MAURI FERRANDIN

CLASSIFICAÇÃO HIERÁRQUICA UTILIZANDO ANÁLISE FORMAL DE CONCEITOS

Tese apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Doutor em Informática.

CURITIBA
2012

MAURI FERRANDIN

CLASSIFICAÇÃO HIERÁRQUICA UTILIZANDO ANÁLISE FORMAL DE CONCEITOS

Tese apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná, como requisito parcial para obtenção do título de Doutor em Informática.

Área de Concentração: Ciência da Computação

Orientador: Prof. Dr. Bráulio Coelho Ávila
Co-Orientador: Prof. Dr. Júlio Cesar Nievola

CURITIBA
2012

Ferrandin, Mauri
F372c Classificação hierárquica utilizando análise formal de conceitos / Mauri
2012 Ferrandin ; orientador, Bráulio Coelho Ávila ; co-orientador, Júlio Cesar Nievola. – 2012.
106 f. : il. ; 30 cm

Tese (doutorado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2012.
Bibliografia: f. 91-97

1. Informática - Classificação. 2. Algoritmos. 3. Base de dados – Gerência. 4. Programas de computador - Precisão. 5. Biologia computacional I. Ávila, Bráulio Coelho. II. Nievola, Júlio César. III. Pontifícia Universidade Católica do Paraná Programa de Pós-Graduação em Informática. IV. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE TESE DE DOUTORADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

ÁREA DE CONCENTRAÇÃO: CIÊNCIA DA COMPUTAÇÃO

DEFESA DE TESE DE DOUTORADO Nº 015/2012

Aos 26 dias de Outubro de 2012 realizou-se a sessão pública de Defesa da Tese de Doutorado intitulada **"Classificação Hierárquica Utilizando Análise Formal de Conceitos"** apresentada pelo aluno **Mauri Ferrandin** como requisito parcial para a obtenção do título de Doutor em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Bráulio Coelho Ávila PUCPR (Orientador)	_____ *(assinatura)*	APROVADO *(aprov/reprov.)*
Prof. Dr. Júlio Cesar Nievola PUCPR (Co-orientador)	_____	APROVADO
Prof. Dr. Fabrício Enembreck PUCPR	_____	APROVADO
Prof. Dr. Emerson Cabrera Paraiso PUCPR	_____	APROV
Prof. Dr. Andre Carlos Ponce de Leon USP/SC	_____	_____
Prof. Dr. Roberto Herai Universidade da Califórnia – San Diego	_____	Aprovado

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Fabrício Enembreck
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Esta tese foi preparada com o formatador de textos LaTeX. A bibliografia é gerada automaticamente pelo BibTeX, utilizando o estilo *apalike* com modificações para o português. O estilo utilizado no documento e as modificações no estilo *apalike* foram desenvolvidas por Ronaldo Cristiano Prati.

Agradecimentos

Primeiramente, agradeço aos meus familiares, especialmente meu pai Pedro (*In memorian*) e minha mãe Zovilde, pela vida e pelos ensinamentos. A minha namorada Scheila que se mostrou paciente durante os momentos em que precisei me concentrar nos estudos.

Aos professores do PPGIa da PUCPR, em especial ao meu orientador Prof. Bráulio e ao Prof. Júlio, co-orientador desta tese.

Aos colegas de estudo do PPGIa, em especial o Osmar e o André, colegas no laboratório de agentes, ao Luiz e a Helyane colegas de estudo no tema deste trabalho.

Ao Ricardo Cerri da USP com quem pude trocar muitas ideias e informações ao longo deste trabalho e ao Prof. Roberto Herai da Universidade da Califórnia - San Diego que me ajudou na execução dos experimentos realizados.

Ao Centro Universitário Católica de Santa Catarina que me apoiou e auxiliou com uma bolsa de estudos para custeio das mensalidades.

A todas as pessoas que ao longo deste doutorado de alguma forma contribuíram para a realização deste trabalho.

Resumo

Problemas de classificação hierárquica multirrótulo na atualidade são considerados complexos e desafiadores por apresentarem grandes diferenças em relação à classificação plana. Fundamentalmente, na classificação hierárquica as classes estão organizadas em uma taxonomia em formato de árvore ou grafo acíclico dirigido que representa relações de superclasse e subclasse entre elas, o que aumenta sua complexidade. Esta tese propõe um novo modelo que emprega técnicas da análise formal de conceitos para classificação hierárquica multirrótulo, usando a abordagem de classificação global na qual o classificador processa e avalia todas as classes da hierarquia em uma única vez. Dentre as principais características da análise formal de conceitos importantes para o seu emprego em problemas de classificação hierárquica, destaca-se a sua capacidade de generalização quanto ao tipo de taxonomia utilizada para representar a hierarquia de classes, possibilitando por meio do uso de reticulados de conceitos a representação dos dados com taxonomia em árvore ou grafo acíclico dirigido. Experimentos com o modelo proposto foram realizados com dados biológicos de 8 bases de dados da Ontologia Gênica e Functional Catalogue. Os resultados foram comparados aos resultados do Clus-HMC por intermédio das medidas de precisão e revocação hierárquicas obtidas, usando as curvas precisão-revocação e a medida da área sob a curva precisão-revocação. Testes estatísticos foram utilizados para comparar os resultados dos dois classificadores que nesta primeira versão do modelo foram promissores, mostrando que as abordagens comparadas obtiveram resultados estatisticamente equivalentes.

Palavras-Chave: Análise Formal de Conceitos, Classificação Hierárquica Multirrótulo.

Abstract

Hierarchical multilabel classification problems are considered nowadays complex and challenging because they had major differences from the flat classification. Fundamentally, in hierarchical classification classes are organized in a taxonomy represented using a tree or directed acyclic graph that represents the superclass and subclass relationships between them increasing its complexity. This thesis proposes a new model that employs techniques of formal concept analysis for hierarchical multilabel classification using the global approach in which the classifier processes and evaluates all classes in the hierarchy once. The main and most important features of the formal concept analysis for this kind of problem, we can highlight the ability to generalize the type of taxonomy used to represent the class hierarchy, allowing through the use of a concept lattice the representation of data organized in both types of taxonomies, tree or acyclic directed graph. Experiments with the proposed model were performed using biological data of 8 databases of Gene Ontology and FunCat. The results were compared against the results of Clus-HMC through the measures of hierarchical precision and hierarchical recall obtained using precision-recall curves and the measure of the area under the precision-recall curve. Statistical tests were used to compare the samples from the results of the two classifiers. The results in the first version of the model were promising and show that the approaches compared results obtained statistically equivalent.

Keywords: Formal Conceptual Analysis, Hierarchical Multi-label Classification.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ARFF *Attribute-Relation File Format.*

AUPRC *Area Under Precision Recall Curve.*

DAG *Directed Acyclic Graph.*

FCA *Formal Conceptual Analysis.*

FD *Full Depth.*

FunCat *Functional Catalogue.*

GC *Global Classifier.*

GO *Gene Ontology.*

hF *Hierarchical F-measure.*

HMCS-FCA *Hierarchical Multi-Label Classifier System - Formal Conceptual Analysis.*

hP *Hierarchical Precision.*

hR *Hierarchical Recall.*

KDD *Knowledge Discovery Database.*

LCL *Local Classifier Per Level.*

LCN *Local Classifier Per Node.*

LCPN *Local Classifier Per Parent Node*.

MHC-AIS *Multi-Label Hierarchical Classification with an Artificial Immune System*.

MLNP *Mandatory Leaf-Node Prediction*.

MPL *Multiple Path of Labels*.

MPP *Multiple Path Prediction*.

NMLNP *Non-Mandatory Leaf-Node Prediction*.

PCT *Predictive Clustering Trees*.

PD *Parcial Depth*.

PR *Precision-Recall*.

RE *Rule Evolution*.

ROC *Receiver Operating Characteristics*.

SAI Sistema Imunológico Artificial.

SC *Sequential Covering*.

SPL *Single Path of Labels*.

SPP *Single Path Prediction*.

TDIDT *Top-down Induction of Decision Trees*.

WEKA *Waikato Environment for Knowledge Analysis*.

Lista de Algoritmos

Capítulo

1

Introdução

O crescimento exponencial do volume de dados disponíveis em bases de dados biológicas demandou a necessidade de biólogos, matemáticos e cientistas da computação somarem esforços na busca de algoritmos e técnicas para suportar a análise destas bases de dados, dando origem a uma área denominada bioinformática (Lesk, 2008). Grande parte destas bases de dados biológicas utiliza taxonomias hierárquicas de classes para classificar os dados. Este trabalho apresenta a proposta de um modelo interdisciplinar definido por meio do uso de técnicas de mineração de dados e a análise formal de conceitos - *Formal Conceptual Analysis* (FCA), para ser aplicado em problemas de classificação hierárquica. A etapa de mineração de dados é um campo interdisciplinar em que podem ser utilizados diferentes tipos de tarefas para a descoberta de padrões, como por exemplo, classificação, agrupamento, associação e regressão. O foco deste trabalho envolve o estudo do paradigma de classificação.

A área de mineração de dados está concentrada principalmente em problemas de classificação (Otero, 2010) e a grande maioria destes trata-se de problemas de classificação plana, no qual cada exemplo usado para treinamento e teste está associado a uma ou mais classes, mas não existe correlação entre estas classes. No entanto, existem problemas de classificação mais complexos nos quais as classes estão organizadas por meio de uma taxonomia hierárquica definindo relações de pai/filho entre elas, e os exemplos ou instâncias podem estar associados a mais do que uma única classe - estes problemas são denominados problemas de classificação hierárquica multirrótulo. O interesse dos pesquisadores neste

tipo de classificação teve origem inicialmente com os problemas de mineração em bases textuais; atualmente existe um grande número de pesquisas para o desenvolvimento de técnicas e algoritmos para predição das funções de proteínas.

Dentre os problemas que têm sido tratados pela bioinformática a predição de funções biológicas de proteínas merece destaque. De acordo com Alves *et al.* (2008), a predição da função de proteínas consiste em associar funções biológicas para proteínas, cujas funções ainda não são conhecidas. Este conhecimento pode auxiliar os pesquisadores no melhor entendimento de doenças, no desenvolvimento de fármacos, na medicina preventiva, entre outras aplicações.

Neste campo, inúmeras propostas de modelos e algoritmos para problemas de classificação hierárquica foram desenvolvidas, sendo necessário o estabelecimento de uma classificação e padronização de nomenclatura e funcionalidades para as diferentes abordagens conforme as suas características e também de acordo com o tipo de problema tratado. Esta classificação foi proposta por Silla e Freitas (2011).

Os principais pontos que distinguem um problema de classificação hierárquica dos outros são: a organização da taxonomia de classes que pode ser em formato de árvore ou grafo acíclico dirigido - *Directed Acyclic Graph* (DAG); o nível da hierarquia de classes em que os exemplos podem estar associados e serão classificados - apenas em nós folha ou em qualquer nível; o número de classes que podem estar associadas a um exemplo - apenas uma classe ou várias; a forma como o classificador explora a estrutura - local ou global; e a técnica ou algoritmo utilizado para solução do problema.

Dentre os principais modelos propostos para solução de problemas de classificação hierárquica utilizando a abordagem de classificação global com taxonomia em DAG pode-se destacar os trabalhos Kiritchenko *et al.* (2005, 2006); Alves *et al.* (2008); Vens *et al.* (2008); Otero *et al.* (2009); Wang *et al.* (2009).

A proposta deste trabalho é a criação de um modelo para classificação hierárquica multirrótulo baseado no paradigma de classificação global, com suporte à taxonomia de classes definida por meio de árvore ou DAG, e utilizando técnicas da FCA para representação dos dados associados com as taxonomias hierárquicas.

A FCA é um conjunto de técnicas baseadas em fundamentos matemáticos desenvolvidos inicialmente por um grupo da Universidade de Darmstadt, no início da década de 1980, que tinha por objetivo reestruturar a teoria de ordem matemática e a teoria dos reticulados (Wille, 1982). De maneira geral, propõe técnicas para manipular tabelas com dados representados na forma binária em que cada coluna é considerada um atributo, cada linha é considerada um objeto, e o conteúdo da tabela descreve os relacionamentos entre os objetos e atributos, relacionamentos estes que podem ser representados pelos conceitos

formais. Neste trabalho, a FCA está no centro do modelo uma vez que as suas técnicas serão empregadas para treinamento do classificador proposto.

1.1 Motivação e Descrição do Problema

Os problemas de classificação multirrótulo têm sido explorados em diversas áreas tendo como motivação o crescimento do número de aplicações, tais como, anotações semânticas em imagens e vídeo, funcionalidades genômicas, categorização de músicas e marketing direcionado (Tsoumakas *et al.*, 2010).

A predição das funções de proteínas é um problema de classificação multirrótulo no qual as classes estão organizadas de acordo com uma taxonomia hierárquica específica que estabelece uma relação de ordem entre as classes. Isto aumenta a complexidade uma vez que existem correlações entre as classes que precisam ser levadas em consideração para se obter um bom desempenho do classificador.

O uso de FCA para solução de problemas de classificação hierárquica é um campo a ser explorado. As técnicas da FCA permitem a manipulação, organização e recuperação das informações hierarquicamente estruturadas para diversas finalidades, e a sua utilização para classificação hierárquica multirrótulo será o objeto de estudo deste trabalho. O ponto chave que motiva o emprego das técnicas de FCA para classificação hierárquica é o fato de que por meio de um reticulado de conceitos, pode-se representar de forma generalizada informações organizadas hierarquicamente com base em ambos os tipos de taxonomia utilizados na classificação hierárquica: árvore e DAG. Esta facilidade de representação será explorada para criação de um modelo que é capaz de operar de forma independente da taxonomia utilizada.

