modifications and localization features. **J. Mol. Biol.**, v. 319, n. 5, p. 1257–1265, 2002.

JENSEN, L. J. et al. Prediction of human protein function according to gene ontology categories. **Bioinformatics**, v. 19, n. 5, p. 635–642, 2003.

JENSEN, V. F. **Bayesian Networks and Decision Graphs**. Berlin: Springer-Verlag, 2001.

JERNE, N. K. Towards a network theory of the immune system. **Ann. Immunol.**, v. 125C, p. 373–389, 1974. Inst. Pasteur.

KARCI, A. Novelty in the generation of initial population for genetic algorithms. In: MEHNEN, J. et al. (Ed.). **Proceedings of the 8th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES)**. [S.l.]: Springer-Verlag, 2004. (Lecture Notes in Artificial Intelligence, 2), p. 268–276.

KENNEDY, J.; EBERHART, R. C.; SHI, Y. **Swarm intelligence**. San Francisco: Morgan Kaufmann Publishers, 2001.

KIM, J.; BENTLEY, P. Towards an artificial immune system for network intrusion detection: An investigation of dynamic clonal selection. In: **Proceedings of the Congress on Evolutionary Computation (CEC)**. [S.l.: s.n.], 2002.

KING, O. et al. Predicting gene function from patterns of annotation. **Genome Research**, v. 13, p. 896–904, 2003.

KIRITCHENKO, S. **Hierarchical Text Categorization and Its Application to Bioinformatics**. Tese (Doutorado) — Information Technology and Engineering, University of Ottawa, Ottawa, Canadá, 2005.

KUMAR, A. et al. Triples: A database of gene function in s. cerevisiae. **Nucleic Acids Res.**, v. 28, p. 81–84, 2000.

LAEGREID, A. et al. Predicting gene ontology biological process from temporal gene expression patterns. **Genome Research**, v. 13, p. 965–979, 2003.

LASKO, T. A. et al. The use of receiver operating characteristic curves in biomedical informatics. **Journal of Biomedical Informatics**, v. 38, n. 5, p. 404–415, 2005.

LEHNINGER, A. L.; NELSON, D. L.; COX, M. M. **Principles of Biochemistry**. 4o.. ed. New York: W. H. Freeman and Company, 2005.

LETOVSKY, S.; KASIF, S. Predicting protein function from protein/protein interaction data: a probabilistic approach. **Bioinformatics**, v. 19, p. i197–i204, 2003.

LEWIS, S. E. Gene ontology: Looking backwards and forwards. **Genome Biology**, v. 6, n. 1, p. 103:1–104:4, 2004.

LI, T.; OGIHARA, M. Detecting emotion in music. In: **International Symposium on Music Information Retrieval**. [S.l.: s.n.], 2003.

LU, X. et al. Automatic annotation of protein motif function with gene ontology terms. **BMC Bioinformatics**, v. 5, p. 122, 2004.

LUCAS, L. A. **Sistema de Inferência da Fuzzy Geral Tipo-2 Aplicado à Classificação**. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, Curitiba, Brasil, 2009.

MANNING, C. D.; RAGHAVAN, P.; SCHüTZE, H. **Introduction to Information Retrieval**. Cambridge: Cambridge University Press., 2008.

MCCALLUM, A. Multi-label text classification with a mixture model trained by EM. In: **Proceedings of the AAAI'99 Workshop on Text Learning**. [S.l.: s.n.], 1999.

McCulloch, W.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 115–133, 1943.

MENDEL, J.; JOHN, R. Type-2 fuzzy sets made simple. **IEEE Trans. of Fuzzy Systems**, v. 10, n. 2, p. 117–127, 2002.

MENOLASCINA, F. et al. Induction of fuzzy rules by means of artificial immune systems in bioinformatics. In: JIM, Y.; WANG, L. (Ed.). **Studies in Fuzziness and Soft Computing**. Berlin: Springer-Verlag, 2008. p. 1–17.

MEWES, H. W. et al. Mips: A database for protein sequences and complete genomes. **Nucleic Acids Res.**, v. 27, p. 44–48, 1999.

MEZYK, E.; UNOLD, O. Improving mining fuzzy rules with artificial immune systems by uniform population. In: MEHNEN, J. et al. (Ed.). **Proceedings of the World Soft Computing (WSC)**. [S.l.]: Springer-Verlag, 2009. (Lecture Notes in Computer Science, v. 58), p. 295–303.

MINGERS, J. An empirical comparison of pruning methods for decision tree induction. **Journal Machine Learning**, v. 4, n. 2, p. 227–243, 1989.

MINSKY, M.; PAPERT, S. **Perceptrons: An introduction to Computational Geometry**. [S.l.]: MIT Press, 1969.

MITCHELL, T. **Machine Learning**. New York: McGraw Hill, 1997.

MüLLER, K.-R. et al. An introduction to kernel-based learning algorithms. **IEEE Trans. on Neural Networks**, v. 129, n. 2, p. 181–201, 2001.

MULDER, N. J. et al. InterPro, progress and status in 2005. **Nucleic Acids Res.**, v. 33, p. D201–D205, 2005.

NASAROUI, O.; GONZALES, F.; DASGUPTA, D. The fuzzy artificial immune system: Motivation, basic concepts, and application to clustering and web profiling. In: **Proceedings of the IEEE International Conference on Fuzzy Systems**. [S.l.: s.n.], 2002. (Lecture Notes in Computer Science), p. 711–716.

NEAPOLITAN, R. E. **Probabilistic Reasoning in Expert Systems: Theory and Algorithms**. [S.l.]: John Wiley & Sons, 1990.

NELDER, J. A.; MEAD, R. A simplex method for function minimization. **Computer Journal**, v. 7, n. 4, p. 308–313, 1965.

OLIVER, S. A network approach to the systematic analysis of yeast gene function. **Trends in Genetics**, v. 12, n. 7, p. 241–242, 1996.

PAPPA, G. L.; BAINES, A. J.; FREITAS, A. A. Predicting post-synaptic activity in proteins with data mining. **Bioinformatics**, v. 21, n. 2, p. ii19–ii25, 2005.

PEARL, J. **Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference**. San Mateo: Morgan Kaufmann, 1988.

PEDRYCZ, W.; GOMIDE, F. **An Introduction to Fuzzy Sets: Analysis and Design**. Cambridge: MIT Press, 1998.

QUINLAN, J. R. **C4.5: Programs for Machine Learning**. San Mateo: Morgan Kaufmann, 1993.

ROTH, F. et al. Finding dna regulatory motifs within unaligned noncoding sequences clustered by whole-genome mrna quantitation. **Nature Biotechnology**, v. 16, p. 939–945, 1998.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 2o.. ed. Upper Saddle River: Prentice Hall, 2003.

SCHAPIRE, R. E.; SINGER, Y. Boostexter: A boosting-based systemfor text categorization. **Machine Learning**, v. 39, n. 2-3, p. 135–168, 2000.

SCHUG, J. et al. Predicting gene ontology functions from prodom and cdd protein domains. **Genome Research**, v. 12, p. 648–655, 2002.

SECKER, A.; FREITAS, A.; TIMMIS, J. Aisec: An artificial immune system for e-mail classification. In: SARKER, R. et al. (Ed.). **Proceedings of the Congress on Evolutionary Computation (CEC)**. [S.l.]: IEEE Press, 2003. v. 3242, p. 131–139.

SIGRIST, C. J. A. et al. PROSITE: A documented database using patterns and profiles as motif descriptors. **Briefings in Bioinformatics**, v. 3, n. 3, p. 265–274, 2002.

SMITH, B. Ontology. In: FLORIDI, L. (Ed.). **Blackwell Guide to the Philosophy of Computing and Information**. Oxford: Blackwell, 2003. p. 155–166.

SMOLA, A. J.; SCHöLKOPF, B. **Learning with Kernels**. Cambridge: The MIT Press, 2002.

SPELLMAN, P. et al. Comprehensive identification of cell cycle-regulated genes of the yeast *Saccharomyces cerevisiae* by microarray hybridization. **Molecular Biology of the Cell**, v. 9, p. 3273–3297, 1998.

STANSFIELD, W. D. **Genética**. 2o.. ed. São Paulo: McGraw Hill do Brasil, 1985.

STRACHAN, T.; ANDREW, P. **Genética Molecular Humana**. 2o.. ed. Porto Alegre: Artmed Editora, 2002.

STRYER, L.; TYMOCZKO, J. L.; BERG, J. M. **Bioquímica**. 5o.. ed. Rio de Janeiro: Guanabara Koogan, 2004.

SU, Y. H.; SHYR, W. J.; SU, T. J. Optimal design using clonal selection algorithm. In: **Proceedings of the 9th International Conference on Knowledge-Based Intelligent Information and Engineering Systems**. [S.l.]: Springer-Verlag, 2005. (Lecture Notes in Computer Science, v. 3681), p. 604–610.

SUN, A.; LIM, E.-P.; NG, W.-K. Performance measurement framework for hierarchical text classification. **Journal of the American Society for Information Science and Technology**, v. 54, n. 11, p. 1014–1028, 2003.

SZAFRON, D. et al. Proteome analyst - transparent high-throughput protein annotation: Function, localization and custom predictors. In: **Proceedings of the International Conference on Machine Learning Workshop on Machine Learning in Bioinformatics**. [S.l.: s.n.], 2003. p. 2–10.

SZAFRON, D. et al. Proteome analyst: Custom predictions with explanations in a web-based tool for highthroughput proteome annotations. **Nucleic Acids Res.**, v. 32, p. W365–W371, 2004.

TARAKANOV, A. O.; SKORMIN, V. A.; SOKOLOVA, S. P. **Immunocomputing: Principles and Applications**. Berlin: Springer, 2003.

The Gene Ontology Consortium. The gene ontology (GO) database and informatics resource. **Nucleic Acids Research**, v. 32, n. 1, p. 258–261, 2004.

The UniProt Consortium. The universal protein resource (UniProt). **Nucleic Acids Res.**, v. 35, p. D193–D197, 2007.

TICKLE, A. et al. The truth will come to light: Directions and challenges in extracting the knowledge embedded within trained artificial neural networks. **IEEE Trans. Neural Networks**, v. 9, n. 6, p. 1057–1068, 1998.

TIMMIS, J.; NEAL, M.; HUNT, J. An artificial immune system for data analysis. **BioSystems**, v. 55, p. 143–150, 2000.

TOWELL, G. G.; SHAVLIK, J. W. Extracting refined rules from knowledge-base neural networks. **Machine Learning**, v. 13, n. 1, p. 71–101, 1993.

TROYANSKAYA, O. G. et al. A bayesian framework for combining heterogeneous data sources for gene function prediction (in saccharomyces cerevisiae). **Proc Natl Acad Sci U S A**, v. 100, n. 14, p. 8348–8353, 2003.

TSOUMAKAS, G.; KATAKIS, I. Multi-label classification: An overview. **International Journal of Data Warehousing and Mining**, v. 3, n. 3, p. 1–13, 2007.

TU, K. et al. Learnability-based further prediction of gene functions in gene ontology. **Genomics**, v. 86, p. 922–928, 2004.

VAN RIJSBERGEN, C. **Information Retrieval**. 2o.. ed. [S.l.]: Butterworths, 1979.

VAPNIK, V. N. **The Nature of Statistical Learning Theory**. Berlin: Springer-Verlag, 1995.

VENS, C. et al. Decision trees for hierarchical multi-label classification. **Machine learning**, v. 73, n. 2, p. 185–214, 2008.

VINAYAGAM, A. et al. Applying support vector machines for gene ontology based gene function prediction. **BMC Bioinformatics**, v. 5, p. 116–129, 2004.

WATKINS, A.; BOGGESS, L. A new classifier based on resource limited artificial immune systems. In: **Proceedings of the Congress on Evolutionary Computation (CEC)**. [S.l.]: IEEE Press, 2002. v. 2, p. 1546–1551.

WATKINS, A.; TIMMIS, J.; BOGGESS, L. Artificial immune recognition system (airs): An immune-inspired supervised learning algorithm. **Genetic Programming and Evolvable Machines**, v. 5, n. 3, p. 291–317, 2004.

WEISS, S. M.; KULIKOWSKI, C. A. **Computer Systems That Learn**. San Francisco: Morgam Kaufmann, 1991.

WITTEN, I. H.; FRANK, E. **Data Mining: Practical Machine Learning Tools and Techniques**. 2o.. ed. San Mateo: Morgan Kaufmann, 2005.

YU, Y.; HOU, C.-Z. A clonal selection algorithm by using learning operator. In: **Proceedings of International Conference on Machine Learning and Cybernetics**. [S.l.: s.n.], 2004. v. 5, p. 2924–2929.

ZADEH, L. A. The concept of a linguistic variable and its application to approximate reasoning-I. **Information Sciences**, v. 8, p. 199–249, 1975.

ZHANG, M. L.; ZHOU, Z. H. A k-nearest neighbor based algorithm for multi-label classification. In: **Proceedings of the 1st IEEE International Conference on Granular Computing**. [S.l.: s.n.], 2005.

APÊNDICE A – CRITÉRIOS ESTATÍSTICOS

A seguir serão brevemente descritos os testes de hipóteses utilizados nesta tese para análise de significância estatística de desempenho entre algoritmos comparados.

A.1 TESTE DE HIPÓTESES DE WILCOXON DE POSTOS COM SINAIS

O teste de postos com sinais de Wilcoxon (*Wilcoxon Signed Rank Test*) é um teste não-paramétrico. Demsar (2006) sugere o teste de Wilcoxon para comparação de desempenho entre dois algoritmos, cuja distribuição dos dados não é conhecida, principalmente por ter menos sensibilidade a *outliers* (valores não normais ou espúrios).

O teste de Wilcoxon classifica em postos a diferença entre os algoritmos sobre cada base usada para avaliação de desempenho. Em seguida, soma as diferenças positivas e a negativas.

$$W^+ = \sum_{d_i>0} r_i; \tag{57}$$

$$W^- = \sum_{d_i<0} r_i; \tag{58}$$

onde r_i é o posto (*ranking*) da i-ésima base avaliada considerando as diferenças entre os algoritmos comparados.

Em seguida, calcula-se o valor de T que representa a menor soma dos postos.

$$T = \min(W^+; W^-); \tag{59}$$

Para $n <= 25$ os valores críticos de T são tabelados, onde n representa o número de base de dados avaliadas, descontados o número de empates ($d_i = 0$). Para valores de $N > 25$ usa-se a estatística z, pois assume-se que a distribuição é aproximadamente normal.

$$z_{calc} = \frac{T - \frac{1}{4}n(n+1)}{\sqrt{\frac{1}{24}n(n+1)(2n+1)}}; \tag{60}$$

A hipótese nula assume que a diferença de desempenho entre algoritmos não é significativa. Com nível de confiaça $\alpha = 0{,}05$, a hipótese nula não pode ser rejeitada se $-1{,}96 \leq z_{calc} \leq 1{,}96$ (FONSECA; MARTINS, 1996). A Tabela 31, adapatada do trabalho de Demsar (2006), permite ilustrar o uso do teste Wilcoxon.

Tabela 31: Exemplo de resultados classificados em postos com sinais pela diferença entre os desempenhos. Tabela adaptada do trabalho de (DEMSAR, 2006).

Bases de Dados	Algoritmo A	Algoritmo B	$d_i = A - B$	posto (r_i)
Base 1	0,763	0,768	+0,005	1,5
Base 2	0,599	0,591	-0,008	5
Base 3	0,954	0,971	+0,017	7
Base 4	0,628	0,661	+0,033	10
Base 5	0,882	0,888	+0,006	3
Base 6	0,936	0,931	-0,005	1,5
Base 7	0,661	0,668	+0,007	4
Base 8	0,583	0,583	0,000	-
Base 9	0,775	0,838	+0,063	12
Base 10	1,000	1,000	0,000	-
Base 11	0,940	0,962	+0,022	9
Base 12	0,619	0,666	+0,047	11
Base 13	0,972	0,981	+0,009	6
Base 14	0,957	0,978	+0,021	8

Substituindo os valores nas equações acima.

$$W^+ = 1{,}5 + 3 + 4 + 6 + 7 + 8 + 9 + 10 + 11 = 49{,}5;$$

$$W^- = 1{,}5 + 5 = 6{,}5;$$

$$T = \min(49{,}5; 6{,}5) = 6{,}5;$$

Considerando que o valor de n ($12 < 25$) - descontando os empates - é tabelado para pequenas amostras, os valores críticos com nível de confiança $\alpha = 0{,}05$ são: limite inferior de 13 e limite superior de 65. A hipótese nula deve ser rejeitada se $W^+ >$ *limite superior* ou $W^- <$ *limite inferior*. Logo, para $W^- = 6{,}5$, a hipótese nula é rejeitada.