1.2 Objetivos

A grande maioria dos algoritmos para a classificação hierárquica de proteínas foi desenvolvida para a estrutura do tipo árvore. Posteriormente, alguns deles foram adaptados para suportar taxonomias hierárquicas de classes em forma de DAG. Este trabalho tem como objeto de estudo o desenvolvimento de um modelo para classificação hierárquica com suporte à taxonomia de classes organizada em árvore ou DAG e usando a abordagem de classificação hierárquica global.

1.2.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver e avaliar um modelo para a classificação hierárquica multirrótulo de dados utilizando a abordagem de classificação global por meio do uso da FCA.

1.2.2 Objetivos Específicos

Os objetivos específicos são:

a) Estudar as características de um conjunto de bases de dados multirrótulo hierárquicas de proteínas disponíveis publicamente avaliando as formas e taxonomias hierárquicas usadas para sua classificação;

b) Desenvolver e implementar um classificador hierárquico multirrótulo global;

c) Aplicar o modelo desenvolvido para predição de função de proteínas;

d) Avaliar os resultados obtidos com o modelo proposto comparando-os a outra abordagem que representa o estado da arte no tema.

1.3 Solução Proposta

A FCA dispõe de um conjunto de técnicas e algoritmos para representar informações e mapear as correlações entre estas informações organizando-as em uma estrutura hierárquica denominada reticulados de conceitos. A ideia central desta proposta é a utilização das técnicas da FCA para criação de um classificador global capaz de mapear as correlações entre as instâncias de treinamento e suas classes de maneira que estas informações possam posteriormente ser usadas para classificação de instâncias não apresentadas ao modelo na etapa de treinamento.

O modelo proposto será denominado *Hierarchical Multi-Label Classifier System - Formal Conceptual Analysis* (HMCS-FCA) representando uma combinação de técnicas da FCA e *Knowledge Discovery Database* (KDD).

Para realização de experimentos e avaliação do desempenho do modelo, foram utilizadas as bases de dados da Ontologia Gênica - *Gene Ontology* (GO) (Ashburner *et al.*, 2000) e Catálogo Funcional - *Functional Catalogue* (FunCat) (Ruepp e et al, 2004), as quais possuem informações representadas e classificadas de acordo com uma taxonomia

de classes hierarquicamente organizada. Os resultados obtidos foram comparados ao classificador Clus-HMC (Vens *et al.*, 2008) que representa o estado da arte em classificação hierárquica na atualidade.

1.4 Organização

O presente trabalho está organizado em seis capítulos. O Capítulo 2 apresenta uma revisão teórica dos principais conceitos e fundamentos relacionados aos temas do trabalho, tais como classificação de dados, FCA e proteínas. O Capítulo 3 aborda os assuntos que representam o estado da arte em relação ao tema estudado, nele é apresentada uma revisão teórica sobre classificação hierárquica e um resumo dos principais trabalhos descrevendo as soluções propostas para problemas de classificação hierárquica. No Capítulo 4, é apresentado o modelo proposto neste trabalho, detalhando o HMCS-FCA e os algoritmos do classificador proposto. O Capítulo 5 relata os resultados dos experimentos realizados com o modelo proposto comparado-os a outro modelo que representa o estado da arte sobre o tema, e o Capítulo 6 apresenta as conclusões.

Capítulo

2

Fundamentos

Neste capítulo, serão apresentados os referenciais teóricos fundamentais relacionados às técnicas matemáticas e computacionais utilizadas no desenvolvimento desta tese. Na Seção 2.1, são abordados os conceitos gerais sobre classificação de dados demonstrando as diferenças fundamentais entre os tipos de problemas de classificação. Na Seção 2.2, são apresentadas as teorias relacionadas à análise formal de conceitos, seus fundamentos matemáticos e principais algoritmos existentes. Na Seção 2.3, são apresentados os fundamentos a respeito da predição das funções de proteínas e esquemas existentes para classificação hierárquica de proteínas. Os três temas apresentados nas seções propostas representam as diferentes áreas de conhecimento relacionadas neste trabalho.

2.1 Classificação de Dados

De acordo com Tan *et al.* (2009), classificação é a tarefa de aprender uma função alvo f que mapeie cada conjunto de atributos x para um dos rótulos de classes y pré-determinados. A classificação é um dos problemas mais importantes da aprendizagem de máquina e da mineração de dados (Freitas e Carvalho, 2007). Para Freitas (2002), a classificação é provavelmente o assunto mais estudado no contexto de mineração de dados.

Uma técnica de classificação (ou classificadora) é uma abordagem sistemática para construção de modelos de classificação a partir dos dados de um conjunto de dados de entrada. Exemplos incluem classificadores baseados em árvores de decisão, regras, redes

neurais, máquinas de vetor de suporte e classificadores baseados no teorema de Bayes. Cada técnica emprega um algoritmo de aprendizagem para identificar um modelo que seja mais apropriado para o relacionamento entre o conjunto de atributos e os rótulos de classes presentes nos dados de entrada.

O modelo gerado pelo algoritmo de aprendizagem deve se adaptar bem aos dados de entrada e prever corretamente os rótulos de classes para exemplos que nunca lhe foram apresentados anteriormente (Tan *et al.*, 2009). Portanto, um objetivo chave do algoritmo de aprendizagem é construir modelos com boa capacidade de generalização. A Figura 2.1 mostra uma abordagem geral para resolver problemas de classificação na qual um modelo é criado mediante o emprego de um algoritmo de aprendizagem sobre o conjunto de dados de treinamento cujos rótulos de classes são conhecidos. Posteriormente, o modelo gerado é aplicado ao conjunto de dados de teste para os quais os rótulos de classes são desconhecidos a fim de predizer as classes para cada exemplo.

Conjunto de Treinamento

Id.	Atrib 1	Atrib 2	Atrib 3	Classe
1	Sim	Grande	125K	Não
2	Não	Médio	100K	Não
3	Não	Pequeno	70K	Não
4	Sim	Médio	120K	Não
5	Não	Grande	95K	Sim
6	Não	Médio	60K	Não
7	Sim	Grande	220K	Não
8	Não	Pequeno	85K	Sim
9	Não	Médio	75K	Não
10	Sim	Pequeno	90K	Sim

Conjunto de Teste

Id.	Atrib 1	Atrib 2	Atrib 3	Classe
11	Não	Pequeno	55K	?
12	Sim	Médio	80K	?
13	Sim	Grande	110K	?
14	Não	Pequeno	95K	?
15	Não	Grande	67K	?

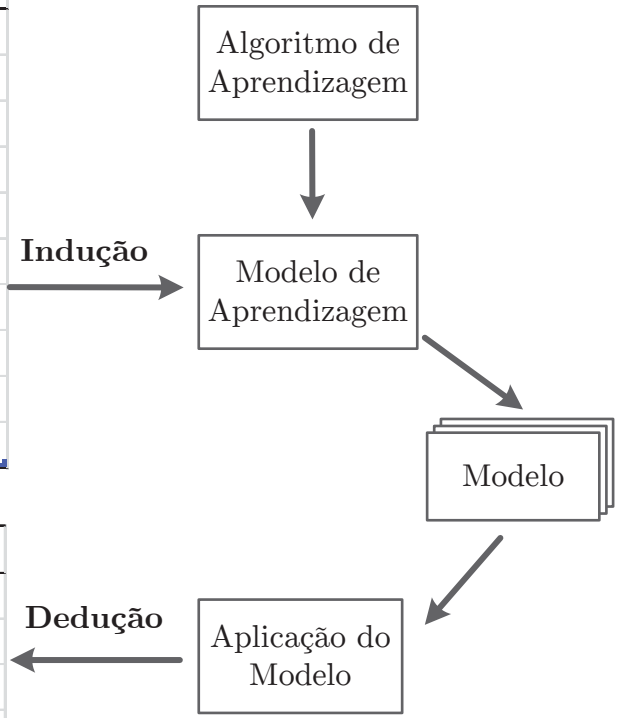


Figura 2.1: Abordagem geral para construção de um modelo de classificação de acordo com Tan *et al.* (2009).

Os problemas de classificação de dados podem ser divididos em dois tipos principais: classificação plana e classificação hierárquica. Estes dois tipos principais podem ser subdivididos em dois subtipos: monorrótulo no qual apenas uma classe é atribuída a uma instância e multirrótulo no qual n classes podem ser atribuídas a uma instância. Instância ou exemplo é a denominação dada a cada item (linha da relação) presente nos conjuntos de dados de teste ou treinamento.

A classificação plana monorrótulo pode ainda ser binária quando existem apenas duas classes no conjunto de classes que podem ser atribuídas a uma instância, ou multiclasse quando o número de classes que podem ser atribuídas a uma instância é maior que dois. A Figura 2.2 representa como podem ser classificados os problemas de classificação.

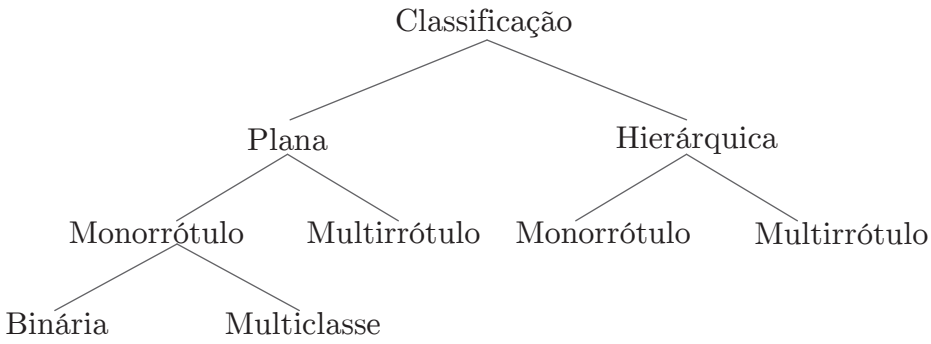


Figura 2.2: Taxonomia dos tipos de problemas de classificação.

A classificação plana é mais simples e é utilizada na maioria dos problemas citados na literatura. O conjunto de treinamento do exemplo da Figura 2.1 mostra um típico problema de classificação plana monorrótulo binária, no qual o atributo classe pode assumir apenas um dos valores do conjunto binário {Sim, Não}.

Na classificação plana multirrótulo, muito comum principalmente em aplicações de categorização de textos, várias classes podem estar associadas a uma mesma instância. Conforme Tsoumakas e Katakis (2007), na classificação plana multirrótulo, um exemplo pode estar associado a um conjunto de classes. A Tabela 2.1 exemplifica um conjunto de dados de um problema de classificação plana multirrótulo.

Uma abordagem comumente utilizada para se tratar problemas de classificação plana multirrótulo é a sua transformação em um problema de classificação plana binária, criando um novo conjunto de dados para cada classe existente no conjunto de dados multirrótulo original. Após a transformação, o atributo classe de cada conjunto de instâncias criado assumirá a forma binária indicando se a instância pertence ou não àquela classe. A Tabela 2.2 demonstra como pode ser feita a representação dos dados da Tabela 2.1 por meio de

Tabela 2.1: Exemplo de um conjunto de dados plano multirrótulo no qual uma instância pode estar associada a mais do que uma classe. Neste exemplo os atributos não classe são omitidos.

Ex.	**Classes**		
	Valor#1	Valor#2	Valor#3
1	esporte		entretenimento
2	esporte	política	
3	esporte	política	entretenimento
4		política	
5			entretenimento
6	esporte		
7		política	entretenimento
8			entretenimento
9		política	
10	esporte	política	

Tabela 2.2: Exemplo de um conjunto de dados no formato plano binário, obtido a partir de um conjunto de dados plano multirrótulo. Neste exemplo os atributos não classe são omitidos. A classe P indica a presença do assunto na instância e a classe A indica a sua ausência.

(a) esportes

Ex.	Classe
1	P
2	P
3	P
4	A
5	A
6	P
7	A
8	A
9	A
10	P

(b) política

Ex.	Classe
1	A
2	P
3	P
4	P
5	A
6	A
7	P
8	A
9	P
10	P

(c) entretenimento

Ex.	Classe
1	P
2	A
3	P
4	A
5	P
6	A
7	P
8	P
9	A
10	A

três conjuntos de dados no formato plano binário.

A transformação de um problema de classificação plana multirrótulo em classificação plana binária é computacionalmente onerosa uma vez que é necessário executar o algoritmo de classificação plana para cada classe e também faz com que sejam perdidas as correlações entre as classes associadas a uma mesma instância.

Na classificação hierárquica, que é o modelo utilizado neste trabalho, todas as clas-

ses estão dispostas em uma estrutura hierárquica previamente conhecida, o que torna a criação do modelo de classificação ainda mais complexo. Para exemplificar de forma prática, suponha-se que se está classificando artigos de jornal e que exista uma taxonomia hierárquica de classes pré-definida, conforme a representação da Figura 2.3.

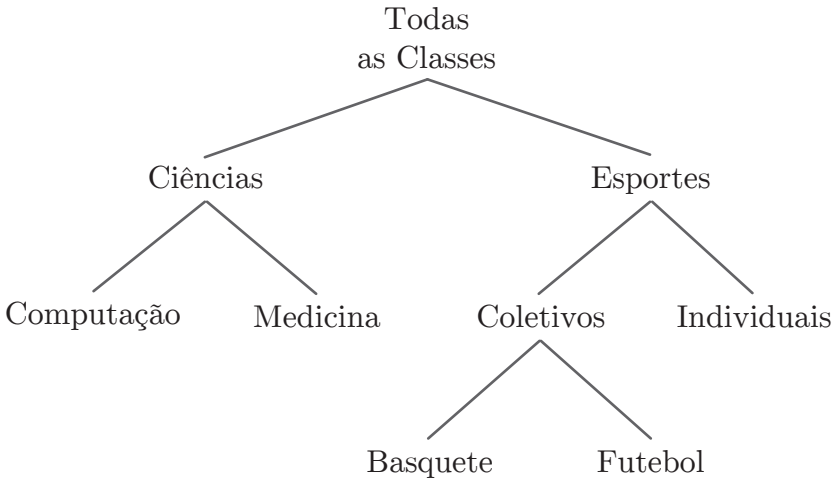


Figura 2.3: Exemplo de taxonomia hierárquica de classes. Adaptado de Cerri (2010).

De maneira geral, nos problemas de classificação hierárquica, quando se associa uma instância a uma determinada classe da hierarquia, automaticamente ela pertencerá também a todas as classes ascendentes da classe atribuída. Assim sendo, de acordo com a taxonomia da Figura 2.3, um artigo de jornal que foi associado à classe *Futebol* também pertence às classes *Coletivos* e *Esportes*, ascendentes da classe *Futebol* na taxonomia. Representa-se o conjunto de classes preditas para a instância por *Esportes/Coletivos/Futebol*, não havendo a necessidade de representar o nó raiz da taxonomia (*Todas as Classes*) pois assume-se que todas as instâncias pertencem a este nó. A Figura 2.4 representa as classes preditas na hierarquia para o artigo de jornal rotulado com a classe *Futebol*, sendo que quando as classes associadas à instância pertencem a um único ramo da hierarquia como neste caso, diz-se que se trata de um problema de classificação hierárquica monorrótulo.

Problemas de classificação hierárquica multirrótulo são problemas em que uma instância pode estar associada a várias classes e estas classes pertencem a ramos diferentes da hierarquia. Para exemplificar, suponha que um artigo de jornal discorra sobre os temas *Futebol* e *Computação* respectivamente, neste caso além do artigo pertencer às classes *Esportes/Coletivos/Futebol*, conforme representado na Figura 2.4, ele também pertence às classes *Ciências/Computação*. Como existem classes atribuídas ao artigo que pertencem

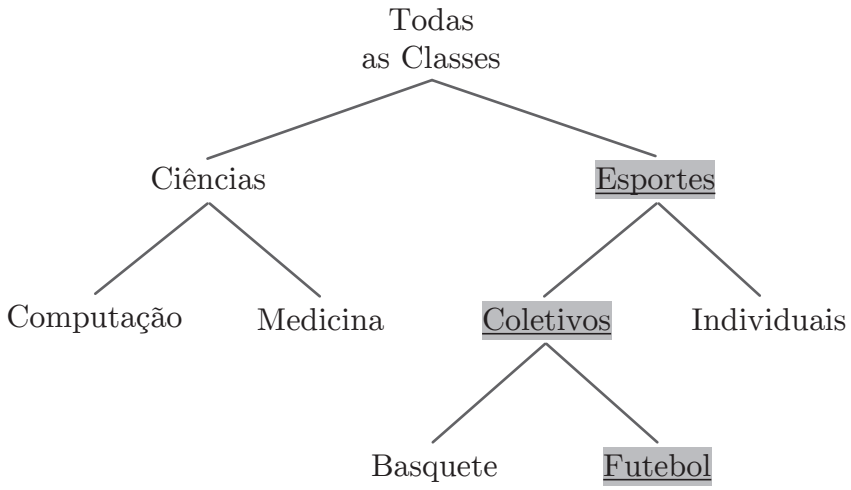


Figura 2.4: Exemplo de classes preditas para uma instância em um problema de classificação hierárquica monorrótulo.

a ramos diferentes da hierarquia, este problema é definido como um problema de classificação hierárquica multirrótulo. A Figura 2.5 representa as classes preditas na hierarquia.

De acordo com Cerri (2010), é importante que não haja confusão entre problemas de classificação hierárquica monorrótulo e problemas de classificação hierárquica multirrótulo. Pode-se erroneamente pensar em um problema de classificação hierárquica monorrótulo como sendo um problema multirrótulo, devido ao fato de que um ramo da hierarquia

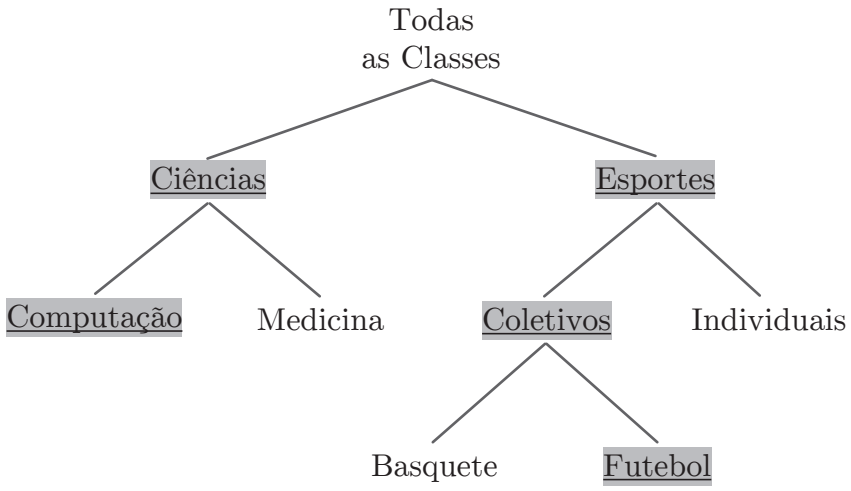


Figura 2.5: Exemplo de classes preditas para uma instância em um problema de classificação hierárquica multirrótulo.

contém mais do que uma classe. Quando se atribui a uma instância as classes *Esportes/Coletivos/Futebol*, conforme o exemplo da Figura 2.4, está se atribuindo um ramo da hierarquia que contém três classes a esta instância. Entretanto, um problema hierárquico somente é considerado multirrótulo quando a predição de classes para uma mesma instância envolve classes pertencentes a ramos distintos da hierarquia. No exemplo da Figura 2.4, apesar da instância pertencer a três classes, trata-se de um problema de classificação hierárquica monorrótulo, pois as três classes *Esportes/Coletivos/Futebol* estão em um único ramo da taxonomia.

Os conceitos envolvendo classificação hierárquica serão aprofundados no Capítulo 3 desta tese, no qual serão detalhadas as características específicas deste tipo de classificação, as técnicas já existentes para classificação de dados neste paradigma e também as medidas de avaliação para classificadores hierárquicos.

2.2 Análise Formal de Conceitos (FCA)

De maneira geral, a FCA manipula tabelas com dados representados na forma binária descrevendo os relacionamentos entre objetos e atributos, tabelas estas chamadas de contextos formais. Estes relacionamentos entre os objetos e os atributos de um contexto formal quando ordenados de acordo com algumas premissas, formam um reticulado de conceitos. Nas próximas subseções, serão abordados os principais conceitos inerentes a FCA, suas técnicas básicas, seus fundamentos, e outras técnicas que serão empregadas na construção do modelo de classificação, como por exemplo a subseção 2.2.6 que aborda o uso de conhecimento de fundo no contexto da FCA. Esta última técnica citada possibilitará imposição da hierarquia das classes nos conjuntos de dados usados para indução do classificador proposto.

2.2.1 Contexto Formal

Segundo Ganter e Wille (1999), um contexto formal $K := (G, M, I)$ é composto por dois conjuntos G e M e uma relação I entre G e M. Os elementos do conjunto G são chamados de objetos do contexto e os elementos de M são chamados de atributos do contexto. Para representar que um objeto g em uma relação I possui o atributo m, denota-se gIm ou $(g, m) \in I$, e lê-se "o objeto g possui o atributo m".

A Tabela 2.3 mostra o exemplo de um contexto formal em que estão representados o conjunto de objetos $G = \{1, 2, 3, 4\}$, o conjunto dos atributos $M = \{a, b, c, d\}$ e a relação de incidência $I = \{(1, b), (1, c), (2, a), (2, c), (2, d), (3, a), (3, b), (4, b), (4, c), (4, d)\}$ que indica

quais atributos estão presentes em cada objeto.

Tabela 2.3: Exemplo de um contexto formal.

	a	b	c	d
1	0	1	1	0
2	1	0	1	1
3	1	1	0	0
4	0	1	1	1

2.2.2 Conceito Formal

A partir de um contexto formal, é possível calcular os conceitos formais nele existentes. Na sequência demonstrar-se-ão as operações básicas para este cálculo de acordo com Carpineto e Romano (2004).

Dado um contexto formal $K := (G, M, I)$, para um conjunto de objetos $A \subseteq G$, pode-se calcular o subconjunto A' contendo os atributos comuns a todos os objetos existentes em A, conforme representado na equação 2.1:

$$A' := \{m \in M \mid gIm \text{ para todo } g \in A\} \tag{2.1}$$

De forma análoga, para um conjunto de atributos $B \subseteq M$, pode-se calcular um subconjunto de objetos B' contendo os objetos que possuem todos os atributos existentes em B, representado na equação 2.2:

$$B' := \{g \in G \mid gIm \text{ para todo } m \in B\} \tag{2.2}$$

Pode-se também agrupar as operações para calcular $(A')'$ como A", o que significa computar as duas operações em sequência usando o resultado da primeira como entrada para a segunda. Como resultado, A" representará o fechamento (*closure*) do conjunto A, e o mesmo pode ser feito com os subconjuntos de objetos, no qual B" representará o fechamento de B.

Dado um contexto formal $K := (G, M, I)$, pode-se calcular os seus conceitos formais. Um conceito formal de um contexto K é um par (A, B) - no qual se chama A de extensão (*extent*) e B de intensão (*intent*) do conceito - cujo resultado da equação 2.3 é verdadeiro.

$$A \subseteq G,\ B \subseteq M,\ A' = B \text{ e } B' = A \tag{2.3}$$

O conjunto de todos os conceitos formais de um contexto K é representado por $C(G, M, I)$. Assim sendo, um conceito é identificado pela sua extensão - que contém todos os objetos que pertencem ao conceito formal - e pela sua intensão - que contém todos os atributos compartilhados pelos objetos da extensão (Carpineto e Romano, 2004).

Para exemplificar, o conceito $(\{1,4\},\{b,c\})$ indica que os objetos $\{1,4\}$ possuem os mesmos atributos $\{b,c\}$, ou da mesma forma, que os atributos $\{b,c\}$ são comuns aos objetos $\{1,4\}$. Para simplificar a representação dos conceitos denotam-se os conjuntos $\{b,c\},\{1,4\}$ que formam os conceitos simplesmente como bc e 14 respectivamente. Assim, usando a representação simplificada denota-se o conceito $(\{1,4\},\{b,c\})$ apenas como $(14, bc)$.

Todos os conceitos formais para determinado contexto formal podem ser obtidos mediante o cálculo de todos os subconjuntos possíveis de G denotados por $\wp(G)$ e a seleção de todo o par (A, B) tal que $A \in \wp(G)$ e $A' = B$ e $B' = A$. Este método é equivalente ao algoritmo *Naive* comentado na próxima seção.

A Tabela 2.4 apresenta a lista de todos os conceitos formais obtidos a partir do contexto representado na Tabela 2.3.

Tabela 2.4: Conceitos formais encontrados para o contexto formal da Tabela 2.3.

Extensão $A \subseteq G$	Intensão $B \subseteq M$
1234	$\emptyset$
124	c
134	b
14	bc
23	a
24	cd
2	acd
3	ab
4	bcd
$\emptyset$	abcd

2.2.3 Reticulado de Conceitos

Se (A_1, B_1) e (A_2, B_2) são conceitos formais de um contexto, (A_1, B_1) é chamado de subconceito de (A_2, B_2), contanto que $A_1 \subseteq A_2$ (o que é analogamente equivalente a $B_2 \subseteq B_1$). Neste caso, (A_2, B_2) é um superconceito de (A_1, B_1) e denota-se $(A_1, B_1) \leqslant (A_2, B_2)$. A relação $\leqslant$ é chamada de ordem hierárquica (ou simplesmente ordem) dos conceitos.

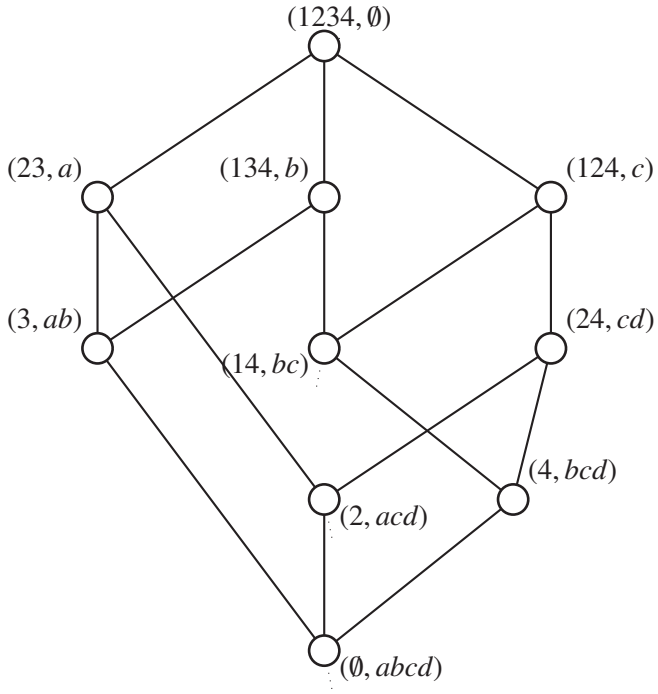


Figura 2.6: Reticulado de conceitos do contexto formal representado na Tabela 2.3.

O conjunto de todos os conceitos de um contexto $K := (G, M, I)$ dispostos sob uma relação de ordem hierárquica é denotado por $\underline{\mathfrak{B}}(G, M, I)$ e chamado de reticulado de conceitos (*concept lattice*) do contexto K (Ganter e Wille, 1999). A Figura 2.6 apresenta o reticulado de conceitos do contexto formal representado na Tabela 2.3.

2.2.4 Algoritmos para cálculo dos conceitos e do reticulado de conceitos

No que diz respeito aos algoritmos para cálculo dos conceitos de um contexto formal, existe na atualidade uma grande quantidade de propostas, merecendo destaque os algoritmos mais básicos tais como o *Naive* e *Intersections* demonstrados em Carpineto e Romano (2004). As abordagens propostas por Ganter (1984) como o *Next Closure* até a atualidade apresentam um bom desempenho.