A.2 TESTE DE HIPÓTESES DE FRIEDMAN

Segundo Demsar (2006), o teste Friedman (FRIEDMAN, 1937, 1940) é recomendado para comparar o desempenho de vários algoritmos sobre diferentes bases de dados. O teste é ainda fortemente sugerido por ser não-paramétrico, haja vista a dificuldade de se conhecer a distribuição dos dados. Quando a distribuição é conhecida, outros testes estatíticos são recomendados, como por exemplo o ANOVA (FISHER, 1959) para uma distribuição normal.

No teste de Friedman, os algoritmos sendo comparados são organizados por postos (*ranking*), de acordo com o desempenho obtido, sobre cada base de dados, atribuindo-se 1 ao primeiro colocado, 2 ao segundo, e assim sucessivamente. Em seguida, calcula-se a média dos postos obtidos pelos algoritmo sobre todos os conjuntos de dados usados nos experimentos.

Tabela 32: Exemplo de resultados ordenados pelos resultados e postos médios. Tabela adaptada do trabalho de (LUCAS, 2009).

Bases de Dados	Algoritmo A	Algoritmo B	Algoritmo C	Algoritmo D
Base 1	1	4	3	2
Base 2	3	4	2	1
Base 3	4	3	2	1
Base 4	3	4	2	1
Base 5	3	4	2	1
Postos médio	2,8	3,8	2,2	1,2

A diferença estatística existente entre os postos médios dos classificadores é obtida pela seguinte equação:

$$\chi_F^2 = \frac{12n}{k(k+1)} \left[\sum_j r_j^2 - \frac{k(k+1)^2}{4} \right]; \tag{61}$$

onde n é o número de base de dados, k é a quantidade de algoritmos e r_j é o posto médio para o j-ésimo algoritmo. Substituindo os valores da Tabela 32 na Equação 61,

$$\chi_F^2 = \frac{12 \times 5}{4 \times (4+1)} \times \left[2{,}8^2 + 3{,}8^2 + 2{,}2^2 + 1{,}2^2 - \frac{4 \times (4+1)^2}{4} \right] = 10{,}68;$$

Segundo Iman e Davenport (1980) teste de Friedman é considerado muito conservador, e por tal motivo, sugere o uso da estatística F_F:

$$F_F = \frac{(n-1)\chi_F^2}{n(k-1) - \chi_F^2}; \tag{62}$$

distribuída de acordo com a distribuição F de Snedecor (FONSECA; MARTINS, 1996)

com $k-1$e $(k-1)\times(n-1)$ graus de liberdade, logo:

$$F_F = \frac{(5-1)\times 10{,}68}{5\times(4-1)-10{,}68} = 9{,}89;$$

Analisando o valor tabelado de Snedecor $F(3;12) = 3{,}29$ e o valor $F_F = 9{,}89$, a hipótese nula deve ser rejeitada ($F_F > F(3;12)$) e portanto há diferença estatística entre os algoritmos analisados. Todavia, deve-se analisar quais algoritmos têm diferença significativa de desempenho. Para tal, Demsar (2006) sugere o teste de Nemenyi, que calcula a distância crítica (CD - *critical distance*) entre desempenhos.

$$CD = q_\alpha \sqrt{\frac{k(k-1)}{6n}}; \tag{63}$$

onde os valores de q_α são tabelados de acordo com o nível de significância (DEMSAR, 2006). Substituindo os valores com nível de significância de 95%:

$$CD = 2{,}569 \times \sqrt{\frac{4\times(4-1)}{6\times 5}} = 2{,}1;$$

Um algoritmo é considerado estatisticamente superior a outro se e a diferença entre os postos médio de cada um for maior que a distância crítica (CD) calculada. Considerando os dados da Tabela 32, o algoritmo D é estatisticamente superior ao algorimo B ($3{,}8-1{,}2 = 2{,}6 > 2{,}1$), por outro lado, o mesmo não se pode dizer entre o algoritmo C e D ($2{,}2-1{,}2 = 1{,}0 < 2{,}1$). Essa análise deve ser feita comparando todos contra todos para identificar quais algoritmos são ou não superiores estatisticamente.

APÊNDICE B – VISÃO GERAL DE TÉCNICAS USADAS EM CLASSIFICAÇÃO

A seguir serão descritas, de forma geral, algumas técnicas de aprendizado de máquina utilizadas com frequência na tarefa de classificação para resolução de problemas de bioinformática.

B.1 ÁRVORES DE DECISÃO

Na sua forma mais simples, uma Árvore de Decisão (AD) é uma lista de perguntas associada a um conjunto de respostas para cada pergunta, hierarquicamente arranjadas, que levam a uma decisão (Figura 29). A estrutura geral da árvore é organizada de forma que:

- O nodo raiz e cada nodo interno são rotulados com atributos previsores (perguntas);
- as arestas saindo de um nodo são rotuladas com valores do atributo daquele nodo (conjunto de respostas); e
- cada nodo folha é rotulado com a classe prevista para exemplos que pertençam àquele nó folha.

Uma AD encontra regras que recursivamente bifurcam o conjunto de dados a fim de produzir sub-conjuntos que sejam homogêneos com relação à classe dentro de um mesmo subconjunto (ou seja, quase todos os exemplos do subconjunto pertencem à mesma classe) e heterogêneos com relação a classe entre subconjuntos diferentes (MITCHELL, 1997; QUINLAN, 1993).

Aspectos positivos no uso de ADs são:

- simplicidade e compreensibilidade do conhecimento descoberto em árvores com pequenas estruturas;
- pode-se descobrir relevância de certos atributos, pois atributos mais relevantes tendem a estar mais próximos da raiz da árvore;

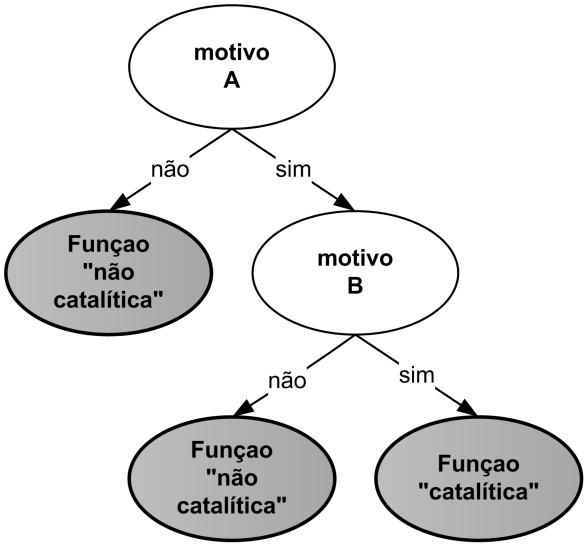


Figura 29: Exemplo de uma árvore de decisão simples. Os nodos em cor branca representam os atributos previsores *Motivo A* e *Motivo B*. Os nodos folhas (cor cinza) representam as classes preditas pela árvore, especificando se uma proteína executa ou não *função catalítica* de acordo com os valores dos atributos previsores.

- facilidade de obtenção de regras do tipo SE-ENTÃO (discutidas na seção 2.5.5) a partir das árvores geradas, onde cada regra pode ser facilmente interpretada de forma modular, independente de outras regras.

Os aspectos negativos são:

- torna-se mais difícil interpretar conhecimento descoberto em árvores com estruturas mais complexas;
- não é possível garantir que a solução ótima será encontrada, pois as soluções parciais são geradas incrementalmente durante a construção da árvore;
- Em estruturas mais complexas, atributos irrelevantes podem ser inseridos nesta estrutura da árvore, surgindo a necessidade de métodos que removam estas irrelevâncias (*pruning*).

B.2 MÁQUINAS DE VETORES DE SUPORTE

As máquinas de vetores suporte (SVMs - *Support Vector Machines*) constituem uma técnica de aprendizado de máquina, e têm sido amplamente usadas para classificação devido à sua capacidade de generalização, robustez diante de dados de grande dimensão (com alto número de atributos) e sólida fundamentação teórica (VAPNIK, 1995).

Basicamente, a partir de um conjunto de treinamento **S**, onde cada exemplo é um par $\langle \mathbf{x}_i, y_i \rangle$, em que $\mathbf{x}_i \in \Re^n$ é o vetor de características e $y_i \in \{-1, +1\}$ é a classe, onde o objetivo é encontrar um classificador linear representado por:

$$h(\mathbf{x}) = \text{sign}(f(\mathbf{x})) = (\langle \mathbf{x} \cdot \mathbf{w} \rangle + b); \tag{64}$$

onde $\mathbf{w} \in \Re^n$ é um vetor de pesos e o escalar b é um *bias*.

A função 64 representa o hiperplano separador ótimo encontrado capaz de classificar os dados de treinamento com um erro mínimo e maximizar a margem ρ de separação entre as classes (Figura 30).

Cada vetor de característica $\mathbf{x}_i$ deve satisfazer a seguinte restrição:

$$y_i(\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b) - 1 \geq 0, \text{para } i = 1, \ldots, n = |\mathbf{V}|; \tag{65}$$

A margem $\rho(\mathbf{x}_i, y_i)$ utilizada para classificar um padrão $\mathbf{x}_i$ é fornecida por $y_i f(\mathbf{x}_i)$. Ela mede a distância do padrão $\mathbf{x}_i$ em relação ao hiperplano separador. A margem ρ do classificador linear f é então definida como a margem mínima observada em todo conjunto de treinamento.

Maximizar ρ equivale a minimizar a norma de $\|\mathbf{w}\|$. Logo, pode-se manter ρ fixo e buscar um hiperplano com $\|\mathbf{w}\|$ pequeno tal que não existam exemplos de treinamento com margem menor que ρ (SMOLA; SCHöLKOPF, 2002). Fixando ρ em 1 a distância entre o hiperplano e os pontos que estão sobre a margem separadora ρ é determinada pela seguinte equação:

$$\frac{y_i(\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}; \tag{66}$$

Assim, tem-se o seguinte problema de otimização:

$$\begin{array}{c} \textbf{minimizar: } \|\mathbf{w}\|^2 \\ \textbf{Sob a restrição: } y_i(\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b) \geq 1, \text{para } i = 1, \ldots, n = |\mathbf{V}| \end{array}; \tag{67}$$

O hiperplano encontrado é capaz de realizar a classificação de conjuntos linearmente separáveis. Para o caso de conjuntos mais gerais, pode-se permitir que alguns padrões tenham margem menor que ρ incluindo variáveis soltas, o que dá um relaxamento nas restrições. Assim, o problema de otimização pode ser reformulado da seguinte maneira:

$$\textbf{minimizar: } \|\mathbf{w}\|^2 + k\sum_{i=1}^{|\mathbf{S}|}\xi_i$$
$$\textbf{Sob as restrições: } \begin{cases} \xi_i \geq 0 \\ y_i(\langle \mathbf{w}_i \cdot \mathbf{x}_i \rangle + b) + \xi_i \geq 1, \text{para } i = 1,\ldots,n = |\mathbf{V}| \end{cases} ; \quad (68)$$

onde k é uma constante que impõe um peso diferente para o treinamento em relação à generalização e ξ representa a variável responsável pela imposição do relaxamento.

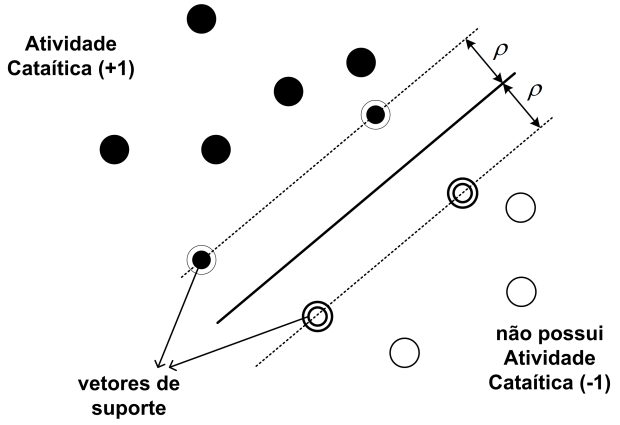


Figura 30: Exemplo simplicado de um problema de classificação com SVM, onde o classificador prevê se uma proteína possui atividade catalítica (+1) ou não (-1).

Entretanto, há muitos casos em que não é possível encontrar um hiperplano que divida satisfatoriamente os dados de treinamento, assim, os dados são não linearmente separáveis. Para superar esta limitação, utilizam-se máquinas não-lineares que projetam os dados de entrada em um *espaço de características* de dimensão maior, o que aumenta o poder computacional das máquinas lineares. Uma característica singular desse espaço é que a escolha de uma função de mapeamento Φ apropriada torna o conjunto de treinamento mapeado linearmente separável. SVMs lineares podem então ser utilizadas sobre o conjunto de treinamento mapeado no espaço de características (CRISTIANINI; Shawe-Taylor, 2000). Para isto, basta aplicar a função de mapeamento Φ a cada padrão nas Equações listadas (64 a 68) para o caso linear.

Neste caso, a única informação necessária sobre o mapeamento é uma definição de como o produto interno $\langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \rangle$ pode ser calculado. Isto é obtido com o uso de funções que recebem dois pontos $\mathbf{x}_i$ e $\mathbf{x}_j$ do espaço de entradas e computam o produto escalar $\langle \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j) \rangle$ no espaço de características, estas funções são chamada de *Kernels* (HAYKIN, 1999). As funções *Kernels* mais utilizados são: lineares, polinomiais, gaussianas ou RBF (*Radial-Basis Function*) e sigmoidais.

As desvantagens no uso de SVMs incluem a dificuldade de escolher a função de *Kernel* mais apropriada e os valores dos parâmetros, pois estes têm influência direta no desempenho do

classificador (MüLLER et al., 2001). Outra deficiência é a dificuldade de **interpretação dos resultados**, pois SVMs fazem parte dos classificadores estilo caixa-preta (*black-box*) (FREITAS, 2002). Entretanto, alguns algoritmos foram apresentados para extração de regras compreensíveis em SVMs (BARAKAT; DIEDERICH, 2005; BARAKAT; BRADLEY, 2006). Por ultimo, os classificadores SVMs convencionais são binários e para suas utilizações em problemas envolvendo, geralmente, mais de duas classes é necessário a adaptação do problema original decompondo-o em subproblemas (SMOLA; SCHöLKOPF, 2002).

B.3 REDES NEURAIS ARTIFICIAIS

Uma Rede Neural Artificial (RNA) é uma técnica computacional que constrói, a partir de um sistema neural biológico simplificado, um modelo matemático com capacidade de aprendizado, generalização, associação e abstração. As RNAs tentam aprender padrões diretamente dos dados através de um processo de repetidas apresentações dos dados à rede, ou seja, por experiência. Dessa forma, uma RNA procura por relacionamentos, constrói modelos e os corrige de modo a diminuir seu próprio erro (HAYKIN, 1999).

Uma RNA é composta por unidades de processamento simples (neurônios) as quais através de determinadas funções matemáticas (funções de ativação) realizam o processamento da informação. Os neurônios são dispostos em uma ou mais camadas e interligados por um grande número de conexões. Estas conexões estão associadas a pesos que armazenam o conhecimento representado no modelo e ponderam as entradas recebidas por cada neurônio da rede. Os pesos das conexões da rede vão sendo ajustados de forma que o conhecimento extraído dos dados possa ser representado internamente (FAUSETT, 1994).

A Figura 31 mostra o esquema de um neurônio artificial criado a partir do modelo simplificado do neurônio biológico por posto por McCulloch e Pitts (1943). Matematicamente, o neurônio da Figura 31 pode ser expresso por:

$$y_k = \varphi(u_k), \text{onde } u_k = \sum_{i=1}^{n} w_{ki} \cdot x_j + b_k; \tag{69}$$

Cada conexão i possui um valor de entrada x_i e um peso w_{ki}, onde o índice k identifica o neurônio e o índice i, a conexão; O *bias* b_k é uma espécie de excitador ou inibidor e tem o efeito de aumentar ou diminuir a entrada líquida (*net*) da unidade, dependendo se o seu valor for positivo ou negativo, respectivamente; u_k é o combinador linear que calcula a entrada líquida e φ é a função de ativação aplicada sobre u_k gerando uma saída y_k.