Com o crescimento das bases de dados, novos algoritmos foram criados para lidar com este novo cenário, bem como para trabalhar com processamento concorrente. Krajca e Vychodil (2009); Krajca *et al.* (2010); Andrews (2009) apresentam abordagens recentes com melhor desempenho e paralelismo para o cálculo dos conceitos formais em grandes conjuntos de dados. Kuznetsov e Obiedkov (2002) apresentam um comparativo entre os

principais algoritmos usados para cálculo dos conceitos formais e construção do reticulado, e uma comparação mais recente entre as abordagens existentes para cálculo dos conceitos formais é apresentada por Strok e Neznanov (2010); no trabalho foram comparados e analisados os algoritmos variando o formato dos dados de entrada quanto a número de objetos, número de atributos e densidade do contexto.

2.2.5 Contextos multivalorados

Um contexto formal representa as informações por meio da indicação da presença ou ausência dos atributos nos objetos no que se pode chamar de contexto monovalorado. No entanto, no mundo real, as informações podem se apresentar das mais variadas formas, por exemplo, um atributo cor descrito para um objeto pássaro pode possuir uma lista de cores como possíveis valores.

Para representar estas informações é necessário organizá-las em um contexto multivalorado, conforme representado na Tabela 2.5, em que o atributo *tamanho* possui como domínio os valores {*pequeno*, *médio*, *grande*}, *distância* pode ser {*próximo*, *distante*} e *possui lua* pode assumir os valores {*sim*, *não*}.

Tabela 2.5: Exemplo de contexto multivalorado. Adaptado de Carpineto e Romano (2004).

	tamanho			distância do sol		possui lua	
	pequeno	médio	grande	próximo	distante	sim	não
Mercúrio	x			x			x
Vênus	x			x			x
Terra	x			x		x	
Marte	x			x		x	
Jupiter			x		x	x	
Saturno			x		x	x	
Urano		x			x	x	
Netuno		x			x	x	
Plutão	x				x	x	

De acordo com Carpineto e Romano (2004), para o cálculo dos conceitos e construção do reticulado de conceitos pode-se transformar um contexto multivalorado em um contexto monovalorado e interpretar os conceitos do contexto transformado como conceitos do contexto original multivalorado.

Uma forma simples de fazer a transformação é substituir cada atributo multivalorado pelos respectivos pares atributo-valor; desta forma, cada objeto é descrito por um par

atributo-valor para cada valor do atributo no contexto multivalorado. Usando esta transformação, o contexto da Tabela 2.5 passará a ter o conjunto de atributos definido pelos pares {*tamanho - pequeno* (tp), *tamanho - médio* (tm), *tamanho - grande* (tg), *distância do sol - próximo* (dp), *distância do sol - distante* (dd), *possui lua - sim* (ls), *possui lua - não* (ln)}. Os valores para os atributos permanecem os mesmos do contexto multivalorado original. Após a transformação é possível o cálculo dos conceitos e a construção do reticulado conforme demonstrado na Figura 2.7.

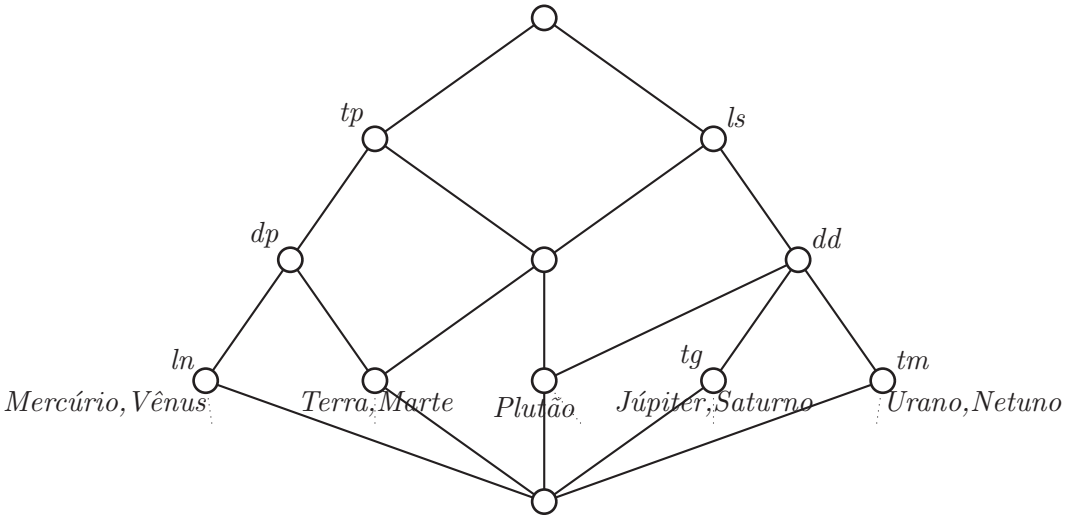


Figura 2.7: Reticulado para contexto multivalorado da Figura 2.7 após a sua transformação em monovalorado (Carpineto e Romano, 2004).

Esta tranformação de contexto formal multivalorado para contexto formal monovalorado proposta é denominada por Ganter e Wille (1999) como ampliação conceitual (*conceptual scaling*), e no método demonstrado é mais especificamente chamada de ampliação conceitual plana (*plain scaling*). Este método é útil com atributos multivalorados nominais, no entanto é bastante insatisfatório quando se tratar de atributos numéricos.

Para Carpineto e Romano (2004), o método além de ser insatisfatório para atributos multivalorados numéricos, não preserva, caso exista, a relação de ordem entre os valores dos atributos. Para exemplificar, o autor sugere a modificação do atributo *distância do sol* do contexto multivalorado original da Tabela 2.5 para representar a distância não mais apenas como *próximo* ou *distante*, mas sim com um conjunto de possíveis valores que representarão a distância em milhões de quilômetros contendo uma relação ordinal entre eles, representados pelo conjunto { $\geq$ *dezenas*, $\geq$ *algumas centenas*, $\geq$ *várias centenas*, $\geq$

milhares}. Os novos valores para o atributo *distância do sol* em cada objeto serão:

- Mercúrio: ≥ dezenas;
- Vênus: ≥ algumas centenas;
- Terra: ≥ algumas centenas;
- Marte: ≥ algumas centenas;
- Júpiter: ≥ várias centenas;
- Saturno: ≥ milhares;
- Urano: ≥ milhares;
- Netuno: ≥ milhares;
- Plutão: ≥ milhares.

Para ampliação de um atributo com estes possíveis valores que possuem uma relação ordinal entre eles, ou mesmo os atributos puramente numéricos mediante um processo de discretização, é conveniente aplicar-se uma segunda abordagem conhecida como ampliação ordinal (*ordinal scaling*). Nesta abordagem, assume-se que determinados valores englobam outros valores pelo fato de que o próximo valor é maior (ou menor) que o anterior. Isto pode ser feito por meio da criação de novos atributos com as propriedades requeridas do valor original.

O subcontexto monovalorado derivado que conceitualmente representa a relação "maior que" é demonstrado na Tabela 2.6.

O reticulado completo do contexto ampliado pelo processo de ampliação ordinal contendo o atributo *distância do sol* mostrado na Tabela 2.6 e os demais atributos inicialmente representados no contexto formal da Tabela 2.5, são demonstrados na Figura 2.8. No exemplo, o relacionamento "maior que" entre os valores possíveis para o atributo *distância do sol* foram mantidos no reticulado final no qual os conceitos formais descritos por estes valores do atributo foram ordenados corretamente. Assim, é possível encontrar no reticulado os planetas cuja distância do sol é maior ou igual a algumas centenas de quilômetros (≥ *algumas centenas*) como um subconceito dos planetas para os quais a distância do sol é algumas dezenas de milhões de quilômetros *(*≥ *dezenas*).

Tabela 2.6: O subcontexto derivado para o atributo *"distância do sol"*. Adaptado de Carpineto e Romano (2004).

	distância do sol (em milhões de quilômetros)			
	≥ dezenas	≥ algumas centenas	≥ várias centenas	≥ milhares
Mercúrio	x			
Vênus	x	x		
Terra	x	x		
Marte	x	x		
Júpiter	x	x	x	
Saturno	x	x	x	x
Urano	x	x	x	x
Netuno	x	x	x	x
Plutão	x	x	x	x

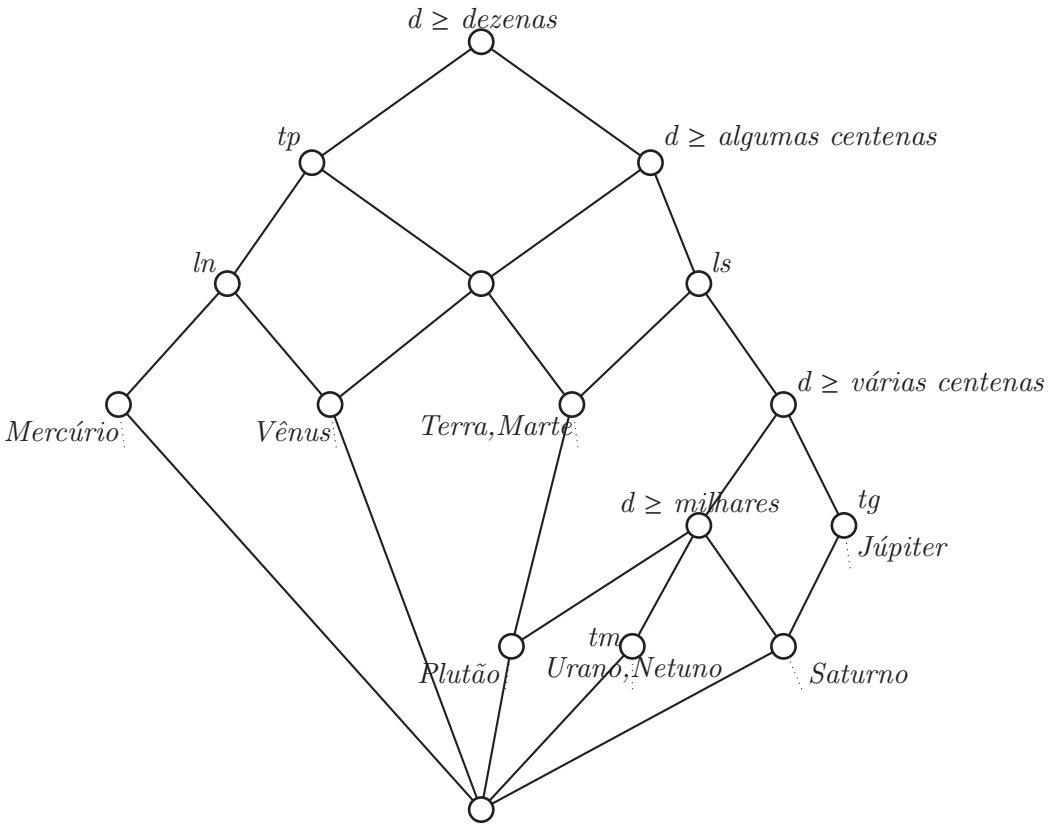


Figura 2.8: Reticulado para contexto multivalorado da Figura 2.7 com o atributo *distância do sol* ampliado de acordo com a Tabela 2.6. Adaptado de Carpineto e Romano (2004).

2.2.6 Adicionando conhecimento a um contexto

Em muitas aplicações para facilitar a análise e melhorar os resultados na exploração dos dados usando reticulados de conceitos, pode-se acrescentar conhecimento externo a eles, como por exemplo, informações referentes a uma taxonomia.

Para melhor demonstrar como esta adição de conhecimento pode ser feita, analisa-se o exemplo proposto por Carpineto e Romano (2004). A Tabela 2.7 apresenta um contexto representando uma relação de documentos (d_1,d_2,...,d_6), representados como objetos do contexto, e um conjunto de termos que representam os assuntos presentes nos documentos, por exemplo, o documento d_1 está relacionado com IA, SE e RI.

Tabela 2.7: Contexto representando uma pequena coleção de documentos e os termos que os descrevem de acordo com Carpineto e Romano (2004). Atributos: Inteligência Artificial (IA); Sistemas Especialistas (SE); Recuperação da Informação (RI); Catalogação (Cat); Indexação (Ind); Ciência da Informação (CI); Sistemas para Recuperação da Informação (SRI); Sistemas Baseados em Conhecimento (SBC).

	IA	SE	RI	Cat	Ind	CI	SRI	SBC
d_1	1	1	1	0	0	0	0	0
d_2	1	1	0	1	0	0	0	0
d_3	1	1	0	0	1	0	0	0
d_4	1	1	0	0	0	1	0	0
d_5	1	1	0	1	0	0	1	0
d_6	1	0	1	0	0	0	0	1

Pode-se construir o reticulado de conceitos a partir dos dados deste contexto, obtendo o resultado, conforme mostra a Figura 2.9, e explorar as correlações entre os documentos e os termos. Para exemplificar, analisando o reticulado pode-se perceber que o documento d_1 está relacionado aos termos RI, SE e IA uma vez que estes são seus nós ascendentes no reticulado.

Suponha-se agora que exista uma organização pré-definida entre os termos que se associam aos documentos, ou seja, uma taxonomia que representa uma organização hierárquica dos termos associados aos objetos na Tabela 2.7. Esta hierarquia pode ser representada poe meio de uma DAG ou uma árvore representando as correlações entre os próprios termos. A Figura 2.10 mostra um exemplo de taxonomia para os termos associados aos documentos na Tabela 2.7. De acordo com esta taxonomia, um documento que possui o termo RI, também possui os termos CI e AC. Igualmente é importante observar que alguns termos utilizados no contexto inicial não estão presentes na taxonomia, como o

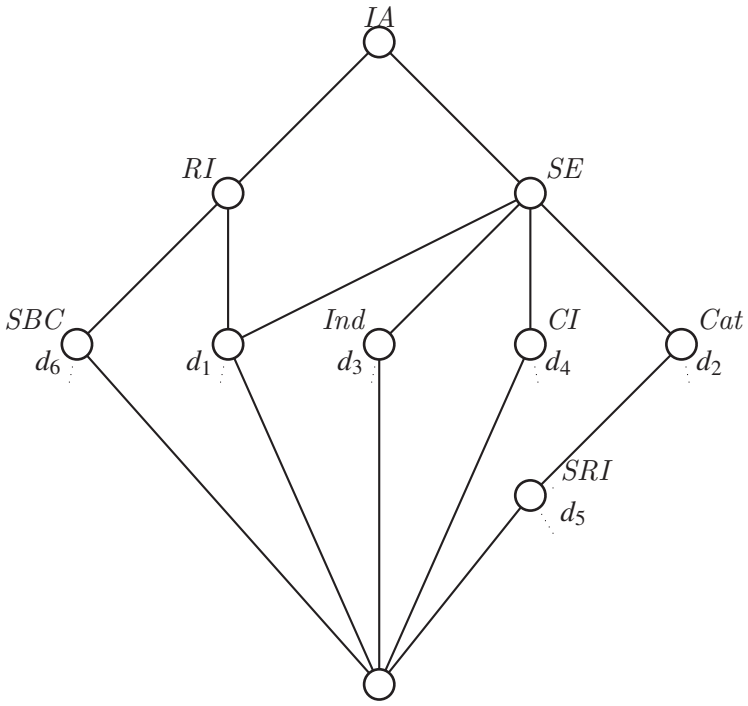


Figura 2.9: Reticulado de conceitos para o contexto de documentos da Tabela 2.7 (Carpineto e Romano, 2004).

caso do termo IA que será mantido sem alterações no contexto aumentado.

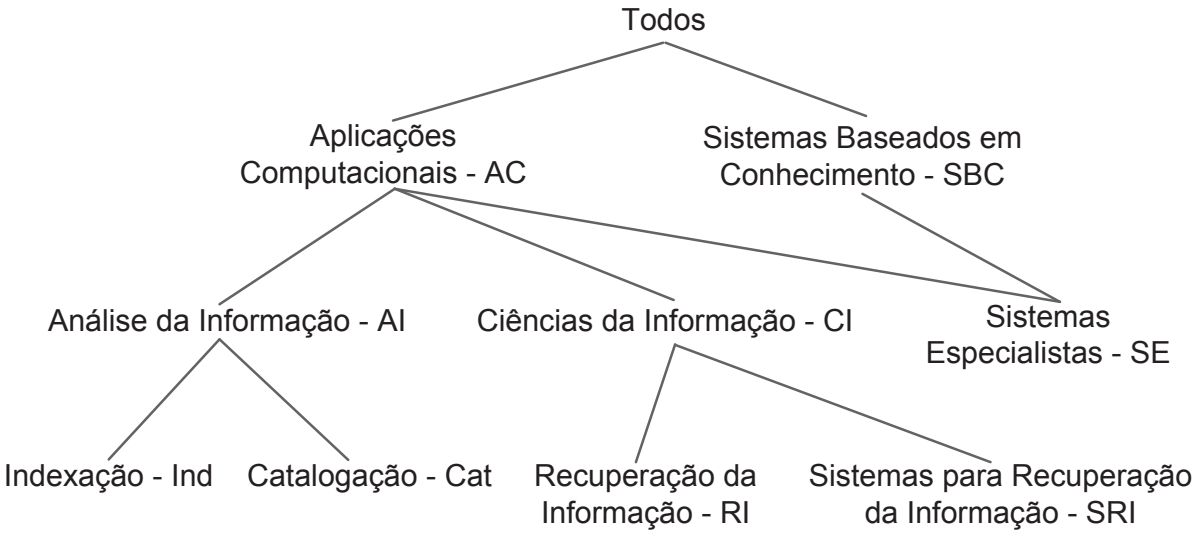


Figura 2.10: DAG representando a taxonomia dos atributos *(background information)* (Carpineto e Romano, 2004).

Esta taxonomia pode ser representada por um contexto que terá os termos como objetos e também como atributos assinalando as correlações existentes entre os próprios termos. Este contexto é representado na Tabela 2.8, sendo que como um reticulado é um conjunto parcialmente ordenado, é necessário o preenchimento da diagonal principal para indicar que um termo relaciona-se com ele próprio (reflexividade) e também calcular os fechos transitivos da matriz a fim de representar as relações transitivas entre os termos.

Tabela 2.8: Contexto que representa a taxonomia dos termos associados aos documentos na Tabela 2.7 (Carpineto e Romano, 2004).

	IA	SE	RI	Cat	Ind	CI	SRI	SBC	AC	AI
IA	1	0	0	0	0	0	0	0	0	0
SE	0	1	0	0	0	0	0	1	1	0
RI	0	0	1	0	0	1	0	0	1	0
Cat	0	0	0	1	0	0	0	0	1	1
Ind	0	0	0	0	1	0	0	0	1	1
SI	0	0	0	0	0	1	0	0	1	0
SRI	0	0	0	0	0	1	1	0	1	0
SBC	0	0	0	0	0	0	0	1	0	0
AC	0	0	0	0	0	0	0	0	1	0
AI	0	0	0	0	0	0	0	0	1	1

Levando em conta o contexto da taxonomia representado na Tabela 2.8, pode-se adicionar a informação referente a esta taxonomia hierárquica ao contexto original da Tabela 2.7. Dessa forma, obter-se-á um contexto que representa as correlações entre documentos e os termos, respeitando a taxonomia hierárquica dos termos. Após este processo obtém-se o contexto representado na Tabela 2.9.

Após a construção do contexto aumentado, pode-se então calcular o reticulado com as informações nele representadas e obter-se uma representação das correlações entre documentos e seus termos, obedecendo a uma taxonomia pré-estabelecida que representa a hierarquia entre os termos. O reticulado final é representado na Figura 2.11, no qual se pode observar que o documento d_1, que no reticulado da Figura 2.9 está associado com os termos RI, SE e IA, após a imposição da hierarquia dos termos, também está associado aos termos CI, SBC e AC além dos anteriores.

Nesta subseção foi demonstrada a ideia central da adição de conhecimento a um contexto formal. Existem diversas propostas envolvendo exploração de dados por meio da adição de conhecimento a contextos, e mais informações sobre o assunto podem ser obtidas em Ganter (1999); Belohlavek e Vychodil (2009); Belohlávek e Vychodil (2010).

Tabela 2.9: Contexto que representa uma pequena coleção de documentos e os termos que os descrevem após a adição do conhecimento de fundo (Carpineto e Romano, 2004). Os atributos adicionados pela imposição da hierarquia dos termos ao contexto da Tabela 2.7 estão em destaque.

	IA	SE	RI	Cat	Ind	CI	SRI	SBC	AC	AI
d_1	1	1	1	0	0	1	0	1	1	0
d_2	1	1	0	1	0	0	0	1	1	1
d_3	1	1	0	0	1	0	0	1	1	1
d_4	1	1	0	0	0	1	0	1	1	0
d_5	1	1	0	1	0	1	1	1	1	1
d_6	1	0	1	0	0	1	0	1	1	0

2.2.7 FCA e Mineração de Dados

Existem muitas propostas e trabalhos que empregam a FCA no contexto de mineração de dados para problemas de classificação plana. Nesta Seção, será feita uma breve explanação sobre as principais propostas existentes com o intuito de elencá-las e apresentar as suas características de forma geral. No contexto de classificação pode-se destacar os algoritmos Grand, Rulearner, Legal, Galois, Similares1 e Similares2 detalhados a seguir.

O algoritmo Grand (nome baseado em "*graph-based induction*"), proposto por Oosthuizen (1988), é a proposta mais antiga e seu funcionamento é proposto por meio de uma primeira fase de exploração que produz regras de classificação com base em um pseudorreticulado construído a partir dos dados de entrada. Em uma segunda fase as regras produzidas são aplicadas a um contexto de entrada sem as informações referentes às classes dos objetos. A predição das classes é feita mediante a seleção de um conjunto de regras que cubram estes objetos.

Sahami (1995) propôs o algoritmo Rulearner que é similar ao Grand com duas diferenças básicas: ignora as classes no processo de construção do pseudorreticulado e requer um parâmetro de impureza como entrada. Sua primeira fase baseia-se na mesma estratégia de produzir regras que possuam poucos atributos e cubram muitos objetos, preferindo, portanto, nós mais ao topo do pseudorreticulado. Os nós do pseudorreticulado são rotulados com valores de classe ou como impuros. Nós não-impuros são aqueles que possuem, dentre seus descendentes, uma quantidade considerável de nós-objeto com mesmo valor de classe. Já para os nós impuros, seus nós-objetos descendentes não definem um valor predominante de classe. O parâmetro de impureza recebido como entrada controla essa predominância determinando a porcentagem tolerada de outros valores de classe dentre os

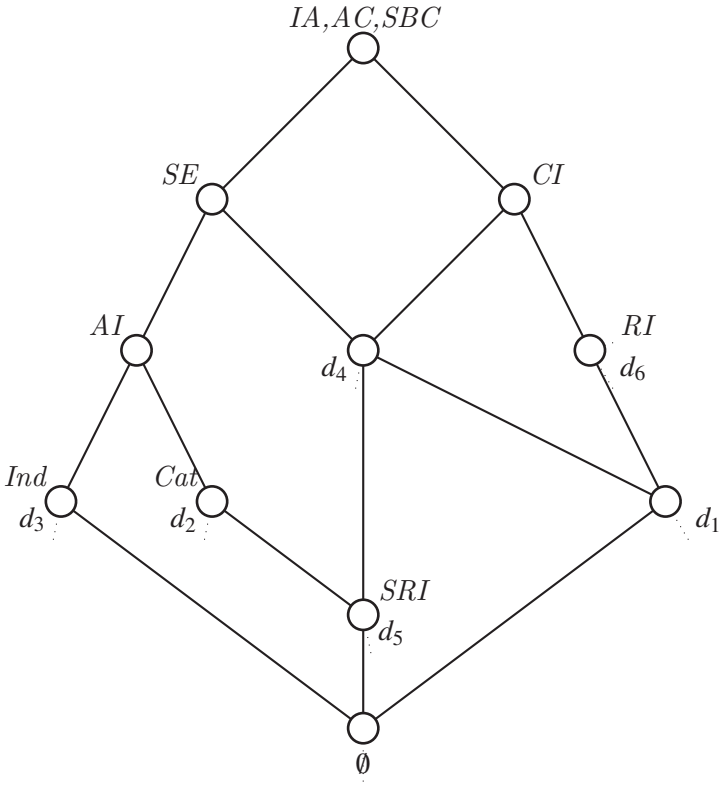


Figura 2.11: Reticulado contendo as correlações entre documentos e termos de acordo com a hierarquia de termos pré-estabelecida (Carpineto e Romano, 2004).

nós-objeto descendentes. Observa-se então que, embora ignore a classe e seus valores para construir o pseudorreticulado, o algoritmo precisa deles para rotular os nós, acessando, para isso, o contexto formal. A sua segunda fase é idêntica ao algoritmo Grand.

O algoritmo Legal (nome baseado em "learning with Galois lattice"), proposto nos trabalhos Liquière e Nguifo (1990); Nguifo (1994), apresenta como características marcantes o uso do reticulado ao invés de um pseudorreticulado e a limitação de suportar apenas classificação binária. A grande vantagem desta proposta é não precisar calcular o reticulado inteiro o que lhe dá uma vantagem considerável no quesito tempo de processamento em relação às outras propostas, sendo que parâmetros indicando o tamanho mínimo que um conceito deverá possuir na extensão são usados para podar a construção do reticulado com base no suporte de cada conceito. Após a obtenção do reticulado, regras são extraídas e posteriormente usadas para classificar objetos sem o valor de classe.

O algoritmo Galois apresentado em Carpineto e Romano (1993, 1996, 2004) usa o reticulado completo de conceitos e não mais trabalha com regras. Para classificar objetos

sem valores de classe, é realizado o caminhamento pelo reticulado, usando o chamado parâmetro de impureza que determina quantos valores de classes são tolerados na extensão dos conceitos formais. Para cada objeto a ser classificado, o algoritmo considera conceitos formais que tenham certas propriedades, e baseando-se nos valores de classe sugeridos por tais conceitos classifica aquele objeto.

No trabalho de Silva *et al.* (2006), os autores demonstram o uso do reticulado para tarefas de classificação e agrupamento. Posteriormente, no trabalho de Silva (2007), é apresentada uma ampla revisão e experimentos envolvendo os algoritmos baseados em FCA para classificação. No estudo também são propostas duas novas abordagens denominadas Similares1 e Similares2. De maneira geral, estas duas abordagens se diferem das anteriores pelo fato de utilizarem a ideia de similaridade para escolher os conceitos que serão utilizados no processo de determinar a classe para novas novas instâncias, enquanto as outras propostas se baseavam no processo de encontrar conceitos que cobrissem os objetos.

Um algoritmo para extrair regras de associação na presença de uma taxonomia usando FCA foi proposto por Cellier *et al.* (2007). Na algoritmo proposto as regras são encontradas por meio de um mecanismo de filtro para determinar se um exemplo é positivo ou negativo.

No *survey* proposto por Poelmans *et al.* (2010), é apresentado um estudo envolvendo 702 publicações realizadas no período entre os anos de 2003 e 2009 sobre o uso de FCA em KDD. O autor fez uso das próprias técnicas de FCA para mapear as áreas de estudo presentes em cada trabalho em que se pode perceber uma grande quantidade de publicações no período que empregaram técnicas de FCA para tarefas de classificação e descoberta de regras de classificação e associação.