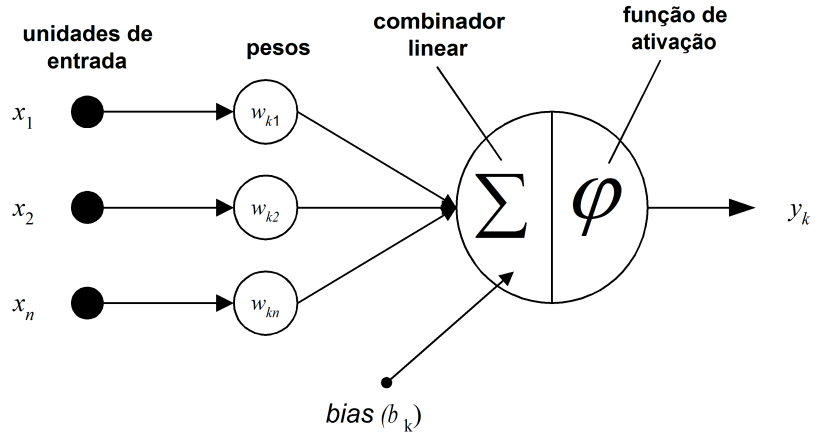


Figura 31: Representação de um neurônio artificial.

Os primeiros modelos de RNAs, como por exemplo o *perceptron*, têm como limitação sua incapacidade de lidar com problemas que não sejam linearmente separáveis (MINSKY; PAPERT, 1969). Esta limitação foi superada com a inclusão uma camada intermediária (entre as camadas de entrada e saída), o que deu origem ao modelo chamado de *Multi-Layer Perceptron* (MLP), o qual tem sido um dos mais usados em diversas aplicações do mundo real. A Figura 32 ilustra a arquitetura de um RNA - MLP com duas camadas intermediárias. Outros modelos de RNAs são: função de base radial (RBF), Hopfield, Kohonen, ART, entre outras. Cada modelo de RNA tem seu algoritmo de treinamento, como por exemplo, o algoritmo *backpropagation* para as redes MLPs. Por fim, várias são as funções de ativição aplicáveis a estes modelos, como por exemplo, as funções lineares e as não-lineares como a função logística, gaussiana, sigmoidal, tangente hiperbólica, sinal, entre outras (HAYKIN, 1999).

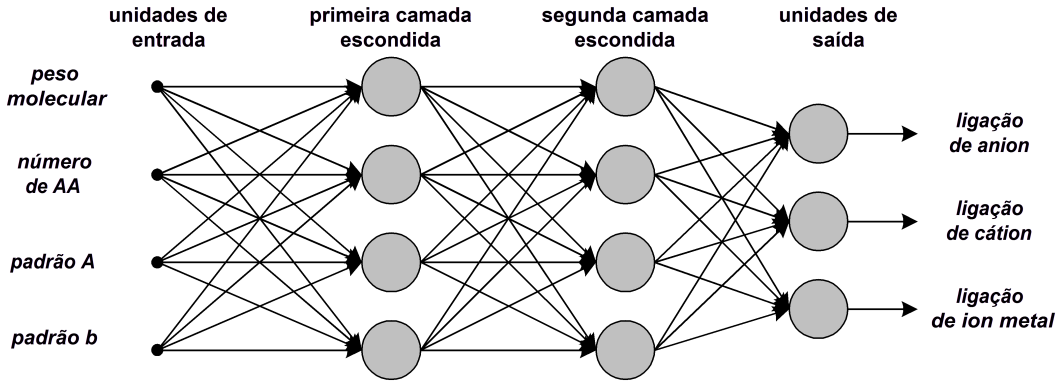


Figura 32: Exemplo de arquitetura de uma rede MLP. A rede possui 4 unidades da camada de entrada, onde cada unidade está associada, respectivamente aos atributos previsores *peso molecular*, *número de aminoácidos na sequência*, *padrão A* e *padrão B*. A camada de saída da rede prevê uma das três classes: ligação de anion, cátion ou ion metal.

.

Em problemas de classificação, cada unidade na camada de entrada pode representar um atributo previsor, enquanto que cada unidade na camada de saída pode representar uma classe (valor do atributo meta). Nos casos onde se deseja prever até duas classes é possível utilizar apenas um neurônio na camada de saída, estabelecendo critérios para definir os valores de saídas que estarão associadas às classes do problema.

As RNAs apresentam algumas vantagens, como tolerância a dados ruidosos; habilidade de representar qualquer função (linear ou não); capacidade de lidar com padrões de entrada representados por vetores de alta dimensão, em que os valores dos atributos podem ser contínuos ou discretos; facilmente paralelizáveis; e fácil aprendizado atingindo taxas de acerto satisfatórias. As desvantagens apresentadas são: dificuldade de determinar a topologia e parâmetros, como por exemplo em redes MLPs, o número de nodos escondidos ou o tipo de função de ativação; o treinamento, em geral, é demorado; e a dificuldade de **interpretação dos resultados**, pois RNAs fazem parte dos classificadores estilo caixa-preta (*black-box*) (FREITAS, 2002), entretanto, vários algoritmos têm sido desenvolvidos para extrair conhecimento de RNAS (TOWELL; SHAVLIK, 1993; TICKLE et al., 1998; ELIZONDO; GONGORA, 2005).

B.4 REDES BAYESIANAS

Matematicamente, uma Rede Bayesiana (RB) é uma representação compacta de uma tabela de conjunção de probabilidades do universo do problema. Por outro lado, do ponto de vista de um especialista, constitui um modelo gráfico que representa de forma simples as dependências (potencialmente causais) entre variáveis de um sistema. Uma Rede Bayesiana consiste dos seguintes elementos (PEARL, 1988):

- Um conjunto de variáveis e um conjunto de arcos ligando as variáveis.
- Cada nodo $n \in \mathscr{N}$ representa uma variável do domínio de aplicação e cada arco $a \in \mathscr{A}$ entre os nodos representa uma dependência probabilística.
- Cada variável possui um conjunto limitado de estados (valores) mutuamente exclusivos.
- As variáveis e arcos formam um grafo direcionado sem ciclos (DAG - *directed acyclic graph*).
- Cada nodo possui uma tabela de probabilidade condicional associada que quantifica a probabilidade de cada estado (valor) da variável do nodo condicionada, em cada combinação de estados das variáveis dos nodos pais.

A rede bayesiana ingênua (*naïve bayes network*) é a estrutura mais simples dentre as RBs (Figura 33), onde o nodo classe é especificado como pai e os atributos previsores são filhos, não existindo uma aresta conectando um atributo previsor a outro (DUDA; HART, 1973). O classificador associado a esse tipo de rede é chamado classificador bayesiano ingênuo por considerar que o efeito do valor de uma classe sobre um determinado atributo é independente dos valores dos outros atributos, simplificando sensivelmente as computações envolvidas. O principio básico desse classificador é aplicar o teorema de bayes (Teorema B.1) para calcular a probabilidade de diferentes *hipóteses*, à medida que novas *evidências* são observadas. Em classificação uma hipótese representa uma classe $c \in \mathscr{C}$ que se deseja prever e as evidências são representadas pelos exemplos que compõem a base de treinamento **S**.

Teorema B.1 (Teorema de Bayes)

$$P(c|\mathbf{S}) = \frac{P(\mathbf{S}|c)P(c)}{P(\mathbf{S})};$$

onde:

$P(c)$ *é a probabilidade a priori de classe c (hipótese);*

$P(\mathbf{S})$ *é a probabilidade a priori do base de treinamento* **S** *(evidências) antes de se admitir alguma hipótese para este conjunto; e*

$P(\mathbf{S}|c)$ *a probabilidade de se observar a base de treinamento* **S** *admitindo-se a hipótese c.*

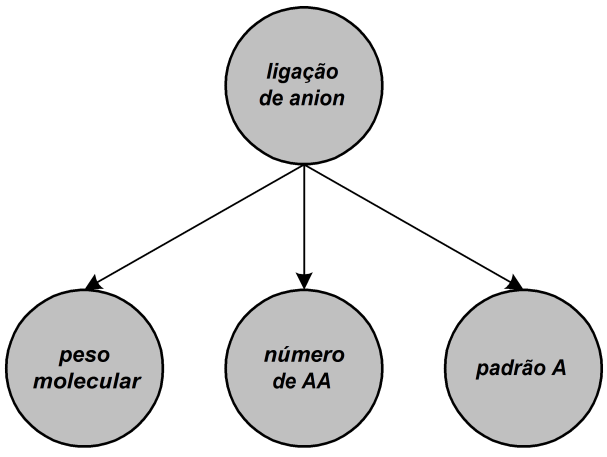


Figura 33: Representação de uma estrutura de um classificador bayesiano ingênuo, onde é apresentada a dependência dos atributos previsores *peso molecular*, *número de aminoácidos na sequência* e *padrão A* com a classe ligação de anion.

A probabilidade do exemplo $\mathbf{x} \in \mathbf{S}$ estar associado a uma classe c é obtida por:

$$P(\mathbf{x}|c) = \prod_{j=1}^{n} P(x_j|c); \tag{70}$$

onde x_j denota o valor do j-ésimo atributo do exemplo $\mathbf{x}$ e n é o número de atributos.

Aplicando o Teorema de Bayes para classificação de um exemplo para uma classe $c \in \mathscr{C}$ através de um classificador bayesiano ingênuo(NB - *Näive Bayes*) obtém-se pela seguinte equação:

$$\begin{aligned} c_{\text{NB}} &= \arg\max_{c\in\mathscr{C}} P(c|x_1,\cdots,x_j) \\ &= \arg\max_{c\in\mathscr{C}} \frac{P(x_1,\cdots,x_j|c)P(c)}{P(x_1,\cdots,x_j)} \\ &= \arg\max_{c\in\mathscr{C}} P(x_1,\cdots,x_j|c)P(c) \\ &= \arg\max_{c\in\mathscr{C}} \prod_{j=1}^{n} P(x_j|c)P(c) \\ &= \arg\max_{c\in\mathscr{C}} P(\mathbf{x}|c)P(c) \end{aligned} \tag{71}$$

As principais vantagens de um classificador bayesiano ingênuo são: facilmente implementável e computacionalmente eficiente. Entretanto, em domínios onde ocorre dependência entre atributos, comum em problemas de bioinformática, pode ocorrer perda de correção preditiva, já que este tipo de classificador não permite representar nenhuma dependência entre atributos.

Dependências entre variáveis podem ser tratadas usando uma tabela de probabilidade condicional. Estas tabelas são usadas para quantificar o(s) efeito(s) de nodo(s) pai(s) sobre um nodo filho. Assim, pode-se representar dependências potencialmente complexas entre atributos. A estrutura da rede codifica as relações de independência condicionada. O conjunto de probabilidades de uma rede bayesiana é a coleção de distribuições locais para cada nodo do domínio de aplicação. A estrutura geral de rede pode ser definida por:

$$P(x_i,\ldots,x_n) = \prod_{i=1}^{n} P(x_i|\text{pais}(x_i)); \tag{73}$$

A distribuição conjunta de probabilidades dada pela Equação 73 pode ser usada para responder a qualquer pergunta sobre valores das variáveis do domínio de aplicação, através de *inferência*. Em problemas de classificação, o processo de inferência é utilizado para classificar exemplos. Inferências podem ser realizadas sobre redes bayesianas para:

•Diagnósticos: dos efeitos para as causas;

- causas: de causas para efeitos;
- relacionamentos intercausais: entre causas de um efeito comum.

Alguns algoritmos para inferência em redes bayesianas podem ser encontrados em Jensen (2001).

Quando a estrutura e as variáveis observadas do problema são conhecidas, o treinamento fica restrito apenas à computação das probabilidades condicionais dos nodos. No contexto de mineração de dados, a estrutura da rede normalmente não é conhecida, e tanto a estrutura quanto as probabilidades condicionais têm que ser extraídas a partir dos dados (JENSEN, 2001; WITTEN; FRANK, 2005).

A maior vantagem de redes bayesianas é a capacidade da estrutura de representar conhecimento de fácil interpretação, além de fácil modificação do modelo para se obter melhor correção preditiva (NEAPOLITAN, 1990). Redes bayesianas com estruturas complexas (não ingênuas) permitem representar dependências entre atributos, que não eram possíveis em classificadores bayesianos ingênuos. Entretanto, para estruturas mais complexas o treinamento da rede bayesiana apresenta um maior custo computacional do que as estruturas ingênuas.

APÊNDICE C – EXEMPLOS DE REGRAS DESCOBERTAS PELO MHCAIS

Conjunto parcial de regras descobertas pelo MHCAIS Global para a base de dados ATP+DNAb. Nas regras foram omitidos as classes previstas pelas regras, tendo em vista que todas as classes do domínio estão representadas no consequente.

IF (PS00211 = 1) AND (PS00871 = 0) AND (PS00018 = 0) AND (PS00750 = 0) AND (PS00751 = 1)

IF (PS00211 = 0) AND (PS00676 = 0) AND (PS00688 = 1) AND (PS00449 = 0) AND (PS00152 = 0) AND (PS00245 = 1) AND (MOL_WE $<$ 249890)

IF (PS00688 = 0) AND (PS00871 = 0) AND (PS00449 = 0) AND (PS00154 = 1) AND (PS00674 = 0) AND (PS00690 = 0) AND (PS01008 = 0) AND (PS00298 = 0) AND (PS00058 = 0)

IF (PS00449 = 0)

IF (PS00676 = 0) AND (PS00688 = 0) AND (PS00154 = 0) AND (PS00390 = 0) AND (PS00391 = 0) AND (PS00605 = 1) AND (PS00152 = 0) AND (PS00153 = 1) AND (PS00690 = 0) AND (PS01008 = 1) AND (PS00636 = 0) AND (PS00297 = 0) AND (PS01036 = 1) AND (PS00058 = 0) AND (PS01046 = 0) AND (PS00750 = 0) AND (PS00751 = 0)

IF (PS00675 = 0) AND (PS00870 = 0) AND (PS00871 = 1) AND (PS00154 = 1) AND (PS00391 = 0) AND (PS00605 = 0) AND (PS00674 = 0) AND (PS01047 = 1) AND (PS00152 = 0) AND (PS00389 = 0) AND (PS00153 = 1) AND (PS00245 = 0) AND (PS00681 = 0) AND (PS00296 = 0) AND (PS00662 = 0) AND (PS01008 = 0) AND (PS00636 = 1) AND (PS00297 = 0) AND (PS00329 = 1) AND (PS00486 = 0) AND (PS01046 = 0) AND (PS00847 = 0) AND (PS00751 = 0) AND (PS00995 = 0) AND (NUM_AA $<$ 2239) AND (MOL_WE $>=$ 91560)

IF (PS00154 = 1) AND (PS01047 = 1)

IF (PS00676 = 0) AND (PS00154 = 0) AND (PS00681 = 1) AND (PS00296 = 0) AND (PS00690 = 0) AND (PS01008 =

IF (PS00154 = 0) AND (PS00636 = 1) AND (MOL_WE >= 30218)

IF (PS00154 = 1) AND (PS00391 = 0) AND (PS01047 = 1) AND (PS00636 = 0) AND (NUM_AA >= 142)

IF (PS00449 = 1) AND (PS00154 = 0) AND (PS00391 = 0) AND (PS00674 = 0) AND (PS00152 = 0) AND (PS00018 = 0) AND (PS00690 = 1) AND (PS01036 = 0) AND (PS00177 = 1) AND (NUM_AA < 1549) AND (MOL_WE >= 78746)

IF (PS00058 = 1)

IF (PS00300 = 0) AND (PS00298 = 1) AND (PS00177 = 1) AND (PS00751 = 0) AND (PS00995 = 0)

IF (PS00636 = 1) AND (NUM_AA >= 336)

IF (PS01047 = 1) AND (MOL_WE >= 77119)

IF (PS00298 = 1)

IF (PS00675 = 0) AND (PS00688 = 0) AND (PS00871 = 0) AND (PS00449 = 1) AND (PS00154 = 0) AND (PS00391 = 0) AND (PS00674 = 0) AND (PS00153 = 0) AND (PS00245 = 0) AND (PS00296 = 0) AND (PS00690 = 1) AND (PS00297 = 0) AND (PS00329 = 1) AND (PS00300 = 0) AND (PS00177 = 1) AND (PS00486 = 0) AND (PS00321 = 0) AND (PS00995 = 0) AND (MOL_WE < 76811)

IF (PS00245 = 1)

IF (PS01046 = 1) AND (NUM_AA < 680)

IF (PS01046 = 1) AND (NUM_AA >= 960)

IF (PS00245 = 0) AND (PS00662 = 1) AND (PS00300 = 1) AND (PS00058 = 0) AND (MOL_WE < 48375)

IF (PS00688 = 0) AND (PS00605 = 1) AND (PS00153 = 0) AND (PS00681 = 0) AND (PS01008 = 1) AND (PS00297 = 0) AND (PS01036 = 1) AND (PS00058 = 0) AND (PS01046 = 0) AND (PS00995 = 0)

IF (PS00211 = 1) AND (PS00245 = 0) AND (PS00662 = 1) AND (PS00177 = 0) AND (PS00995 = 0)

IF (PS00688 = 1) AND (NUM_AA < 476)

IF (PS00390 = 1) AND (PS00636 = 1) AND (PS01046 = 0) AND (PS00321 = 0) AND (NUM_AA >= 674) AND (MOL_WE < 152153)

IF (PS01008 = 1)

IF (PS00676 = 1) AND (NUM_AA < 456)

IF (PS00152 = 1) AND (NUM_AA >= 943)

IF (PS00018 = 1)

IF (NUM_AA >= 671)

IF (PS01046 = 1) AND (NUM_AA < 924)

IF (PS00676 = 0) AND (PS00871 = 0) AND (PS00449 = 0) AND (PS00154 = 0) AND (PS00391 = 1) AND (PS00153 = 0) AND (PS00245 = 1) AND (PS00018 = 0) AND (PS00296 = 0) AND (PS00662 = 0) AND (PS00636 = 0) AND (PS00300 = 0) AND (PS00177 = 0) AND (PS01046 = 0) AND (PS00847 = 1) AND (PS00321 = 1) AND (PS00750 = 1) AND (PS00751 = 0) AND (NUM_AA < 2111) AND (MOL_WE < 77739)

IF (NUM_AA >= 697)

IF (PS00675 = 0) AND (PS00688 = 0) AND (PS00870 = 1) AND (PS00154 = 0) AND (PS00605 = 1) AND (PS01047 = 0) AND (PS00389 = 0) AND (PS00690 = 0) AND (PS00297 = 0) AND (PS00329 = 1) AND (PS00298 = 0) AND (PS00486 = 0) AND (PS00847 = 1) AND (PS00751 = 0) AND (NUM_AA >= 85)

IF (PS01047 = 1) AND (MOL_WE >= 80058)

IF (PS00847 = 1)

IF (PS00211 = 0) AND (PS00675 = 1) AND (PS00676 = 1) AND (PS00688 = 1) AND (PS00870 = 0) AND (PS00871 = 0) AND (PS00449 = 0) AND (PS00154 = 1) AND (PS00391 = 0) AND (PS00674 = 0) AND (PS00389 = 0) AND (PS00153 = 0) AND (PS00296 = 0) AND (PS00662 = 1) AND (PS00690 = 0) AND (PS00329 = 0) AND (PS01046 = 0) AND (PS00995 = 1) AND (NUM_AA < 495) AND (MOL_WE >= 166681)

IF (PS00688 = 1) AND (PS00870 = 0) AND (PS00871 = 0) AND (PS00154 = 0) AND (PS00152 = 0) AND (PS00245 = 1) AND (PS00681 = 0) AND (PS00690 = 0) AND (PS00329 = 1) AND (PS01036 = 0) AND (PS00298 = 0) AND (PS00058 = 0) AND (PS01046 = 0) AND (PS00847 = 0) AND (NUM_AA >= 935)

IF (PS00675 = 0) AND (PS00676 = 0) AND (PS00674 = 0) AND (PS00389 = 0) AND (PS00245 = 0) AND (PS00662 = 0) AND (PS00690 = 1) AND (PS00636 = 1) AND (PS01036 = 0) AND (PS01046 = 0) AND (PS00847 = 1) AND (PS00750 = 0) AND (MOL_WE < 37913)

IF (PS00154 = 1) AND (MOL_WE < 77252)

IF (MOL_WE >= 103214)

IF (PS00211 = 1) AND (NUM_AA >= 939)

IF (PS00211 = 1) AND (NUM_AA >= 602)

IF (PS00321 = 1)

IF (PS01046 = 1)

IF (PS00676 = 1) AND (PS00688 = 0) AND (PS01047 = 0) AND (PS00636 = 1) AND (PS00847 = 0)

IF (PS00636 = 1)

IF (PS00152 = 1) AND (MOL_WE < 50407)

APÊNDICE D – EXEMPLOS DE REGRAS DESCOBERTAS PELO CLUS

Conjunto parcial de regras descobertas pelo Clus para a base de dados ATP+DNAb. Nas regras foram omitidos as classes previstas pelas regras, tendo em vista que todas as classes do domínio estão representadas no consequente.

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 41178.0 AND NUM_AA <= 622 AND NUM_AA > 320.0 AND MOL_WE <= 52.019

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 41178.0 AND NUM_AA <= 622 AND NUM_AA > 388.0 AND MOL_WE <= 52.019

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 41178.0 AND NUM_AA <= 388 AND NUM_AA > 320.0 AND MOL_WE <= 52.019

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 622 AND NUM_AA > 320.0 AND MOL_WE <= 41.178

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 622 AND NUM_AA > 345.0 AND MOL_WE <= 41.178

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 38031.0 AND NUM_AA <= 622 AND NUM_AA > 345.0 AND MOL_WE <= 41.178

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 622 AND NUM_AA > 345.0 AND MOL_WE <= 38.031

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 345 AND NUM_AA > 320.0 AND MOL_WE <= 41.178

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 345 AND NUM_AA > 337.0 AND MOL_WE <= 41.178

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 337 AND NUM_AA > 320.0 AND MOL_WE <= 41.178

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 320

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 320 AND NUM_AA > 278.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 320 AND NUM_AA > 301.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 301 AND NUM_AA > 278.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 30500.0 AND NUM_AA <= 301 AND NUM_AA > 278.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 30500.0 AND NUM_AA <= 301 AND NUM_AA > 281.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 30500.0 AND NUM_AA <= 301 AND NUM_AA > 284.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 30500.0 AND NUM_AA <= 284 AND NUM_AA > 281.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 30500.0 AND NUM_AA <= 281 AND NUM_AA > 278.0

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 301 AND NUM_AA > 278.0 AND MOL_WE <= 30.500

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27838.0 AND NUM_AA <= 278

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 27882.0 AND NUM_AA <= 278

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE > 29969.0 AND NUM_AA <= 278

IF PS00152 = 0 AND PS00211 = 1 AND MOL_WE <= 23.521 AND NUM_AA <= 191

IF PS00152 = 1 AND NUM_AA > 498.0 AND NUM_AA <= 670

IF PS00152 = 1 AND NUM_AA <= 498

IF PS00152 = 1 AND NUM_AA <= 498 AND NUM_AA > 32.0

IF PS00152 = 1 AND NUM_AA <= 498 AND NUM_AA > 32.0 AND MOL_WE > 49316.0 AND MOL_WE <= 54.662

IF PS00152 = 1 AND NUM_AA <= 498 AND NUM_AA > 467.0 AND MOL_WE > 49316.0
AND MOL_WE <= 54.662

IF PS00152 = 1 AND NUM_AA <= 467 AND NUM_AA > 32.0 AND MOL_WE > 49316.0
AND MOL_WE <= 54.662

IF PS00152 = 1 AND NUM_AA <= 467 AND NUM_AA > 32.0 AND MOL_WE > 51311.0
AND MOL_WE <= 54.662

IF PS00152 = 1 AND NUM_AA <= 467 AND NUM_AA > 32.0 AND MOL_WE > 49316.0
AND MOL_WE <= 51.311

IF PS00152 = 1 AND NUM_AA <= 498 AND NUM_AA > 32.0 AND MOL_WE <= 49.316

IF PS00152 = 1 AND NUM_AA <= 498 AND NUM_AA > 32.0 AND MOL_WE <= 49.316
AND MOL_WE > 42591.0

IF PS00152 = 1 AND NUM_AA <= 498 AND NUM_AA > 32.0 AND MOL_WE <= 42.591

APÊNDICE E – CURVAS DE *PRECISION-RECALL* OBTIDAS NOS EXPERIMENTOS

Abaixo são apresentadas as curvas de *Precision-Recall* (AUPRC) obtidas no experimentos computacionais realizados neste trabalho.

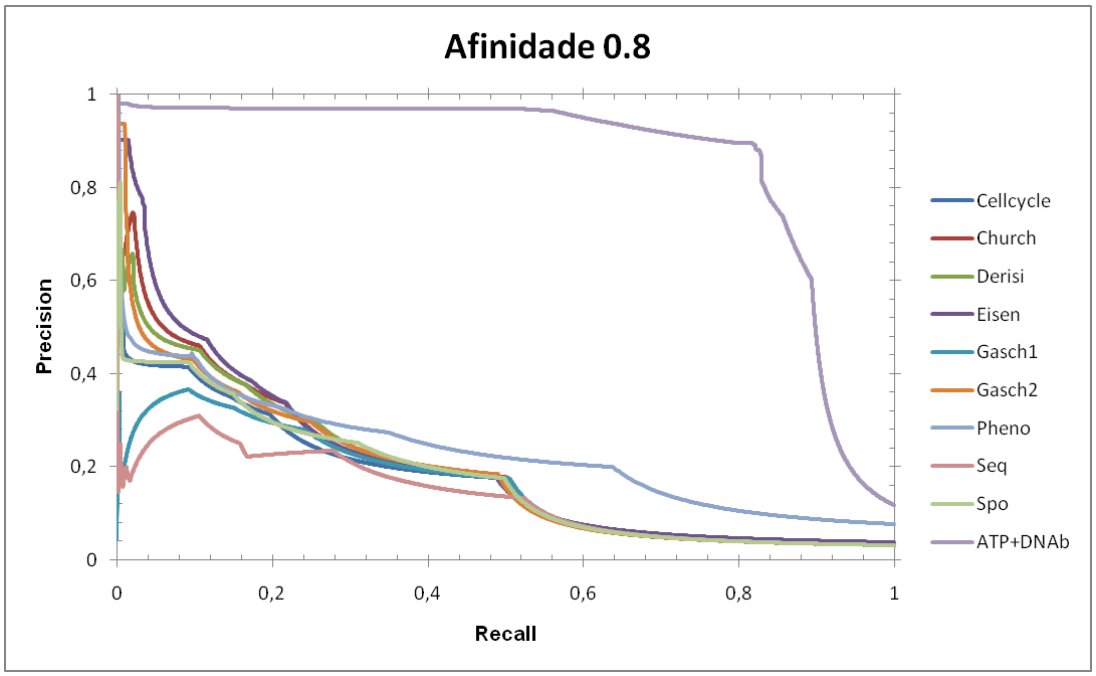


Figura 34: Curvas de precision-recall obtidas pelo MHCAIS-Local $\delta_{AF=0,8}$ sobre todas as bases avaliadas.

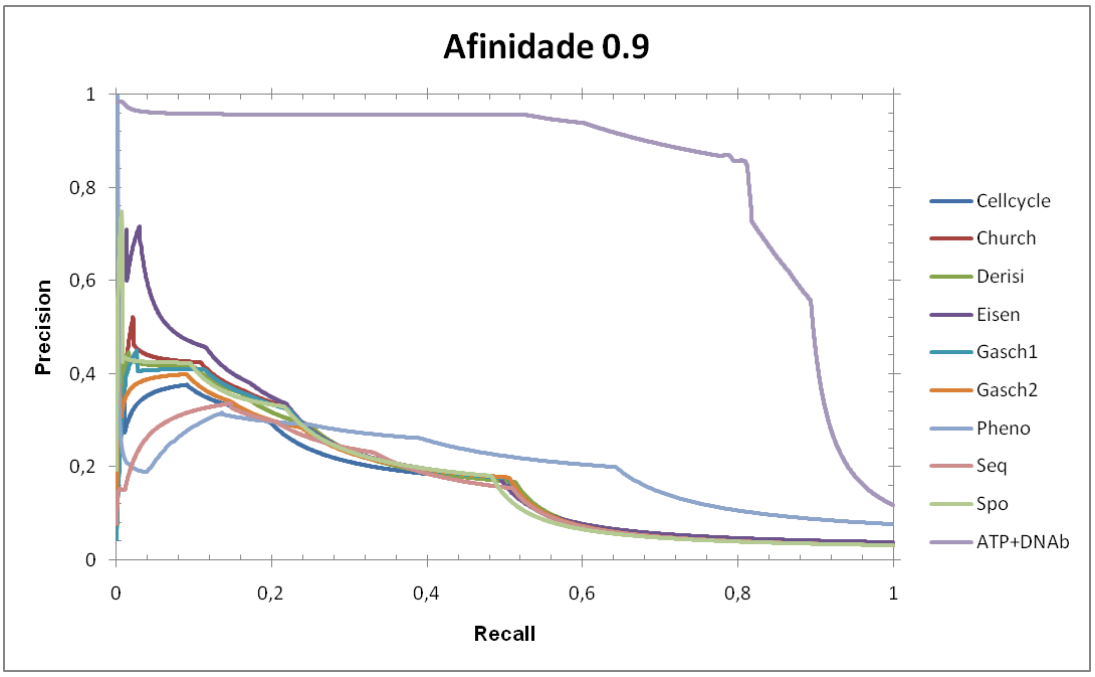


Figura 35: Curvas de precision-recall obtidas pelo MHCAIS-Local $\delta_{AF=0,9}$ sobre todas as bases avaliadas.