Esta tese tem como foco a construção de um sistema classificador com suporte a taxonomia hierárquica de classes para sua aplicação em problemas de classificação hierárquica. Neste contexto, a predição da função de proteínas é um problema bastante explorado entre os pesquisadores por se tratar de bases de dados que apresentam as classes organizadas em uma taxonomia hierárquica, representada no formato de árvore ou DAG. A próxima Seção aborda os conceitos fundamentais sobre proteínas, ontologias para classificação de proteínas, e o problema da predição das funções das proteínas.

2.3 Proteínas

Proteínas são grandes moléculas que consistem em longas sequências ou cadeias de aminoácidos, também chamadas de cadeias polipeptídicas, as quais se dobram em um número de diferentes estruturas e realizam quase todas as funções de uma célula em um

organismo vivo (Freitas e Carvalho, 2007).

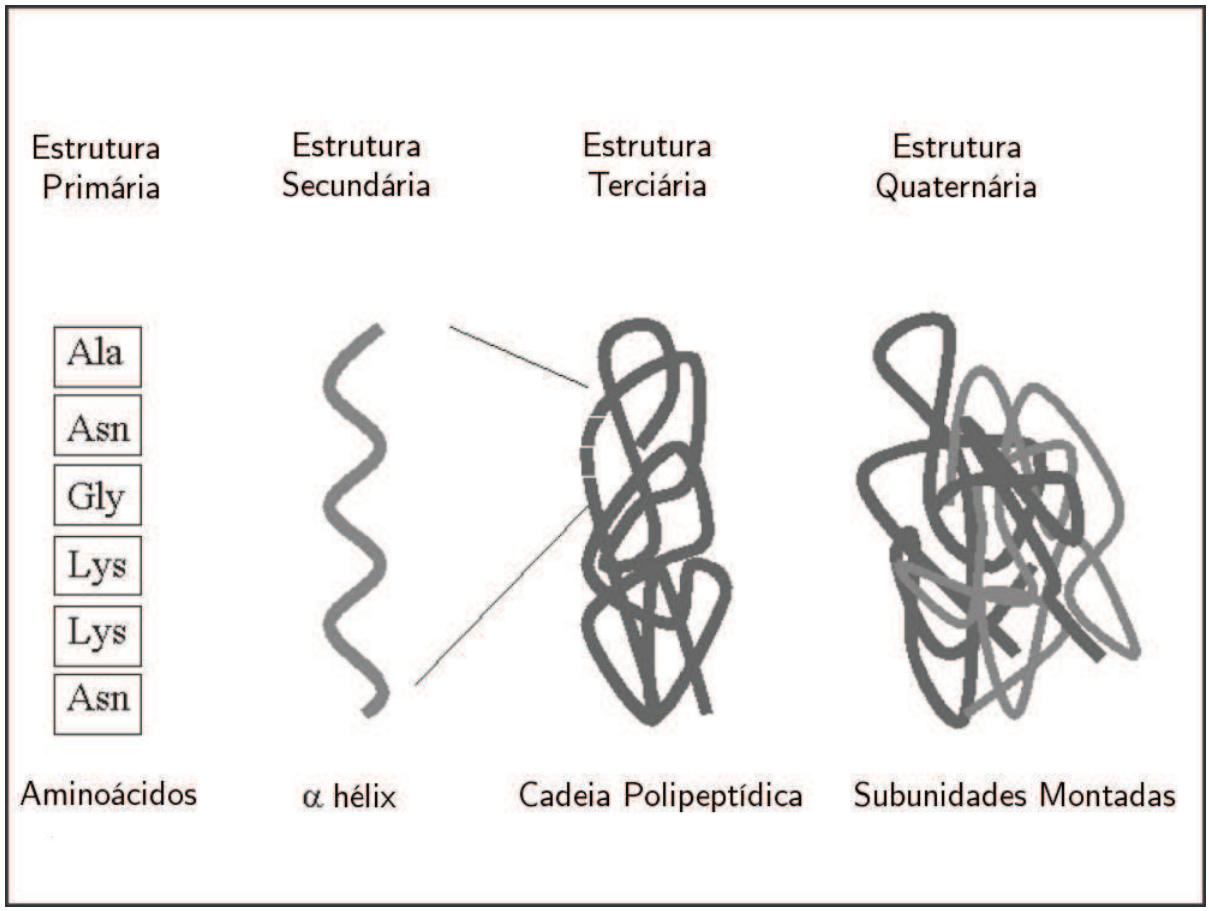


Figura 2.12: Quatro tipos de representação das estruturas das proteínas (Sugumaran, 2007).

As proteínas podem ser representadas por quatro tipos de estruturas. Conforme a Figura 2.12, as estruturas são:

- **Estrutura Primária:** A estrutura primária indica a ordem na qual os aminoácidos estão ligados, o peptídeo Leu-Gly-Thr-Val-Arg-Asp-His possui a estrutura primária diferente do peptídio Val-His-Asp-Leu-Gly-Arg-Thr, mesmo que ambos possuam o mesmo número e os mesmos tipos de aminoácidos. A estrutura primária é a primeira etapa unidimensional na especificação da estrutura tridimensional da proteína, que, por sua vez, determina as suas funções;
- **Estrutura Secundária:** A estrutura secundária apresenta o arranjo dos átomos do esqueleto da cadeia polipeptídica no espaço. Os arranjos em α-hélice e folhas β pregueadas, mantidos por pontes de hidrogênio são dois tipos diferentes de estrutura

secundária. Em proteínas muito grandes, o dobramento de partes da cadeia pode ocorrer independentemente do dobramento de outras partes. As porções de proteínas dobradas de modo independente são chamadas de domínios;

- **Estrutura Terciária:** A estrutura terciária representa o arranjo tridimensional de todos os átomos da proteína. As conformações das cadeias laterais e as posições de quaisquer grupos prostéticos são partes da estrutura terciária, assim como o arranjo de seções helicoidais e em folha pregueada uma em relação a outra;
- **Estrutura Quaternária:** A estrutura quaternária é uma propriedade das proteínas constituídas por mais de uma cadeia polipeptídica. O número de cadeias pode variar de duas até mais de 12, as quais podem ser idênticas ou diferentes.

As proteínas são conhecidas por desempenharem funções importantes na célula e necessárias em todos os processos químicos que ocorrem nos organismos vivos. Suas principais funções são:

- **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 proteicos;
- **Sustentação mecânica:** a alta força de tensão da pele e do osso é devido à 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.

Existem atualmente diversas bases de dados disponíveis on-line contendo informações sobre proteínas. A Tabela 2.10 apresenta uma lista das principais bases de dados sobre proteínas disponíveis on-line, conforme Otero (2010) elenca em seu trabalho.

O entendimento da estrutura de uma proteína é essencial para o entendimento de sua função. A função de uma proteína pode ser descrita em vários níveis de detalhes, do fisiológico, em que uma proteína X está envolvida no processo de replicação de células, até o químico, no qual uma proteína X catalisa a hidrólise de um determinado substrato

Tabela 2.10: Exemplo de bases de dados biológicas disponíveis publicamente contendo informações sobre proteínas.

UniProt (http://www.ebi.ac.uk/uniprot/)
IntAct (http://www.ebi.ac.uk/intact/)
CATH (http://www.cathdb.info/)
PROSITE (http://www.expasy.org/prosite/)
PubMed (http://www.ncbi.nlm.nih.gov/pubmed/)
Pfam (http://pfam.sanger.ac.uk/)
InterPro (http://www.ebi.ac.uk/interpro/)
DIP (http://dip.doe-mbi.ucla.edu/)
MEDLINE (http://medline.cos.com/)
TIGRFAMs (http://www.tigr.org/TIGRFAMs/)
PRINTS (http://www.bioinf.manchester.ac.uk/dbbrowser/PRINTS/)
ArrayExpress (http://www.ebi.ac.uk/arrayexpress/)
Protein Data Bank http://www.rcsb.org/pdb/

(Mello e Rigden, 2002). Para Campbell *et al.* (2001), as proteínas podem variar na sua composição, sequência e número de aminoácidos, por isso é possível uma enorme variação de sua estrutura e de sua função.

2.3.1 Esquemas para Classificação Funcional de Proteínas

As diversas bases de dados contendo informações sobre proteínas (relacionadas na Tabela 2.10) foram criadas para finalidades específicas e organizam as informações de acordo com este propósito. Para facilitar a integração dos dados entre as diferentes bases de dados foram desenvolvidos esquemas para anotações em proteínas baseados em um vocabulário controlado (ontologia). Nas próximas duas subseções são apresentados dois vocabulários propostos para padronização dos esquemas de anotações em proteínas: Ontologia Gênica e Catálogo Funcional.

Ontologia Gênica

A ontologia gênica comumente representada pela sigla GO (Ashburner *et al.*, 2000) provê um conjunto de termos que representam as propriedades do gene. Esse vocabulário de termos é consistente e estruturado para descrever domínios chave da biologia molecular. De acordo com Sullivan (2012), o projeto tem como objetivo padronizar a representação dos genes, dos atributos de produtos de genes entre as espécies e os bancos de dados, fornecendo um vocabulário controlado de termos que descrevem as características genéticas

de produtos e anotações genéticas de produtos, bem como um conjunto de ferramentas de suporte.

A estrutura dos termos é modelada como um grafo acíclico direcionado e hierárquico representado neste trabalho pela sigla DAG, com um único elemento maximal (raiz), em que os termos são os nós e os relacionamentos entre termos são os arcos do grafo. Um nó (termo) pode ter um ou mais pais, representando uma descrição mais genérica do termo. Um nó também pode ter filhos que representam uma definição mais específica do termo.

A GO descreve atributos de produtos gênicos em três domínios disjuntos da biologia molecular:

- **Função molecular:** descreve atividades realizadas por produtos gênicos no nível celular, tais como, atividades catalíticas ou de ligação;
- **Processo biológico:** descreve 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.

A Figura 2.13 mostra como a hierarquia das classes da GO está subdividida nos três domínios citados. Esta subdivisão está organizada no primeiro nível da hierarquia dividindo a taxonomia principal em três partições, uma para cada domínio, sendo os códigos GO0005575, GO0003674 e GO0008150 os respectivos identificadores da partição referente a cada um dos domínios.

Na Figura 2.14, é demonstrado um subconjunto da hierarquia das classes da GO referente às atividades *ion channel*. Aqui é importante observar que a DAG referente a estas atividades é um subconjunto da partição "Função Molecular" da GO.

Catálogo Funcional

O catálogo funcional, representado neste trabalho pela sigla FunCat, é um esquema para classificação que define uma taxonomia hierárquica organizada em árvore (Ruepp e et al, 2004). Este esquema pode ser utilizado para classificação de uma grande quantidade de proteínas em diferentes organismos. A hierarquia é composta por 27 nós no primeiro nível e pode ter até 6 níveis de especialização. No FunCat, as categorias são separadas por pontos, por exemplo, "20.03.01.01" representa mais especificamente o subconjunto *"ion channels"*, sendo que "20" representa as informações referentes a *"cellular transport"*, *"transport facilities and transport routes"*, "20.03" equivale a *"transport facilities"* e "20.03.01" *"channel/pore class transport"*.

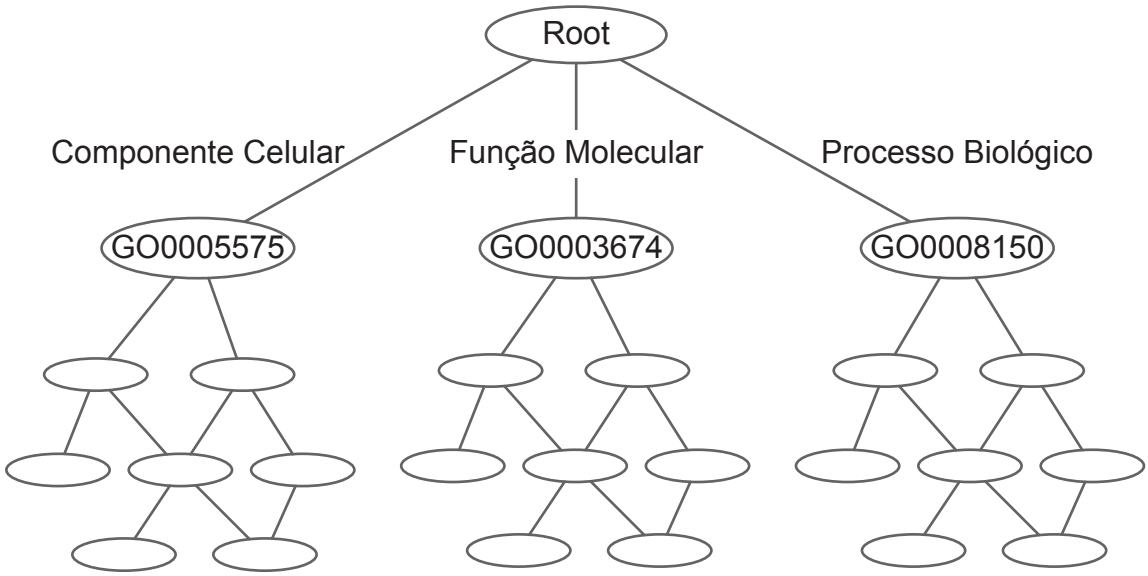


Figura 2.13: Exemplo da subdivisão da GO.

Para a maioria das proteínas, a função celular não pode ser completamente descrita por uma única categoria funcional, e sim, deve ser considerada a soma de propriedades diferentes. O FunCat permite a atribuição de várias categorias a uma única proteína.

Quanto ao tamanho, o FunCat versão 2.1, de 9 de janeiro de 2007, apresenta 1362 categorias, a GO, versão de 22 de novembro de 2009, inclui um total de 28625 termos (não obsoletos).

2.3.2 Predição da Função de Proteínas

O crescimento do número de experimentos envolvendo proteínas levou à descoberta de uma grande quantidade de proteínas que precisam ser classificadas. Este fato fez com que aumentasse o interesse dos pesquisadores por métodos automatizados para a predição das funções destas proteínas.

Na prática, o processo de mapeamento da função das proteínas deveria ocorrer de forma simples em que dada uma sequência, encontra-se sua estrutura e tem-se a sua função. Entretanto, de acordo com Lesk (2008), embora se possa estar seguro de que sequências similares de aminoácidos resultarão em estruturas de proteínas similares, a relação entre estrutura e função é mais complexa. Proteínas de estruturas semelhantes, e até mesmo de sequências semelhantes, podem realizar diferentes funções. Mais do que isso, assim como muitas sequências diferentes são compatíveis com uma mesma estrutura, dobramentos de

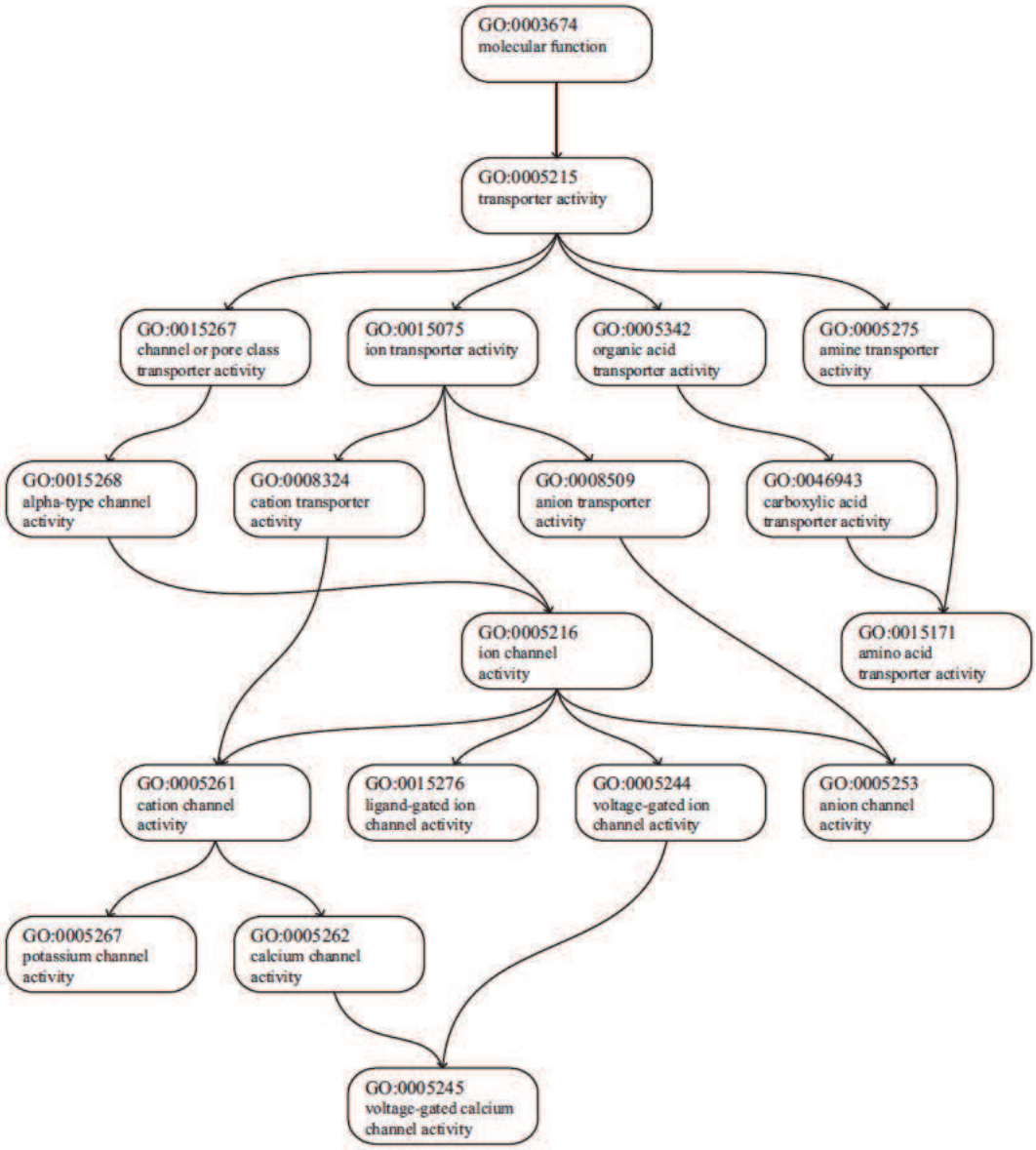


Figura 2.14: Subconjunto da GO referente ao *ion channel* de acordo com Otero (2010). As atividades do *"ion channel"* são parte da subdivisão "Função Molecular" da GO.

proteínas diferentes, que é um processo químico por meio do qual a estrutura de uma proteína assume sua configuração funcional, podem ter a mesma função.

As principais ferramentas existentes para predição da função de proteínas utilizam o princípio da homologia, tendo como justificativa biológica para a transferência baseada em homologia a afirmação de que se duas sequências têm um elevado grau de semelhança, então elas evoluíram de um ancestral comum, e desta forma, têm funções similares ou até mesmo idênticas. A FASTA (Pearson e Lipman, 1988) e a PSI-BLAST (Altschul *et al.*,

1997) são exemplos de ferramentas baseadas em homologia.

No entanto, conforme anteriormente mencionado, nem sempre esta comparação por homologia é possível no caso das proteínas. Friedberg (2006) mostra alguns exemplos em que, mesmo com uma alta similaridade na sequência, a transferência de anotação pode ser errada e que esta observação estatística varia muito com relação ao tipo de função. Outro problema encontrado pelo autor é o fato de que assim como os bancos de dados de sequência estão crescendo rapidamente, a diversidade das sequências também está crescendo e as ferramentas que utilizam estes dados para predição não são sensíveis o suficiente para descobrir outras semelhanças entre estas proteínas. Com isso, existe o fato de que anotações errôneas estão sendo propagadas pelos bancos de dados na mesma velocidade com que novas sequências vêm sendo analisadas. Baseando-se nesta situação, outras formas de predição da função de proteínas têm sido desenvolvidas.

Em King *et al.* (2004), é demonstrado um método alternativo à transferência baseada em homologia chamado de predição por mineração de dados. Desta forma, vários métodos de aprendizado de máquina têm sido utilizados e estes métodos se baseiam na combinação de informações de diversos tipos de elementos da proteína, como por exemplo:

- Atributos de Aminoácidos;
- Estrutura;
- Fusão de Genes;
- Proximidade dos cromossomos;
- Padrões Filogenéticos.

O uso da predição da função de proteínas por mineração de dados vem crescendo nos últimos anos e os resultados iniciais têm sido bastante expressivos. Contribuíram muito para este crescimento a criação dos vocabulários padronizados como a GO e o FunCat.

2.4 Considerações Finais

Neste capítulo, foram abordadas as teorias e técnicas computacionais e da área biológica que serão empregadas para construção do HMCS-FCA. A revisão abordou os conceitos teóricos fundamentais sobre classificação de dados, FCA e proteínas.

Capítulo

3

Estado da Arte em Classificação Hierárquica

3.1 Considerações Iniciais

Neste capítulo, serão apresentados os conceitos e trabalhos que representam o estado da arte em relação aos problemas de classificação hierárquica. Para efeitos deste trabalho, o foco principal serão os trabalhos que utilizam a abordagem de classificação global e outros a eles correlacionados. A Seção 3.2 discorre a respeito dos temas: classificação hierárquica de dados e avaliação do desempenho de sistemas de classificação. A Seção 3.3 elenca e classifica os principais trabalhos em classificação hierárquica com base em um amplo estudo realizado por Silla e Freitas (2011). Na Seção 3.4, são apresentadas as características dos principais trabalhos na atualidade sobre classificadores hierárquicos globais.

3.2 Classificação Hierárquica de Dados

Para Wu *et al.* (2005), a classificação de dados torna-se muito mais complexa quando uma instância precisa ser associada a uma classe presente em uma taxonomia de classes hierarquicamente organizada, e define uma taxonomia de classes como uma árvore estruturada contendo a hierarquia das classes definida com base em um conjunto parcialmente ordenado $(C, \prec)$, em que C é um conjunto finito contendo todas as classes de um domínio,

e a relação $\prec$ representa um relacionamento do tipo "É-UM". Silla e Freitas (2011) definem que o relacionamento "É-UM" é assimétrico, antirreflexivo e transitivo em que:

- Um único e maior elemento R é a raiz da árvore;
- $\forall c_i, c_j \in C$, se $c_i \prec c_j$ então $c_j \not\prec c_i$ (assimetria);
- $\forall c_i \in C$, $c_i \not\prec c_i$ (antirreflexividade);
- $\forall c_i, c_j, c_k \in C$, $c_i \prec c_j$ e $c_j \prec c_k$ implica em $c_i \prec c_k$ (transitividade).

Estas definições foram propostas inicialmente para uma taxonomia estruturada em forma de árvore, mas podem ser utilizadas para definir taxonomias estruturadas em forma de DAG.

Uma das grandes diferenças entre as classificações plana binária e multirrótulo para com a classificação hierárquica reside no fato de que devido ao relacionamento "É-UM" definido por meio da taxonomia hierárquica das classes, quando uma instância é classificada como pertencente a uma classe disposta na taxonomia, ela automaticamente também pertence a todas as classes ancestrais ao nó em que foi classificada. Por exemplo, observando a Figura 3.1 (em que R é a raiz da árvore), se a saída de um classificador apontar que uma determinada instância pertence a classe 2.1.1, ela também pertence naturalmente às classes 2.1 e 2.

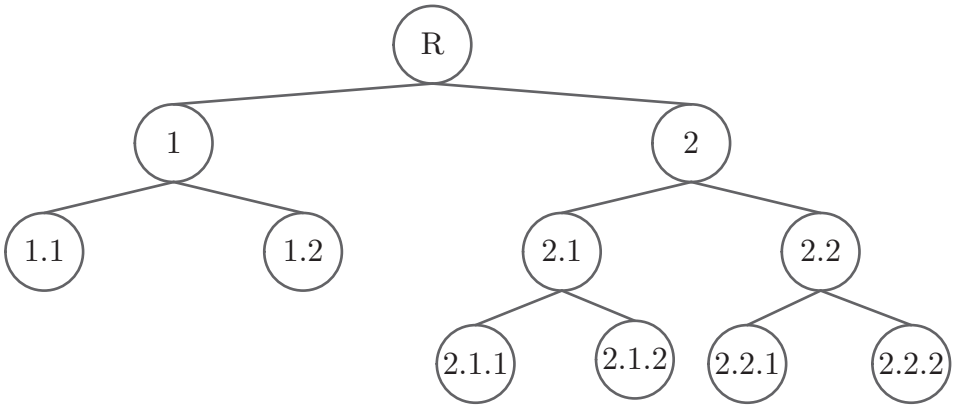


Figura 3.1: Exemplo de taxonomia hierárquica de classes baseada em árvore de acordo com Silla e Freitas (2011).

Para Freitas e Carvalho (2007), as técnicas utilizadas para classificação hierárquica se diferenciam por muitos critérios, sendo três deles os que merecem destaque:

- O tipo de estrutura utilizada para representar a taxonomia de classes, em que duas estruturas são tipicamente utilizadas: árvore ou DAG;
- A profundidade na hierarquia de classes em que as instâncias serão classificadas, somente nos nós folha ou também em nós intermediários;
- A maneira pela qual a estrutura hierárquica é explorada, local ou global.

As próximas subseções (3.2.1, 3.2.2, 3.2.3, 3.2.4, 3.2.4 e 3.2.5) abordarão estes assuntos de acordo com a uniformização de nomenclatura e termos propostos por Silla e Freitas (2011).

3.2.1 Estrutura utilizada para representar a taxonomia

Os métodos para classificação hierárquica podem ser diferenciados por vários critérios, o primeiro a ser considerado é a estrutura hierárquica contendo a taxonomia de classes que será utilizada. Esta estrutura é normalmente uma árvore ou uma DAG (Freitas e Carvalho, 2007).

A Figura 3.2 mostra duas estruturas básicas de taxonomia hierárquica de classes organizada em árvore e em DAG respectivamente. A principal diferença entre as representações está no fato de que utilizando DAG um nó pode possuir mais do que um nó pai, o que aumenta a complexidade do processo de classificação. Na representação, o nó X localizado na parte inferior da figura possui esta característica ao estar diretamente conectado a duas classes ascendentes.

3.2.2 Profundidade em que a classificação ocorre

Um método para classificação pode ser implementado de maneira a sempre classificar as instâncias como sendo pertencentes a classes presentes em nós folha da taxonomia hierárquica de classes proposta, ou pode permitir que uma instância seja classificada como sendo de uma classe pertencente a qualquer nó em qualquer nível da estrutura.

Para efeito de padronização da terminologia, Silla e Freitas (2011) propõem denominar as duas formas citadas respectivamente como *Mandatory Leaf-Node Prediction* (MLNP) e *Non-Mandatory Leaf-Node Prediction* (NMLNP). A Figura 3.3 demonstra as classes da hierarquia que poderão ser atribuídas a uma instância se a classificação for MLNP e a Figura 3.4 demonstra as classes que podem ser associadas a uma instância quando a classificação é NMLNP.