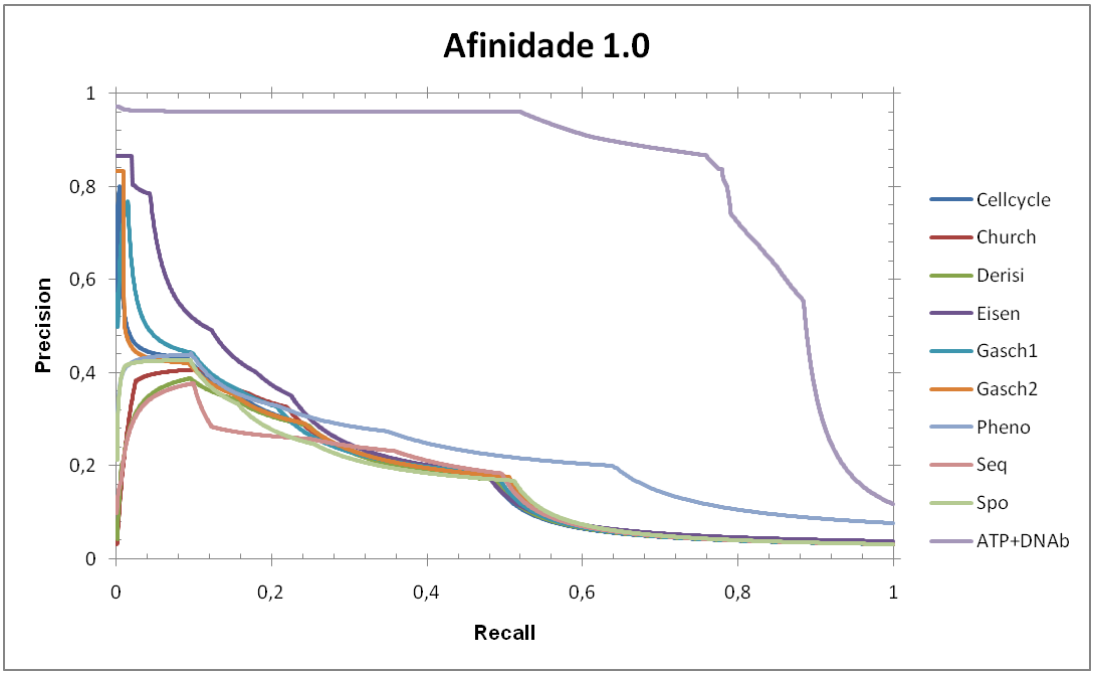
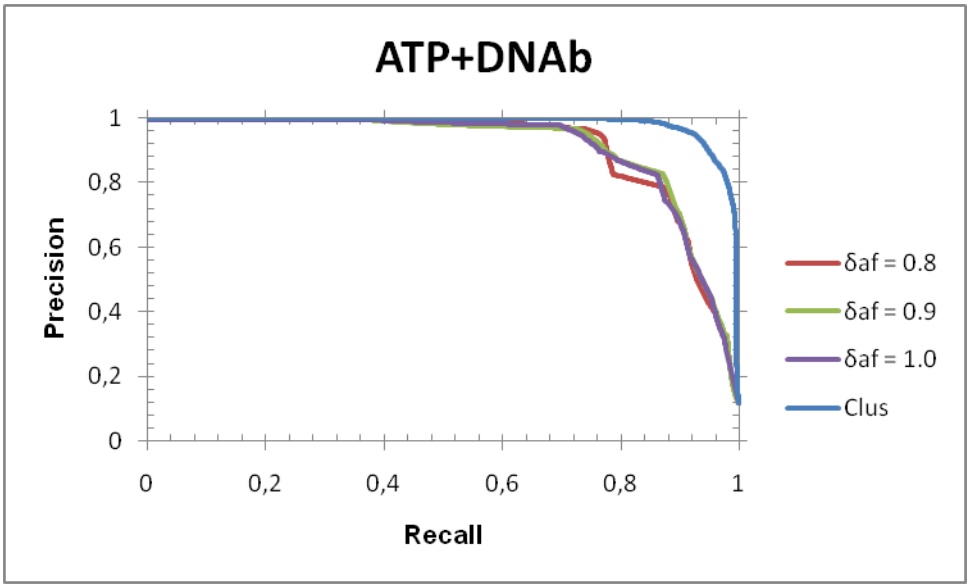
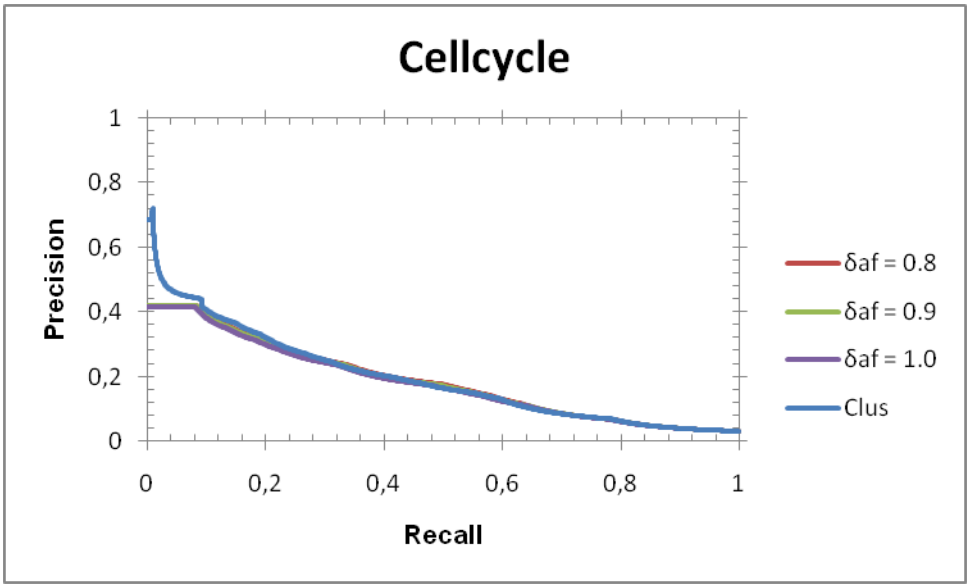


Figura 36: Curvas de precision-recall obtidas pelo MHCAIS-Local $\delta_{AF=1,0}$ sobre todas as bases avaliadas.

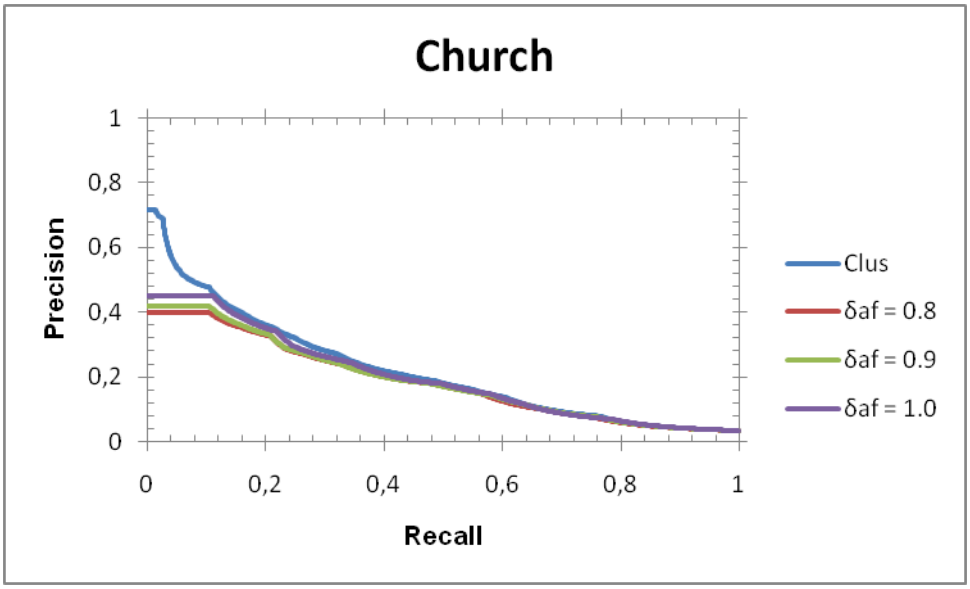


(a) AUPRC: base ATP+DNAb

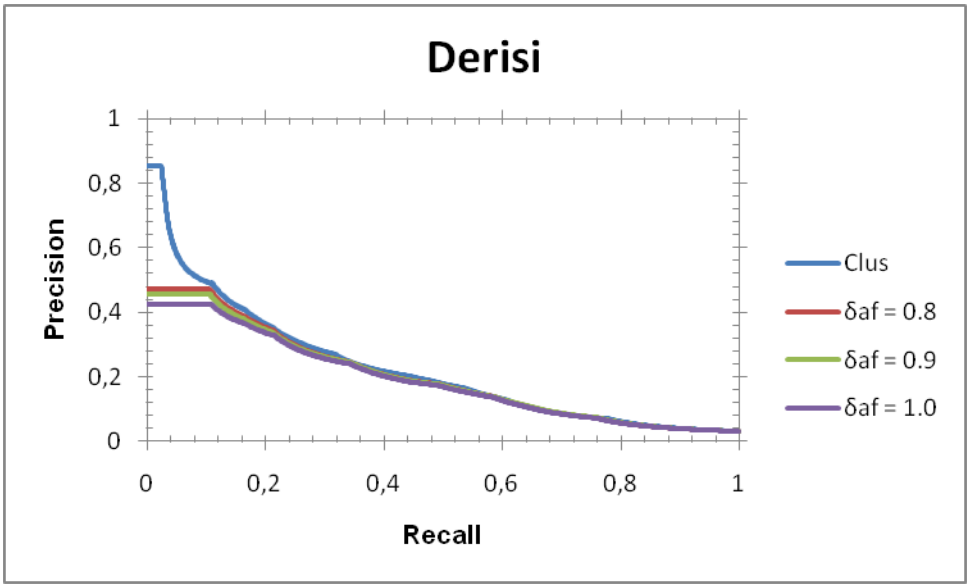


(b) AUPRC: base Cellcycle

Figura 37: Curva de *Precision-Recall* obtidas para o MHCAIS-global e o CLUS: Bases ATP+DNAb e Cellcycle.

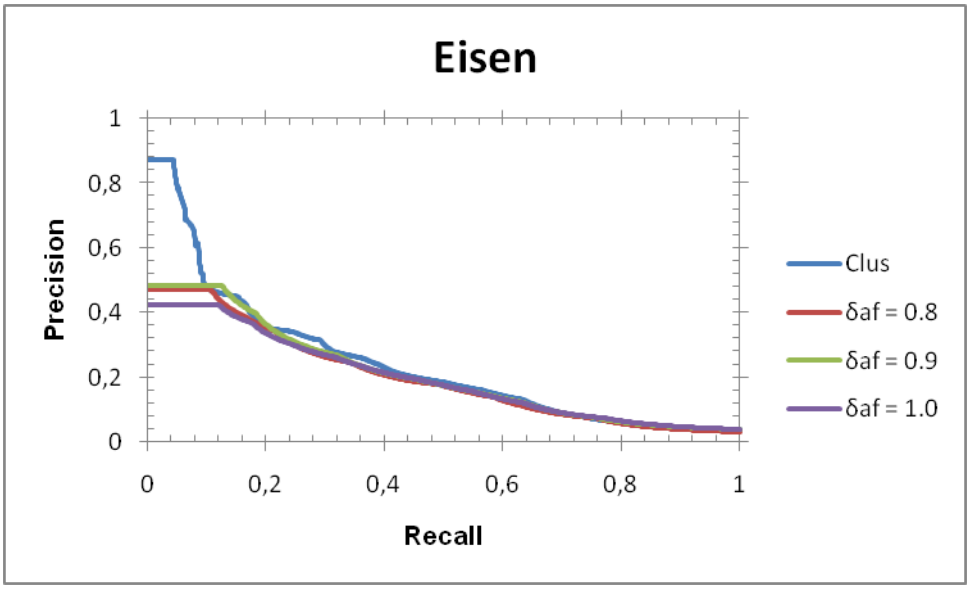


(a) AUPRC: base Church

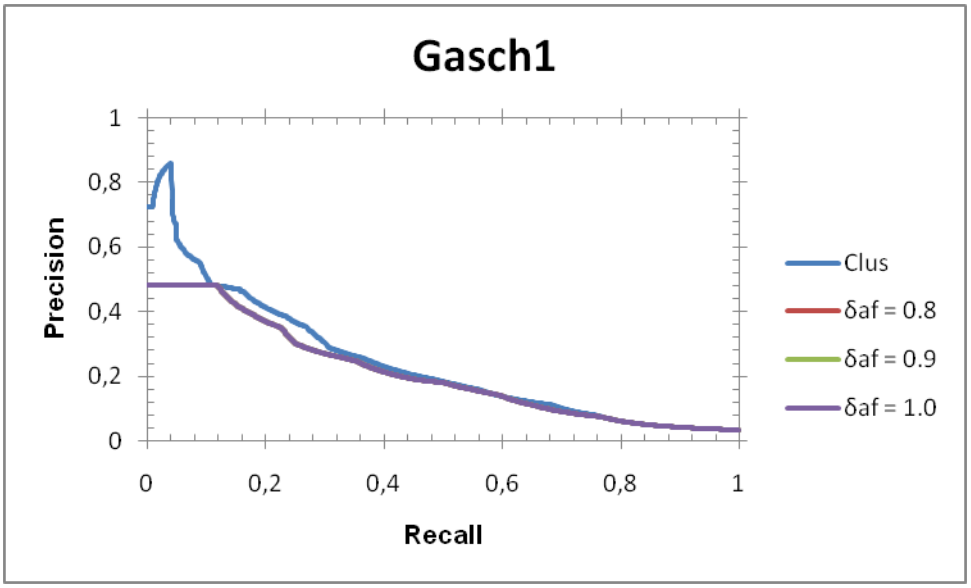


(b) AUPRC: base Derisi

Figura 38: Curva de *Precision-Recall* obtidas para o MHCAIS-global e o CLUS: Bases Church e Derisi.

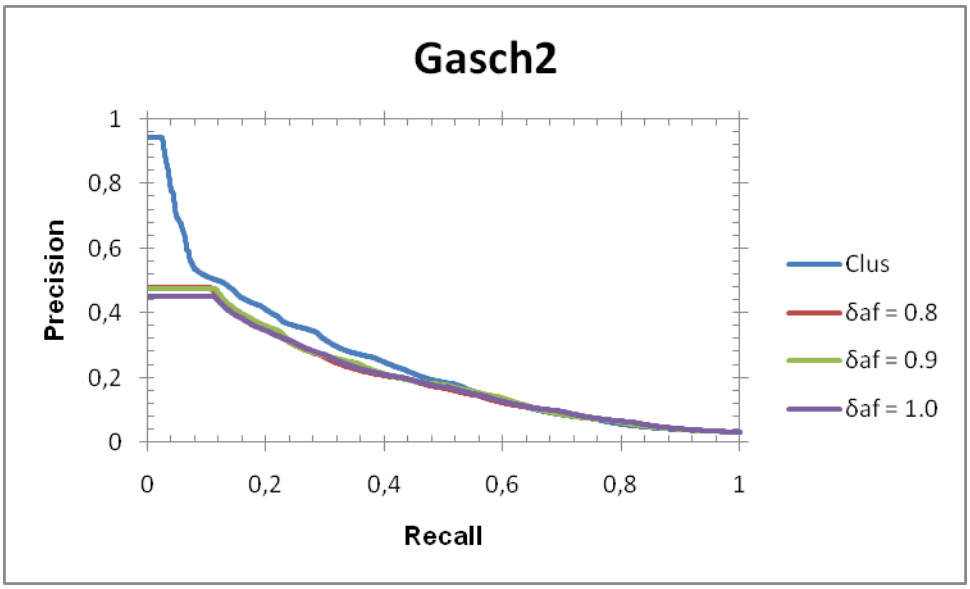


(a) AUPRC: base Eisen

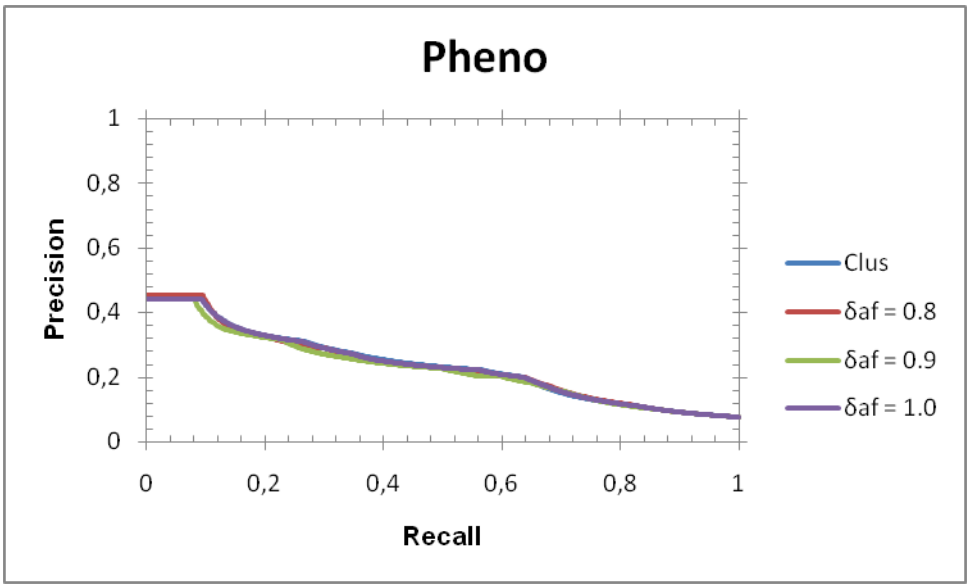


(b) AUPRC: base Gasch1

Figura 39: Curva de *Precision-Recall* obtidas para o MHCAIS-global e o CLUS: Bases Church e Derisi.

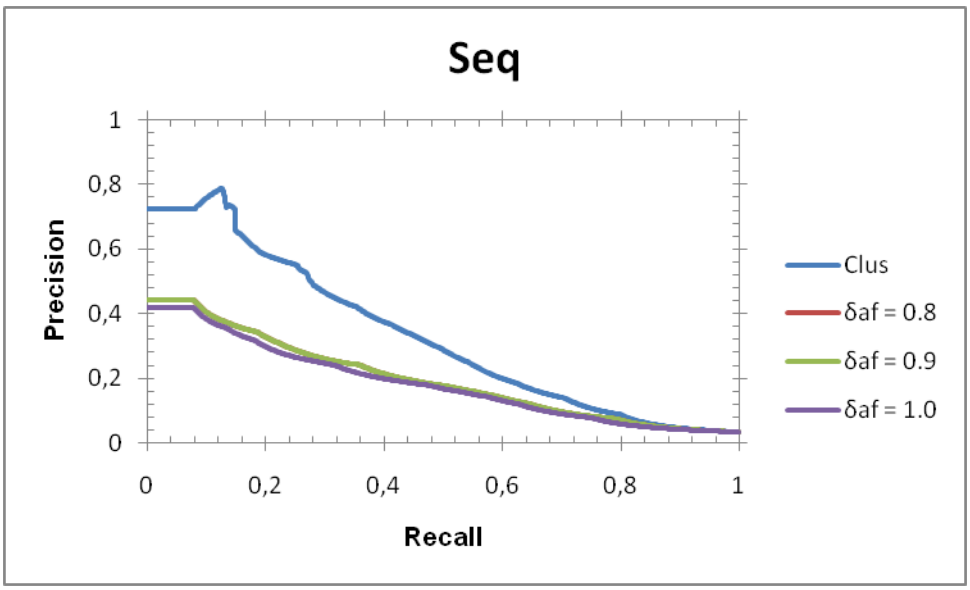


(a) AUPRC: base Gasch2

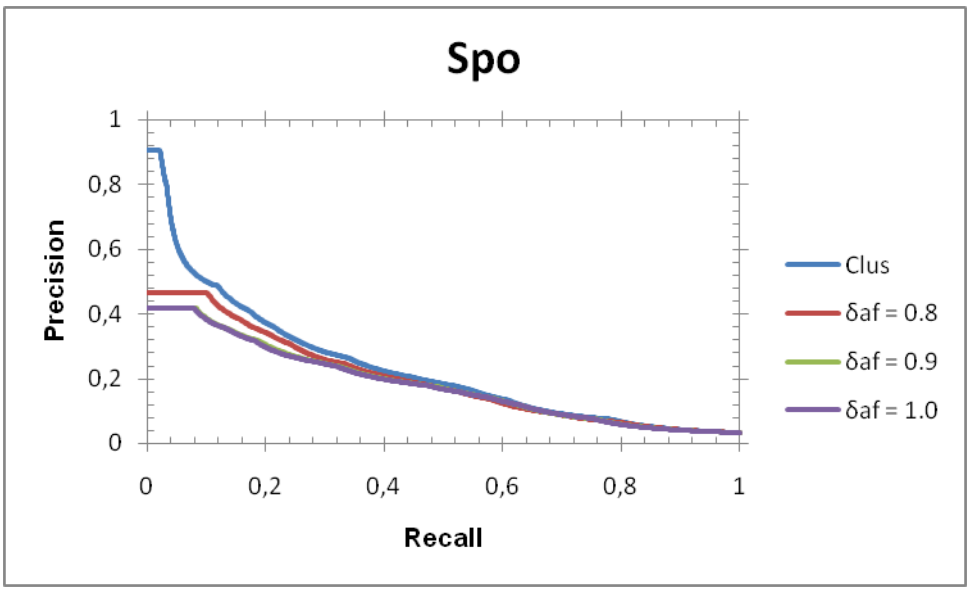


(b) AUPRC: base Pheno

Figura 40: Curva de *Precision-Recall* obtidas para o MHCAIS-global e o CLUS: Bases Gasch2 e Pheno.



(a) AUPRC: base Seq



(b) AUPRC: base Spo

Figura 41: Curva de *Precision-Recall* obtidas para o MHCAIS-global e o CLUS: Bases Seq e Spo.

ANEXO A – ARTIGO WAAMD (2007)

Discovering Multi-Label Hierarchical Classification Rules for Protein Function Prediction

Roberto Alves[1], Myriam Delgado[1], Fernando Camargo[1], Elaine Benelli[2], Alex Freitas[3]

[1]Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, UFTPR
Av. Sete de Setembro, 3165, CEP: 80230-901, Curitiba – PR – Brazil

[2]Departamento de Bioquímica e Biologia Molecular , UFPR
Cx. Postal 19046, CEP: 81531-990, Curitiba – PR – Brazil

[3]Computing Laboratory and Centre for BioMedical Informatics, University of Kent
CT2 7NF, Canterbury, U.K.

ralves2006@gmail.com, fernubio@yahoo.com, myriamdelg@utfpr.edu.br
benelli@ufpr.br, A.A.Freitas@kent.ac.uk

Abstract. *This work proposes the use of an Artificial Immune System (AIS) for predicting biological protein functions described in the gene ontology. The main challenge of this work is to discover a set of classification rules that are both hierarchical and multi-label, in the sense that a single classification rule can assign to a protein several classes (labels) in several levels of the gene ontology's hierarchy. The performance of the proposed algorithm is evaluated considering 3 protein data sets.*

1. Introduction

This work addresses the multi-label hierarchical classification task of data mining, where the goal is to discover a classification model that predicts one or more classes – at each hierarchical level – for an example (data instance). Note that in a conventional single-label flat classification task, the classification model must predict just one class for an example. Multi-label hierarchical classification is considerably more difficult than single-label classification, because there are many more classes to be assigned to an example and the hierarchical level of each class must be also considered.

In essence, Bioinformatics involves the development of computational methods (including data mining methods) for the analysis of biological data. This paper focuses on a major bioinformatics problem, viz. the prediction of protein functions from protein sequence data. This is a very challenging problem in general, due to the very complex relationship between a protein's primary sequence and its biological function [Devos, and Valencia, 2000].

Protein functions are comprehensively defined in a structured, standardized dictionary of terms called the Gene Ontology (GO) [The Gene Ontology Consortium, 2004]. GO actually consists of 3 separate "domains" (very different types of GO terms): molecular function, biological process and cellular component. The GO is structurally organized in the form of a direct acyclic graph (DAG), where each GO term represents a node of the hierarchical structure. A "child" node can have one or more parent nodes in the DAG. This work aims at assigning GO terms (classes) to proteins, where each term

represents a biological function (process). The data mining method used in this paper discovers knowledge interpretable by the user, in the form of IF-THEN classification rules, unlike other methods proposed in the literature, whose classification model is typically a "black box" which normally does not provide any insight to the user about interesting hidden relationships in the data.

2. Multi-label Hierarchical Classification with an Artificial Immune System

The algorithm used in this paper is called MHC-AIS (Multi-label Hierarchical Classification with an Artificial Immune System). MCH-AIS is an instance of the relatively new computational intelligence paradigm of AIS [De Castro and Timmis 2002]. The training phase of the algorithm is performed by two major procedures, called *Sequential Covering* (SC) and *Rule Evolution* (RE) procedures, The former is usual in induction rules algorithms [Witten e Frank 2005], whilst the latter is often used in evolutionary algorithms, e.g. AIS based on the clonal selection theory. The SC procedure iteratively calls the RE procedure until (almost) all “antigens” (proteins, examples) are covered by the discovered rules. The RE procedure essentially evolves artificial “antibodies” (multi-label classification rules, in our case) that are used to classify antigens. Each antibody (candidate classification rule) consists of two parts: the rule antecedent (IF part), represented by a vector of conditions (attribute-value pairs), and the rule consequent (THEN part), represented by a subset of predicted classes. In this work the classes correspond to GO terms denoting protein functions (see Section 3 for a practical example). The details of MHC-AIS will be described in another paper. This paper focuses instead on the application of the algorithm to a challenging bioinformatics problem. To place this algorithm in the context of the broader hierarchical classification literature, hierarchical classification methods can be divided in two approaches: local (including the top-down) and global (or big-bang) approaches. A local approach builds separate classifiers for each internal node of a hierarchy. In a global approach only one classifier is built to discriminate all classes in a hierarchy simultaneously. MHC-AIS belongs to the latter approach.

In biological databases a protein is annotated only with its most specific GO term. Given the semantics of the GO’s functional hierarchy, this implicitly means the protein also contains all the functional classes of its ancestral GO terms in the GO's DAG. Hence, MHC-AIS explicitly assigns to each antigen (protein) both its most specific class(es) (GO term(s)) and all its ancestral classes. MHC-AIS also considers the semantics of the GO’s functional hierarchy when creating classification rules – i.e., it guarantees that, if a rule predicts a given GO term, all its ancestral GO terms are also predicted by the rule.

3. Computational Results

MHC-AIS is evaluated considering 3 protein data sets. The first one contains 610 DNA-binding proteins, which are involved in gene expression [Alberts et al. 2002]. The second one contains 1411 ATPases, which catalyze ATP hydrolysis [Alberts et al. 2002]. The third one results from the union of the former two data sets. These datasets contain 11, 15 and 24 attributes, respectively. The predictor attributes in all data sets are mainly PROSITE patterns – a kind of biological motif well-known in bioinformatics [Hulo et al. 2006] – but each data set also has two other attributes, the sequence length and molecular weight. PROSITE patterns are binary attributes, indicating whether or not

the pattern occurs in a given protein's sequence of amino acids, whilst sequence length and molecular weight are continuous attributes. The number of GO terms (classes to be predicted) in each data set is 38, 57 and 78, respectively.

Recall that in data mining the discovered knowledge should be not only accurate, but also comprehensible to the user [Witten e Frank 2005]. In this spirit, the results can be evaluated according to two criteria, viz. the predictive accuracy and simplicity of the discovered rule set. In this paper, the predictive accuracy is evaluated by the F-measure, which involves computing the precision and recall of the discovered rule set on the test set (unseen during training). More precisely, the set of GO terms predicted for a test example *t*, denoted *PredGO(t)*, consists of the union of all GO terms in the consequent of all rules covering *t* – i.e. all rules whose conditions are satisfied by *t*'s attribute values. MHC-AIS computes the Precision and Recall for a test example *t* – denoted *P(t)* and *R(t)*, respectively – as per equations (1) and (2), where *TrueGO(t)* is the set of true GO terms for test example *t*.

$$P(t) = |PredGO(t) \cap TrueGO(t)| \,/\, PredGO(t) \qquad (1)$$

$$R(t) = |PredGO(t) \cap TrueGO(t)| \,/\, TrueGO(t) \qquad (2)$$

In other words, precision is the proportion of true classes among all predicted classes, whilst recall is the proportion of predicted classes among all true classes. The F-measure for a test example *t* is given by equation (3), the harmonic mean of *P* and *R*.

$$F(t) = (2 \times P(t) \times R(t)) \,/\, (1 + P(t) + R(t)) \qquad (3)$$

Finally, once *P(t)* and *R(t)* have been computed for each test example *t*, the system computes the overall F-measure over the entire test set **T** by equation (4), where |**T**| denotes the cardinality of the test set **T**.

$$\text{Predictive Accuracy} = F(\mathbf{T}) = (\Sigma_{t \in \mathbf{T}} F(t)) \,/\, |\mathbf{T}| \qquad (4)$$

The simplicity of the discovered knowledge was measured by the number of discovered rules and average number of rule conditions. The predictive accuracy and simplicity results are presented in Table 1, where the "±" symbol represents the standard deviations considering the well-known 10-fold cross-validation procedure.

Table 1. Performance of MHC-AIS on the used protein data sets

Data Set	#attrib.	#classes	predictive accuracy	#rules	#conditions by rule
DNA binding	11	38	94.39±0.79	15.30±0.39	3.80±0.10
ATPase	15	57	89.65±0.93	15.20±0.21	5.77±0.80
ATPase + DNA bind.	24	78	79.54±3.35	31.33±0.95	8.84±0.17

MHC-AIS obtained satisfactory results with respect to predictive accuracy, considering the complexity of the target problem. The worst accuracy was obtained for the ATPase + DNA-binding data set, whilst the best accuracy was obtained for the DNA-binding data set. A possible explanation for these results is that the DNA-binding data set has the smallest number of classes to be predicted (38) among the three data sets, whilst the ATPase + DNA-binding data set has the largest number of classes (78).

In any case, MHC-AIS succeeded in discovering relatively small rule sets –

particularly in the first two data sets, DNA-binding and ATPase – which facilitates the interpretation of the discovered knowledge by the user. As the third data set is the result of the former two data sets' union, a larger rule set was discovered by MHC-AIS. Considering simplicity aspects, it is important to point out that, for all data sets, on average less than 40% of predictor attributes were represented in the rule antecedents. An example of a rule discovered rule in the DNA binding data set is presented below:

IF (*PS00676* = 1) and (*PS00688* = 1) and (*SQ_LEN* $\geq$ 464) and (*MOL_WE* $\geq$ 56098)
THEN (5488 or 43167 or 46872)

The biological interpretation of this rule is: if a protein presents "sigma-54 interaction domain ATP-binding region B" and "sigma-54 interaction domain C-terminal part" signatures and "sequence length is greater than 464" and "molecular weight is greater than 56098" then the predicted classes (biological functions) are: "binding" (5488) or "ion binding" (43167) or "metal ion binding" (46872). It is essential to emphasize that the GO hierarchy was considered in the example rule consequent above, i.e. the true hierarchical path is 5488 → 43167 → 46872 (from shallower to deeper nodes).

4. Conclusion and Future Work

This work described the application of an artificial immune system-based rule induction algorithm to a challenging and important problem in bioinformatics, namely the prediction of protein function. The problem is challenging, from a data mining point of view, because it is a multi-label and hierarchical classification problem, where many classes can be assigned to an example at each level of the class hierarchy. The obtained results were considered satisfactory in terms of both predictive accuracy and simplicity of the discovered rules.

Future work will involve mainly to compare the results of MHC-AIS with other multi-label hierarchical classification algorithms and to analyze – in collaboration with a biologist – how biologically relevant and useful the discovered rules are.

References

Devos, D. Valencia, A. (2000) "Practical Limits of Functional Prediction", in *Proteins: structure, function and genetics*, 41(1), pages 98-107.

The Gene Ontology Consortium (2004) "The Gene Ontology (GO) Database and Informatics Resource", in *Nucleic Acids Research*, 32(1), pages 258-261.

Alberts, B. Johnson, A. Lewis, J. Raff, M., Roberts, K., Water, P. (2002) Molecular Biology of the Cell, Garland Science, New York, 4th Edition.

Hulo, N. Bairoch, A. Bulliard, V. Cerutti, L. De Castro, E. Langendijk-Genevaux, P. S. Pagni, M. Sigrist, C. J. A. (2006). The PROSITE Database, in *Nucleic Acids Res*., 34, pages D227-D230.

De Castro, L. N. Timmis, J. (2002) Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag, Berlin.

Witten, I. H. Frank, E. (2005) Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, San Mateo, 2nd Edition.

ANEXO B – ARTIGO BSB (2008)

Multi-Label Hierarchical Classification of Protein Functions with Artificial Immune Systems

Roberto T. Alves[1], Myriam R. Delgado[1], Alex A. Freitas[2]

[1] Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial, UFTPR
Av. Sete de Setembro, 3165, CEP: 80230-901, Curitiba – PR – Brazil
roberto.t.alves@gmail.com, myriamdelg@utfpr.edu.br
[2] Computing Laboratory and Centre for BioMedical Informatics, University of Kent
CT2 7NF, Canterbury, U.K.
A.A.Freitas@kent.ac.uk

Abstract. This work proposes two versions of an Artificial Immune System (AIS) - a relatively recent computational intelligence paradigm – for predicting protein functions described in the Gene Ontology (GO). The GO has functional classes (GO terms) specified in the form of a directed acyclic graph, which leads to a very challenging multi-label hierarchical classification problem where a protein can be assigned multiple classes (functions, GO terms) across several levels of the GO's term hierarchy. Hence, the proposed approach, called MHC-AIS (Multi-label Hierarchical Classification with an Artificial Immune System), is a sophisticated classification algorithm tailored to both multi-label and hierarchical classification. The first version of the MHC-AIS builds a global classifier to predict all classes in the application domain, whilst the second version builds a local classifier to predict each class. In both versions of the MHC-AIS the classifier is expressed as a set of IF-THEN classification rules, which have the advantage of representing comprehensible knowledge to biologist users. The two MHC-AIS versions are evaluated on a dataset of DNA-binding and ATPase proteins.

Keywords: Artificial Immune System, Hierarchical and Multi-label Classification, Prediction of Protein Function.

1 Introduction

Artificial Immune Systems (AIS) are one of the most recent natural computing approaches to emerge from computer science. The immune system is a distributed system, capable of constructing and maintaining a dynamical and structural identity, learning to identify previously unseen invaders and remembering what it has learnt. These computational techniques have many potential applications, such as in distributed and adaptive control, machine learning, pattern recognition, fault and anomaly detection, computer security, optimization, and distributed system design[1].

In data mining, ideally the discovered knowledge should be not only accurate, but also comprehensible to the user [2]. This work addresses the multi-label hierarchical classification task of data mining, where the goal is to discover a classification model

that predicts more than one class for an example (data instance) across several levels of a class hierarchy, based on the values of the predictor attributes for that example.

Bioinformatics is an inter-disciplinary field, involving the areas of computer science, mathematics, biology, etc. [3]. Among many bioinformatics problems, this paper focuses on the prediction of protein functions from information associated with the protein's primary sequence. As proteins often have multiple functions which are described hierarchically, the use of multi-label hierarchical techniques for the induction of classification models in Bioinformatics is a promising research area. At present, the biological functions that can be performed by proteins are defined in a structured, standardized dictionary of terms called the Gene Ontology (GO) [4].

The AIS algorithms proposed in this paper combine the adaptive global search of the AIS paradigm with advanced concepts and methods of data mining (hierarchical and multi-label classification), in order to solve a challenging bioinformatics problem (protein function prediction – assigns GO terms (classes) to proteins). The AIS presented in this paper discovers knowledge interpretable by the user, in the form of IF-THEN classification rules, unlike other methods proposed in the literature, whose classification model is typically a "black box" which normally does not provide any insight to the user about interesting hidden relationships in the data [5].

2. Multi-Label Hierarchical Classification

The classification task of data mining [2] consists of building, in a training phase, a classification model that maps each example t_i to a class $c \in \mathcal{C}$ of the target application domain, with $i = 1, 2, ..., n$, where n represents the number of examples in the training set.

The majority of classification algorithms cope with problems where each example t_i is associated with a single class $c \in \mathcal{C}$. These algorithms are called single label. However, some classification problems are considerably more complex because each example t_i is associated with a subset of classes $C \in \mathcal{C}$ of the application domain. Protein function prediction is a typical case of this type of problem, since a protein can perform several biological functions. Algorithms for coping with this kind of problem are called multi-label [6].

There has been a very large amount of research on conventional “flat” (non-hierarchical) classification problems, where the classes to be predicted are not hierarchically organized. However, in some problems the classes are hierarchically organized, which makes the classification problem much more challenging. Problems of this type are known as hierarchical classification problems [7].

In hierarchical classification problems, typically the classes are hierarchically organized in one of the following two forms: as a tree (where each class has at most one parent class) or as a direct acyclic graph (DAG), where each class can have more than one parent. In bioinformatics, two of the most well-known hierarchical structures for classifying protein functions are the enzyme commission hierarchy [8] – organized in the form of a tree and GO [4] – organized in the form of a DAG. The GO consists of a dictionary that defines gene products independent from species. GO actually consists of 3 separate "domains" (very different types of GO terms): molecular function, biological process and cellular component. The GO is structurally organized

in the form of a direct acyclic graph (DAG), where each GO term represents a node of the hierarchical structure.

In hierarchical classification, there are basically two types of classifiers that can be built to cope with the full set of classes to be predicted: local or global classifiers. In local classifiers, for each class $c \in \mathcal{C}$ a (local) classifier is built to predict whether or not each class c is associated with an example t_i. After all classifiers are built, an example t_i is submitted to all those classifiers (one for each class) in order to determine which classes are predicted for that example. In global classifiers, a single (global) classifier is built to discriminate among all classes of the application domain and so t_i is submitted to a single (potentially very complex) classifier [7].

3. Multi-label Hierarchical Classification with an Artificial Immune System

The immune system as a biological complex adaptive system has provided inspiration for a range of innovative problem solving techniques, including classification tasks [9] In this paper, the construction of a immune-based learning algorithm is explored whose recognition, distributed, and adaptive nature offer many potential advantages over more traditional models. The AIS algorithm used in this paper is called MHC-AIS (Multi-label Hierarchical Classification with an Artificial Immune System). MHC-AIS is based on the following natural immunology principles: clonal selection, immune network and somatic hypermutation [10,11]. In AIS, antibodies (*ab*) represent candidate solutions to the target problem, whilst antigens (*ag*) represent specific instances of the problem. In the context of this work, *ab´*s represent IF-THEN classification rules and *ag´*s represent proteins in the training set whose classes have to be predicted by the AIS.

In essence, in the clonal selection theory antibodies are cloned in proportion to their degree of matching ("affinity") to antigens, so that the antibodies which are better in recognizing antigens produce more clones of themselves. The just-generated clones are then subject to a process of somatic hypermutation, where the rate of mutation applied to a clone is inversely proportional to its affinity with the antigens. In computer science terms, the best antibodies are cloned more often and undergo a smaller rate of mutation (have fewer parts of their candidate solution modified) than the worst antibodies. With time this process of clonal selection and hypersomatic mutation leads to better and better candidate solutions to the target problem.

In essence, the theoretical immunology principle of immune networks states that antibodies can recognize not only antigens but also other antibodies. The first kind of recognition stimulates antibody production, but the latter suppresses antibodies, which in computer science terms mean a candidate solution tends to suppress other similar candidate solutions, which has the effect of improving the diversity of the search for a (near-)optimal candidate solution.

The training phase MHC-AIS is performed by two major procedures, called Sequential Covering (SC) and Rule Evolution (RE) procedures. The SC procedure iteratively calls the RE procedure until (almost) all “antigens” (proteins, examples) are covered by the discovered rules. The RE procedure essentially evolves artificial

"antibodies" (IF-THEN classification rules) that are used to classify antigens. Then, the best evolved antibody is added to discovered rule set. Each antibody (candidate classification rule) consists of two parts: the rule antecedent (IF part), represented by a vector of conditions (attribute-value pairs), and the rule consequent (THEN part) that represents the classes predicted by the rule. In this work the classes correspond to GO terms denoting protein functions. This work proposes two versions of the MHC-AIS, viz.: local and global versions (more details in the following subsections).

3.1 Global Version

In biological databases a protein is annotated only with its most specific GO term. Given the semantics of the GO's functional hierarchy, this implicitly means the protein also contains all the functional classes of its ancestral GO terms in the GO's DAG. Hence, in a data preprocessing step, MHC-AIS explicitly assigns to each antigen (protein) both its most specific class(es) (GO term(s)) and all its ancestral classes. MHC-AIS also considers the semantics of the GO's functional hierarchy when creating classification rules – i.e., it guarantees that, if a rule predicts a given GO term, all its ancestral GO terms are also predicted by the rule.

Fig. 1 shows the high-level pseudocode of the SC procedure.

```
Input: full protein training set;
Output: set of discovered rules;
DiscoveredRuleSet = ∅;
TrainSet = {set of all protein training examples};
Re-label TrainSet regarding GO's functional class hierarchy;
WHILE |TrainSet| > MaxUncovExamp
       BestRule = RULE-EVOLUTION(TrainSet); //based on AIS
       DiscoveredRuleSet = DiscoveredRuleSet U BestRule;
       updateCoveredClasses(TrainSet, BestRule)
       removeExamplesWithAllClassesCovered(TrainSet);
END WHLE
```

Fig. 1. Sequential Covering (SC) procedure.

First, it initializes the set of discovered rules with the empty set and initializes the training set with the set of all original training examples. Next, each example in the training set is extended to contain both the original class and all its ancestral classes in the GO hierarchy. Thereafter, the algorithm starts a WHILE loop which, at each iteration, calls the RE procedure. The latter receives, as parameters, the current training set and use AIS algorithm to discover classification rules. The RE procedure returns the best classification rule discovered by the AIS for the current training set. Then the SC procedure adds that rule to the discovered rule set and removes the training examples covered by that rule, as follows. In conventional rule induction algorithms for single-label classification, examples correctly covered by the just discovered rule are removed from the training set. However, in multi-label classification this process is more complex, since different rules and different training examples have different numbers of classes. In the global version of the AIS, the process of example removal works as follows. First, the training examples covered by

the just-discovered rule (i.e. examples satisfying the rule's antecedent) are identified. For each of those examples, its annotated (true) classes which are predicted by the just-discovered rules are marked as covered. As more and more rules are discovered, more and more of the annotated classes of each example will be covered. Only when all the classes of an example are covered that example is removed from the training set. The process of rule discovery terminates when the number of examples in the current training set becomes smaller than a user-defined parameter called *MaxUncovExamp*. Such procedure avoids the discovery of rules covering too few examples, unlikely to generalize well to the test set [12].

Fig. 2 shows the high-level code of the RE procedure, where rules are obtained by the proposed MHC-AIS. First, the initial population of antibodies $AB_{t=0}$ is created, where the consequent of each rule contains (initially) all GO classes in the data being mined. At the end of the evolutionary process, the AIS updates the consequent of the discovered rule (to be returned by the RE procedure) to contain only a subset of classes, representing the classes predicted by that rule, as will be explained later.

```
Input: current TrainSet;   Output: the best evolved rule;
AB_{t=0} = Create initial population of antibodies at random;
Computefitness(AB_{t=0},TrainSet);
FOR t = 1 to Number of Generations
       CL = ProduceClones(AB_{t-1});
       CL* = MutateClones(CL);
       AB_t = AB_{t-1} U CL*;
       Computefitness(AB_t,TrainSet);
       Suppresion(AB_t);
       Elitism(AB_{t+1});
END FOR t;
Determine the final subset of classes of the best antibody found so far;
return(best antibody);
```

Fig. 2. Rule Evolution (RE) procedure.

After its creation, the fitness (quality measure) of each antibody $ab_i^{t=0}$ of the initial population is calculated on the training set, where each example represents an antigen ag_j. The fitness of each ab_i is computed in two stages. First, a fitness value is associated with each *k*th-class c_k^i contained in the consequent of rule (antibody) ab_i. The value of this fitness is computed according to the following equation:

$$fit\left(c_k^i\right) = \frac{TP_{c_k^i}}{TP_{c_k^i} + FN_{c_k^i}} \times \frac{TN_{c_k^i}}{TN_{c_k^i} + FP_{c_k^i}} \quad (\mathbf{1})$$

where:

- TP (true positives) = number of training examples satisfying affinity(ab_i,ag_j) $\geq \delta_{AF}$ and having the annotated class c_k^i.
- TN (true negatives) = number of training examples satisfying affinity(ab_i,ag_j) $< \delta_{AF}$ and not having the annotated class c_k^i.
- FP (false positives) = number of training examples satisfying affinity (ab_i,ag_j) $\geq \delta_{AF}$ and not having the annotated class c_k^i.
- FN (false negatives) = number of training examples satisfying affinity (ab_i,ag_j) $< \delta_{AF}$ and having the annotated class c_k^i.

The function affinity (ab_i,ag_j) returns the degree of matching between the rule ab_i and the training example ag_j. The value of the parameter δ_{AF} represents the minimum degree of matching required for the antigen ag_j to be deemed as classified by the rule ab_i. It is important to note that δ_{AF} is a user-specified parameter, which gives more flexibility to the use of the algorithm, allowing the use of a partial or total degree of matching (δ_{AF} = 1.0) in the classification process. MHC-AIS is a hierarchical classification algorithm, and so it must consider the hierarchical structure of classes in the classification process, to reduce classification errors. A common hierarchical classification error occurs when a classifier correctly predicts a given class *c* for an example but does not predict an ancestral class of *c*. Recall that all the ancestral classes of a given predicted class must also be predicted by the trained classifier, due to the semantics of the class hierarchy in the GO. Some hierarchical classification algorithms try to correct hierarchical classification errors after the classifier has been built, in a post-processing phase. By contrast, MHC-AIS maintains a set of consistent hierarchical classifications during the construction of the global classifier. This kind of consistency is given by equation (2):

$$fit\left(c_{k*}^{i}\right) = \max\left[\, fit\left(c_{k*}^{i}\right), fit\left(c_{k}^{i}\right)\right], \quad c_{k*}^{i} \in Ancestors(c_{k}^{i}) \tag{2}$$

Hence, if the fitness of some ancestral class $c_{k*}{}^{i}$ is smaller than the fitness of its descendant class $c_k{}^{i}$, then the fitness of $c_k{}^{i}$ is assigned to its ancestral class, therefore maintaining the consistency of hierarchical classifications during training.

The fitness of an entire rule (computed as an aggregated value of the fitness of all the classes predicted by the rule) is calculated by equation (3):

$$fitness\left(ab_i\right) = \frac{1}{n} fit\left(c_{k}^{i}\right), \quad fit\left(c_{k}^{i}\right) > \delta_{FT} \tag{3}$$

where *n* indicates the number of classes c_i^k with fitness greater than the value of the parameter δ_{FT}.

Next, the AIS starts to evolve the population of antibodies. Once the global fitness of the entire rule has been calculated for each ab_i, the algorithm executes the clonal expansion process, typical in AIS [1]. Each ab_i produces *NumCl* clones of itself, where *NumCl* is proportional to the fitness of ab_i. The number of clones to be produced for each ab_i is determined by equation (4):

$$NumCl_i = \mathrm{int}\left(fitness\left(ab_i\right) \times NumMaxCl \times ClRate \right) \tag{4}$$

where the value of $NumCl \in [1,NumMaxCl]$. The parameter *NumMaxCl* represents the maximum number of clones that can be generated for a given *ab*. The function *int* truncates the fractional part of its parameter. The *ClRate* is calculated in every iteration with the goal of controlling the size of the antibody population, stimulating or inhibiting the production of clones. The value of *ClRate* is given by equation (5):

$$ClRate = \begin{cases} HyperClRate & \text{if } |AB| < nIP \\ 0 & \text{if } |AB| > nMaxP \\ 1 - \left(\dfrac{|AB| - nIP}{nMaxP - nIP} \right) & \text{otherwise} \end{cases} \quad (5)$$

where *HyperClRate*, *nIP* and *nMaxP* are specified in the beginning of the execution of the algorithm and indicate, respectively, clonal hyper-expansion rate, initial antibody population size and maximum antibody population size. It is important to emphasize that the parameter *nMaxP* does not represent the maximum size that the antibody population *AB* can take during the evolution. Rather, it indicates that, if the size of *AB* is greater than the value of that parameter, the generation of clones proportional to antibody fitness is turned off. Next, the population *CL* of clones undergoes a process of somatic hypermutation just on the IF part of the rule. A mutation rate applied to each clone *cl* is inversely proportional to the fitness of the antibody *ab* from which the clone was produced. The mutation rate is determined by equation (6):

$$MutRate_{cl} = mutMin + (mutMax - mutMin) \times (1 - fitness(cl)) \quad (6)$$

where *MutMin* and *MutMax* indicate, respectively, the minimum and maximum mutation rates to be applied to a clone *cl*; and the function fitness(*cl*) is presented in equation (3). The *MutRate* represents the probability that each gene (rule condition – IF antecedent) will undergo mutation. The population CL^*, which is formed only by clones that underwent some mutation, is then inserted in *AB*. Other procedures are also applied to *AB* during the rule evolution procedure: suppression of antibodies and elitism. The suppression procedure, characteristic of AIs based on the immune network theory, removes from AB_t similar antibodies. More precisely, if two antibodies ab_i and ab_i^* have a similarity degree greater than or equal to the value of δ_{SIM}, then, out of those two antibodies, the one with the smallest fitness is removed. The degree of similarity between two antibodies is computed as the number of conditions (attribute-value pairs) in the rule antecedents of both antibodies divided by the number of conditions in the rule antecedent of the antibody with the greatest number of conditions – which produces a measure of antibody similarity normalized in the range from 0 (no rule conditions in common) to 1 (identical rule antecedents). Elitism, a mechanism quite common in evolutionary algorithms [13], selects the antibody with the best fitness to be included in the next-iteration population AB_{t+1}.

During the rule evolution procedure all the classes occurring in the data being mined are represented in the consequent. The choice of the final subset of classes to be assigned to the consequent of the best discovered rule is given by equation (7):

$$PC = \bigcup c_k \in C \mid fit(c_k) > \delta_{FT} \quad (7)$$

where PC represents the set of classes predicted by the best discovered rule whose fitness value is greater than δ_{FT}.

3.2 Local Version

Like the global MHC-AIS, the local MHC-AIS consists of the SC (Fig. 3) and RE procedures, but with some differences. In the local version, the SC procedure labels the training examples as positive or negative. Positive examples represent examples associated with the class of the current node of the GO's DAG (a classifier is trained for each node of the GO's DAG), denoted class Y, whilst examples that do not have the class Y are labeled as negative examples. MHC-AIS is an algorithm for constructing hierarchical classifiers, and therefore the hierarchical structure has to be coped with like in the global version. Hence, all training examples labeled with any descendant class or ancestor class of the current class Y are labeled as positive class. Concerning the latter type of positive examples, it is often the case that, when a hierarchical classifier is being built, examples annotated with an ancestor class of the current class Y are removed, since they are considered as ambiguous – they do not have an annotation suggesting that they have class Y, but maybe they actually have class Y, which was not annotated yet simply due to the lack of evidence for its presence (note that "absence of evidence is different from evidence of absence"). However, in this work we use examples with an annotated class that is an ancestral of the current class Y in order to increase the number of positive examples and so hopefully increase the predictive accuracy of the algorithm.

```
Input: full training set; Output: set of discovered rules;
DiscoveredRuleSet = ∅;
FOR EACH class c
       TrainSet = {set of all training examples};
       WHILE |TrainSet| > MaxUncovExamp
           BestRule = RULE-EVOLUTION(TrainSet, class c);//based on AIS
           DiscoveredRuleSet=DiscoveredRuleSet U BestRule;
          TrainSet = TrainSet – {examp. correctly covered by BestRule};
       END WHILE;
END FOR EACH class;
```

Fig. 3. Sequential Covering (SC) procedure for Local Version

In this local version, MHC-AIS first discovers as many classification rules as necessary in order to cover the positive examples. Next, the algorithm discovers as many rules as necessary to cover the negative examples. Every time that a given rule is discovered, all the examples correctly covered by that rule (i.e. examples satisfying the conditions in the rule antecedent and having the class predicted by the rule consequent) are removed from the current training set, as usual in rule induction algorithms. This iterative process of rule discovery and removal of training examples is repeated until the number of examples in the current training set becomes smaller than a user-defined threshold *MaxUncovExamp*.

The other procedures of the local MHC-AIS are the same as in the global version of the algorithm, described in the previous subsection.

4. Computational Results

The two versions of the MHC-AIS were evaluated on a dataset of proteins created from information extracted from the well-known UNIPROT database [14]. This dataset contains two protein families: DNA-binding and ATPase [15]. These two protein families were chosen for our experiments because there are many proteins that belong to both families, increasing the difficult of the problem of building a multi-label classifier. The dataset used in the experiments contains 7877 proteins, where each protein (example) is described by 40 predictor attributes, 38 of which are PROSITE[1] patterns and 2 of which are continuous attributes (molecular weight and the number of amino acids in the primary sequence). In total, the dataset contains 214 classes (GO terms) to be predicted.

As previously discussed, in data mining the discovered knowledge should be not only accurate, but also comprehensible to the user [2,5]. In this spirit, the results can be evaluated according to two criteria, viz. the predictive accuracy and simplicity of the discovered rule set. In this paper, the predictive accuracy is evaluated by the F-measure (adapted to the scenario of multi-label hierarchical classification), which involves computing the precision and recall of the discovered rule set on the test set (unseen during training). Interpretability will be measured in terms of the size of the discovered rule set, an approach which is not ideal but is still used in the literature.

In the global version, the set of GO terms predicted for a test example *t*, denoted *PredGO(t)*, consists of the union of all GO terms in the consequent of all rules covering *t* – i.e. all rules whose conditions are satisfied by *t*'s attribute values.

In the local version of MHC-AIS, each test example *t* is submitted to the *n* trained classifiers. Each classifier consists of a set of discovered rules. The class predicted by each classifier is the class represented in the consequent of the rule with the greatest fitness value (computed during training) out of all rules discovered by that classifier that cover the example *t*. If no discovered rule covers the example *t*, the latter is classified by the default rule, which predicts the majority class in the training set. Hence, *PredGO(t)* consists of all GO terms whose trained classifiers predicted their corresponding positive class for the example *t*.

MHC-AIS computes the Precision and Recall for a test example *t* – denoted *P(t)* and *R(t)*, respectively – as per equations (8) and (9), where *TrueGO(t)* is the set of true GO terms for test example *t*.

$$P(t) = |PredGO(t) \cap TrueGO(t)| \,/\, PredGO(t) \quad (\mathbf{8})$$

$$R(t) = |PredGO(t) \cap TrueGO(t)| \,/\, TrueGO(t) \quad (\mathbf{9})$$

Thus, precision is the proportion of true classes among all predicted classes, whilst recall is the proportion of predicted classes among all true classes. The F-measure for a test example *t* is given by equation (10), the harmonic mean of *P* and *R*.

$$F(t) = (2 \times P(t) \times R(t)) \,/\, (1 + P(t) + R(t)) \quad (\mathbf{10})$$

[1] PROSITE patterns are motifs well-known in bioinformatics [16] and they are represented as binary attributes – i.e., each attribute indicates whether or not the corresponding PROSITE pattern occurs in the sequence of amino acids of a protein.

Finally, once *P(t)* and *R(t)* have been computed for each test example *t*, the system computes the overall F-measure over the entire test set **T** by equation (11), where |**T**| denotes the cardinality of the test set **T**.

$$\text{Predictive Accuracy} = F(\mathbf{T}) = (\Sigma_{t \in \mathbf{T}} F(t)) / |\mathbf{T}| \quad (\mathbf{11})$$

Table 1 shows the predictive accuracy for precision, recall and F-measure for global and local version. The numbers after the "±" symbol represent the standard deviations associated with a well-known 10-fold cross-validation procedure [2]. In the columns F-measure, the best result (out of both version of MHC-AIS) is shown in bold. The results presented in Table 1 consider different affinity (matching) thresholds for both versions of MHC-AIS, to evaluate the predictive performance of the algorithms using partial matching ($\delta_{AF} < 1.0$) or total matching ($\delta_{AF} = 1.0$).

Table 1. Predictive accuracy (%) of MHC-AIS versions on the used protein data set.

Affinity	Global Version			Local Version		
Threshold	Precision	Recall	F-Measure	Precision	Recall	F-Measure
0.8	45.93±2.71	98.23±0.61	**58.35±2.23**	80.58±1.01	44.65±1.59	55.65±1.45
0.9	50.79±3.18	92.86±3.76	58.34±2.86	75.61±1.12	52.57±2.35	**59.75±1.77**
1.0	28.91±1.31	99.50±0.12	42.84±1.37	58.56±1.01	69.91±1.13	**61.37±0.82**

Table 1 shows that the global MHC-AIS performed worst (according to the F-measure) when using total matching. Note that the global MHC-AIS obtained the worst results for the precision measure with all affinity threshold values. By contrast, the global MHC-AIS obtained very good recall values with all affinity thresholds. This performance behavior of global MHC-AIS indicates that the trained global classifier has a bias favoring the prediction of a large number of classes, mainly because the set of classes predicted for a test example consists of the union of all classes in the consequents of all rules covering that example - regardless of the fitness of the individual rules in question and the fact that the predictions of some of those rules might be inconsistent with each other. This tends to predict more classes than the actual number of true classes for a given test example, which tends to increase recall but reduce precision (given the definition of these terms).

In both cases of MHC-AIS, as the value of the affinity threshold δ_{AF} increases the value of precision is reduced, showing a disadvantage in the use of total matching. As expected, due to the trade-off between precision and recall, the local version of the algorithm had the opposite performance behavior in the case of recall, where the largest value was obtained with total matching.

Table 2 shows the results of both local and global versions of MHC-AIS with respect to the simplicity (interpretability) of the discovered rule set. This simplicity was measured by the number of discovered rules and total number of rule conditions (in all rules). The averages were computed over 10-fold cross-validation.

Table 2. Simplicity of the discovered rule set of MHC-AIS versions.

Threshold	Global Version		Local Version	
Affinity	#rules	#Conditions	#rules	#Conditions
0.8	**63.90±1,59**	**1164,30±28.20**	788.00±3.68	2901.30±42.83
0.9	**58.09±3.08**	**1066.60±53.39**	1016.80 ±8.09	4829.80±67.44
1.0	**79.90±1.83**	**1361.00±41.16**	1232.90±16.07	7069.53±18298

Note that, as shown in Table 2, the global MHC-AIS obtained much better results concerning rule set simplicity than the local MHC-AIS, in all experiments. This advantage of the global MHC-AIS is probably due to the fact that, by building a single set of rules predicting all classes in a single run of the algorithm, the algorithm can avoid the need for discovering redundant rules covering the same set of true classes for some examples. In particular, when the local version discovers rules predicting the "negative" class at each node of the GO's DAG, it should be noted that those rules predicting the negative class tend to be redundant with respect to rules predicting positive classes in other nodes of the GO's DAG, since some of the negative class examples for a given GO node will inevitably be positive class examples in another GO node. An example of a rule discovered rule by global MHC-AIS in the used data set is presented below:

IF (PS00676 == 1) and (PS00390 == 1) and (MOLECULAR_WEIGHT < 29353)
then (5488, 5515, 51087)

The biological interpretation of this rule is: if a protein presents "Sigma-54 interaction domain signatures and profile" and "Sodium and potassium ATPases beta subunits signatures" signatures and "molecular weight is less than 29353" then the predicted classes (biological functions) are: "binding" (5488) and "protein binding" (5515) and "chaperone binding" (51087). Note that the GO hierarchy was considered, i.e. the true hierarchical path is 5488 → 5515 → 51087 (from shallower to deeper nodes).

5. Conclusion and Future Work

This work described an artificial immune system (AIS)-based rule induction algorithm to the prediction of protein function. The paper proposed two versions of the AIS algorithm, a global version, where a single global classifier is built predicting all classes of the application domain; and a local version, where a local classifier is built for each node of the GO class hierarchy. Both versions have the advantage of discovering IF-THEN classification rules, constituting a type of knowledge representation that can, in principle, be easily interpretable by biologist users. The global and local versions of the AIS have different (roughly dual) advantages and disadvantages with respect to predictive accuracy, but the global version at least has the advantage of discovering much simpler (smaller) rule sets.

Future work involves: (a) comparing the predictive performance of both versions of the AIS with other classification algorithms designed for hierarchical classification

(e.g. [17]); (b) investigating new criteria for selecting, out of all classes in the consequent of the rules covering a test example in the global approach, which classes should be actually predicted for the test example; (c) incorporating an explicit mechanism during the training phase to improve the rules´ interpretability (d) analyzing the biological relevance of the discovered rules; and (e) evaluating the proposed AIS in datasets of other protein families and other types of predictor attributes.

References

1. De Castro, L. N., Timmis, J. Artificial Immune Systems: A New Computational Intelligence Approach, Springer-Verlag, Berlin, (2002).
2. Witten, I,. H. Frank, E. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Mateo, 2nd Edition, (2005).
3. Fogel, G. B., Corne, D. W. Evolutionary Computation in Bioinformatics. Morgan Kaufmann Publishers, San Franciso, (2003).
4. The Gene Ontology Consortium. The Gene Ontology (GO) Database and Informatics Resource. Nucleic Acids Research, 32(1), pp. 258--261, (2004).
5. Freitas, A. A. Data Mining and Knowledge Discovery with Evolutionary Algorithms. Springer-Verlag, Berlin, (2002).
6. Tsoumakas, G., Katakis, I. Multi-Label Classification: An Overview. International Journal of Data Warehousing and Mining, 3(3), pp 1--13, (2007).
7. Sun, A., Lim, E.-P., Ng, W.-K. Performance Measurement Framework for Hierarchical Text Classification. Journal of the American Society for Information Science and Technology, 54(11) , pp 1014--1028, (2003).
8. E. Nomenclature, of the IUPAC-IUB, American Elsevier Pub. Co., New York, NY 104, (1972).
9. Freitas, A.A. and Timmis, T. Revisiting the foundations of artificial immune systems for data mining. IEEE Trans. on Evolutionary Computation, 11(4), pp. 521-540, (2007).
10. Ada G.L., Nossal G. V. The Clonal Selection Theory. Scientific American, 257, pp 50--57, (1987).
11. Jerne, N. K. Towards a Network Theory of Immune System. Ann. Immunol. (Inst. Pasteur), 125C, pp. 373--389, (1974).
12. Alves, R. T., Delgado, M. R., Lopes, H. S., Freitas, A. A. An artificial immune system for fuzzy-rule induction in data mining. In Yao, X., Burke, E., Lozano, J. A., Smith, J., Merelo-Guervos, J. J., Bullinaria, J. A., Rowe, J., Tino, P., Kaban, A., Schwefel, H. P., (Eds.), Proc. of the 8th International Conference on Parallel Problem Solving from Nature (PPSN). LNCS, vol 3242, pp 1011--1020, Springer-Verlag, (2004).
13. Goldberg, D. E. Genetic Algorithms in Search Optimization and Machine Learning. Addison-Wesley Reading, MA, (1989).
14. The UniProt Consortium. The Universal Protein Resource (UniProt). Nucleic Acids Res., Vol. 35, pp D193--D197, (2007).
15. Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K., Water, P. Molecular Biology of the Cell, Garland Science, New York, 4th Edition, (2002).
16. Hulo, N., Bairoch, A., Bulliard, V., Cerutti, L., De Castro, E., Langendijk-Genevaux, P. S., Pagni, M., Sigrist, C. J. A. The PROSITE Database, in Nucleic Acids Res., 34, pp. D227--D230, (2006).
17. Wolstencroft, K., Lord, P.W., Tabernero, P., Brass, P., Stevens, R. Protein classification using ontology classification. Bioinformatics, 22, pp 530--538, (2006).

ÍNDICE DE AUTORES

Gasch et al. (2000), 119
Gasch et al. (2001), 119
Goldberg (1989), 97
Guarino (1998), 47

Haykin (1999), 31, 175, 176
Hoos e Stützle (2005), 93
Hughey e Karplus (2001), 29, 39
Hulo et al. (2006), 44
Hunt e Cooke (1996), 62

Iman e Davenport (1980), 169
Ishibuchi e Nakashima (1999), 91
IUPAC-IUB (1972), 56

Jensen (2001), 180
Jensen et al. (2002), 65, 70
Jensen et al. (2003), 53, 54, 64, 65, 68, 73
Jerne (1974), 62
Lu et al. (2004), 68
Tu et al. (2004), 54

Karci (2004), 115
Kennedy, Eberhart e Shi (2001), 154
Kim e Bentley (2002), 60
King et al. (2003), 54, 64, 65, 69, 70, 72, 73
Kiritchenko (2005), 31, 54, 89, 112
Kumar et al. (2000), 119

Laegreid et al. (2003), 54
Lasko et al. (2005), 108
Lehninger, Nelson e Cox (2005), 35, 36, 38
Letovsky e Kasif (2003), 64, 65, 69, 70, 72, 73
Lewis (2004), 31, 47
Li e Ogihara (2003), 52
Lu et al. (2004), 54, 64, 65, 70, 71, 73
Lucas (2009), 169

Manning, Raghavan e Schütze (2008), 109
MCCALLUM (1999), 52
McCulloch e Pitts (1943), 175
Mendel e John (2002), 116
Menolascina et al. (2008), 115, 116
Mewes et al. (1999), 119
Mezyk e Unold (2009), 115, 155
Mingers (1989), 91
Minsky e Papert (1969), 176
Mitchell (1997), 40, 91
Mitchell (1997), 51, 171
Mulder et al. (2005), 46
Müller et al. (2001), 175

Nasaroui, Gonzales e Dasgupta (2002), 62
Nelder e Mead (1965), 154
Neapolitan (1990), 180

Oliver (1996), 119

Pappa, Baines e Freitas (2005), 30
Pearl (1988), 177
Pedrycz e Gomide (1998), 32, 61, 115, 154
Szafron et al. (2003), 73

Quinlan (1993), 58, 91, 171

DE RAEDT e Blockeel (1997), 121
Roth et al. (1998), 119
Russell e Norvig (2003), 93

Schapire e Singer (2000), 52
Schug et al. (2002), 54
Secker, Freitas e Timmis (2003), 62
Smith (2003), 31, 47
Smola e Schölkopf (2002), 175
Spellman et al. (1998), 119
Stansfield (1985), 35
Strachan e Andrew (2002), 35

Stryer, Tymoczko e Berg (2004), 37
Su, Shyr e Su (2005), 62
Sun, Lim e Ng (2003), 54, 56
Szafron et al. (2003), 40
Szafron et al. (2004), 67

Tarakanov, Skormin e Sokolova (2003), 60
The Gene Ontology Consortium (2004), 31, 47
The UniProt Consortium (2007), 42
Tickle et al. (1998), 177
Timmis, Neal e Hunt (2000), 62
Towell e Shavlik (1993), 177
Troyanskaya et al. (2003), 54
Tsoumakas e Katakis (2007), 31
Tu et al. (2004), 64, 66, 69, 71, 73

VAN RIJSBERGEN (1979), 89
Vapnik (1995), 172
Vens et al. (2008), 64, 67, 69, 70, 72, 73, 109, 119, 121
Vinayagam et al. (2004), 53, 54, 64, 66, 68, 70, 71, 73

Watkins e Boggess (2002), 112
Watkins, Timmis e Boggess (2004), 60, 61, 112
Weiss e Kulikowski (1991), 121
Witten e Frank (2005), 30, 50, 51, 58, 76, 87, 97, 105, 114, 180

Yu e Hou (2004), 62

Zadeh (1975), 115
Zhang e Zhou (2005), 54