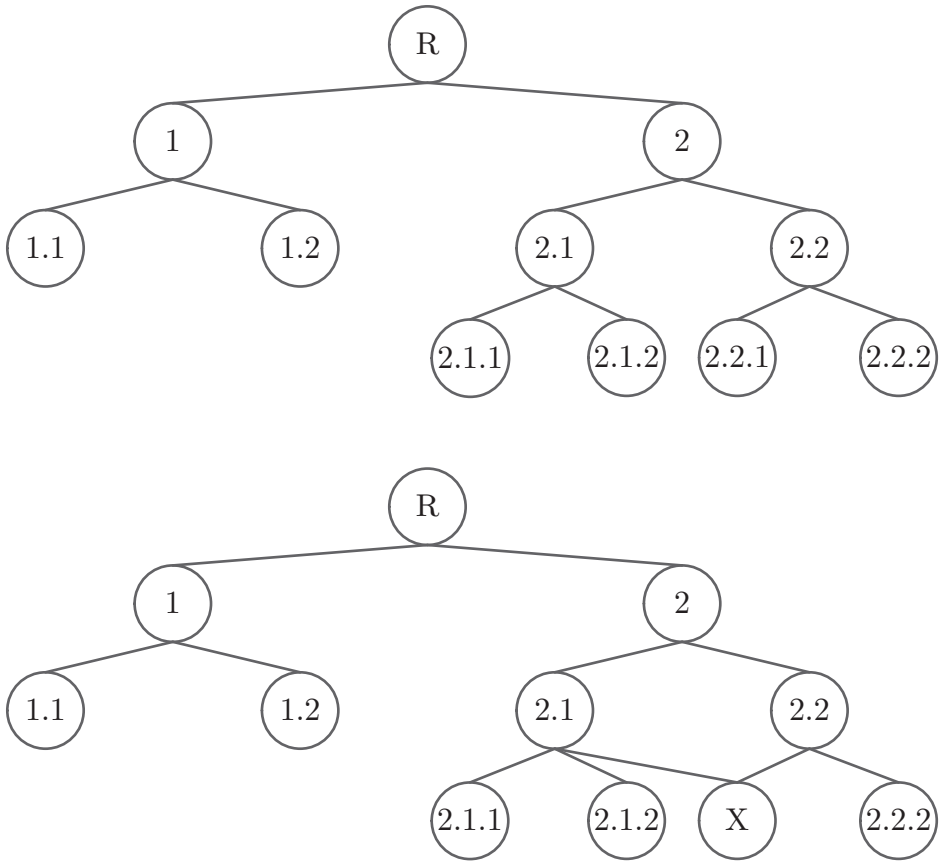


Figura 3.2: Exemplo simples de taxonomia hierárquica de classes baseada em árvore (acima) e em DAG (abaixo), adaptado de Silla e Freitas (2011).

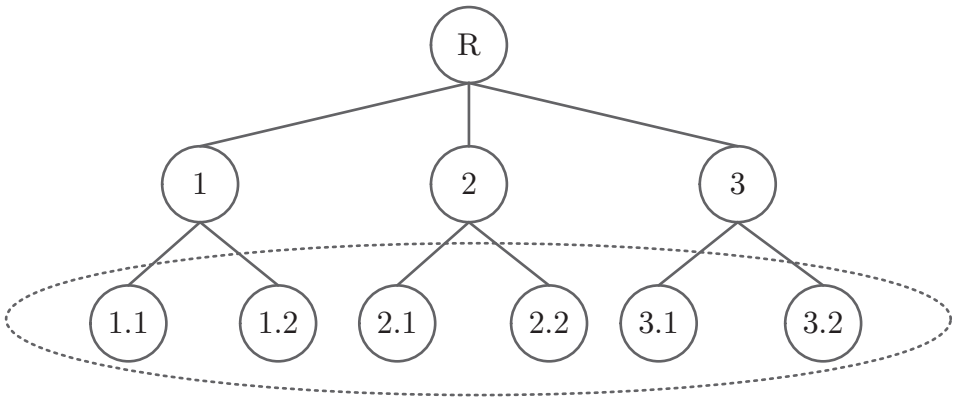


Figura 3.3: Exemplo de predição obrigatória em nós folha - MLNP.

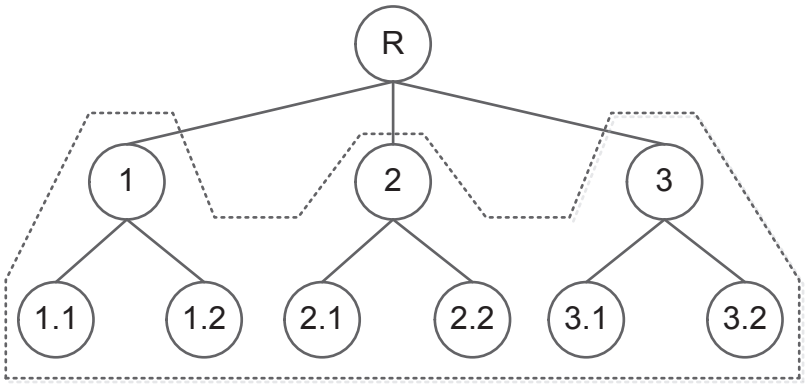


Figura 3.4: Exemplo de predição não obrigatória em nós folha - NMLNP.

3.2.3 *Exploração da estrutura pelo classificador*

Para explorar os problemas de classificação hierárquica, alguns algoritmos têm sido propostos, sendo divididos em três abordagens principais: (i) algoritmos de classificação hierárquica plana; (ii) algoritmos de classificação hierárquica local, subdivididos em local por nó, local por nó pai e local por nível; (iii) algoritmos de classificação hierárquica global (Freitas e Carvalho, 2007). Todas essas abordagens indicam como os classificadores são construídos e não um método de classificação, como o método *top-down* que muitas vezes é citado na literatura como sendo uma das abordagens.

A classificação hierárquica plana, de acordo com Silla e Freitas (2011), é a mais simples abordagem para se lidar com problemas de classificação hierárquica. Consiste em ignorar completamente a hierarquia de classes, geralmente prevendo apenas a classes dos nós folha (MLNP).

Esta abordagem se comporta como um algoritmo de classificação tradicional durante as fases de treinamento e teste, fornecendo uma solução indireta para o problema da classificação hierárquica. Apesar desta abordagem ser muito simples, ela tem a desvantagem de ter que construir um classificador para discriminar entre um grande número de classes (todas as classes da folha), sem explorar informações sobre as relações pais/filhos presentes na hierarquia das classes. A Figura 3.5 exemplifica esta abordagem.

A abordagem de classificação hierárquica local por nó é a mais utilizada na literatura, de acordo com Silla e Freitas (2011) e consiste em treinar um classificador binário para cada nó da hierarquia de classes como mostra a Figura 3.6.

Na abordagem da Figura 3.6, é necessário utilizar n classificadores locais independentes, um para cada classe exceto o nó raiz. Com isso, a quantidade de classificadores a

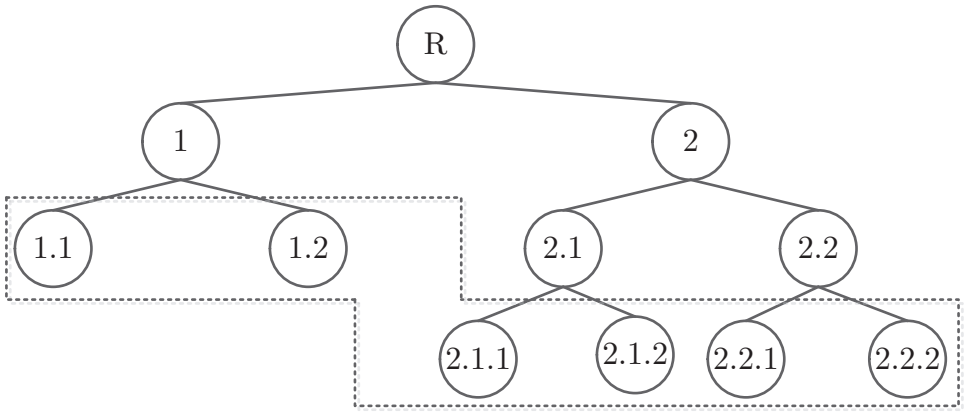


Figura 3.5: Classificação plana multirrótulo usando algoritmo para predição sempre em nós folha (Silla e Freitas, 2011).

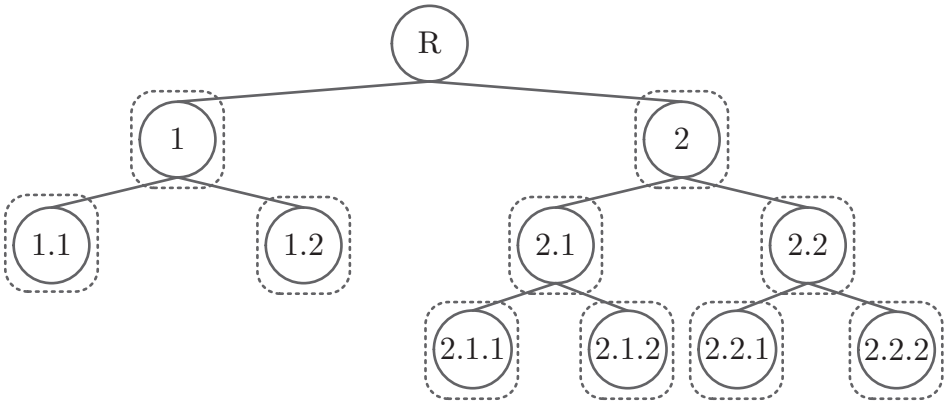


Figura 3.6: Classificação local por nó (os círculos representam as classes e os quadrados pontilhados representam os classificadores) (Silla e Freitas, 2011).

serem treinados pode ser muito grande em situações em que existam muitas classes. Além disso, não há nenhuma garantia de que a hierarquia de classes será respeitada.

Na abordagem de classificação hierárquica local por nó pai, para cada nó pai na hierarquia de classes, um classificador multiclasse é treinado para distinguir entre seus nós filho (Silla e Freitas, 2011).

Considerando que a classificação ocorra de cima para baixo e considerando a taxonomia da Figura 3.7, suponha-se que no primeiro nível o classificador prediz a classe 2. Na sequência, o classificador é treinado com as classes filhas da classe predita anteriormente; no exemplo as classes 2.1 e 2.2, evitando o problema de predições inconsistentes e respeitando o relacionamento entre as classes. Esta abordagem se torna mais complexa se

utilizada em uma estrutura do tipo DAG.

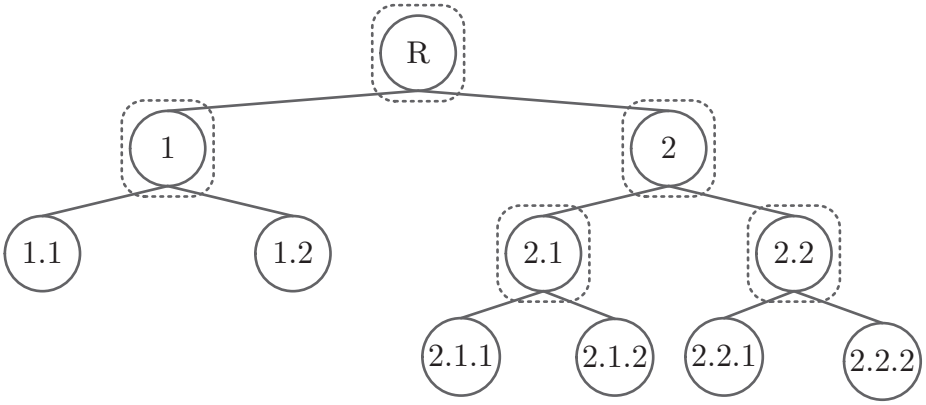


Figura 3.7: Classificação local por nó pai (Silla e Freitas, 2011).

De acordo com Silla e Freitas (2011), a abordagem de classificação hierárquica local por nível é a menos utilizada na literatura e consiste em um classificador multiclasses para cada nível da hierarquia de classes, conforme mostra o exemplo da Figura 3.8.

Esta técnica pode ser usada em estruturas em árvore e em DAG. Com a taxonomia no formato de um DAG, a aplicação desta técnica é mais complexa, visto que pode existir mais de um caminho ligando dois nós neste tipo de estrutura. Assim, uma classe pode pertencer a mais de um nível na hierarquia o que pode trazer redundância entre os classificadores.

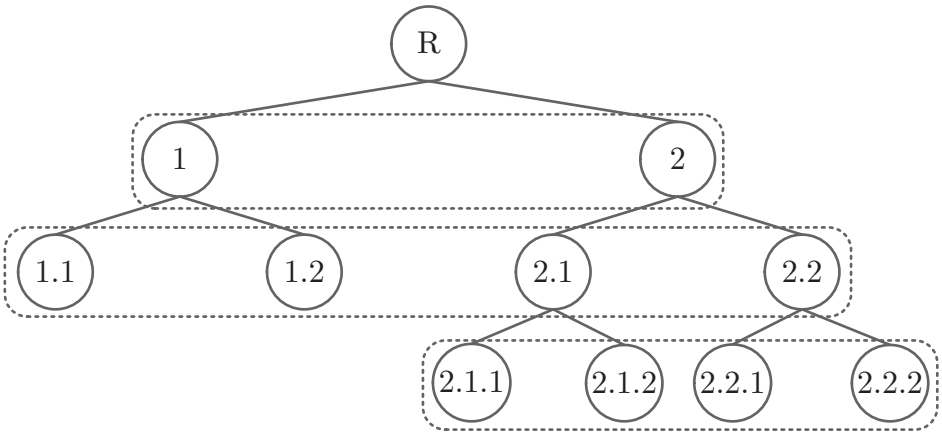


Figura 3.8: Classificação local por nível (Silla e Freitas, 2011).

Na última abordagem chamada de classificação hierárquica global, conforme Silla e Freitas (2011), um único modelo de classificação (relativamente complexo) é construído a partir do conjunto de treinamento, levando em conta a hierarquia das classes como um

todo durante uma única execução do algoritmo de classificação, conforme mostra a Figura 3.9.

Esta abordagem, que será seguida neste trabalho também é conhecida como *big-bang* e tem a vantagem de que o tamanho total do modelo de classificação é geralmente menor em comparação com as outras abordagens. Além disso, o fato de o algoritmo manter as relações de hierarquia entre as classes durante as fases de treinamento e teste faz com que o resultado da predição seja mais facilmente compreendido.

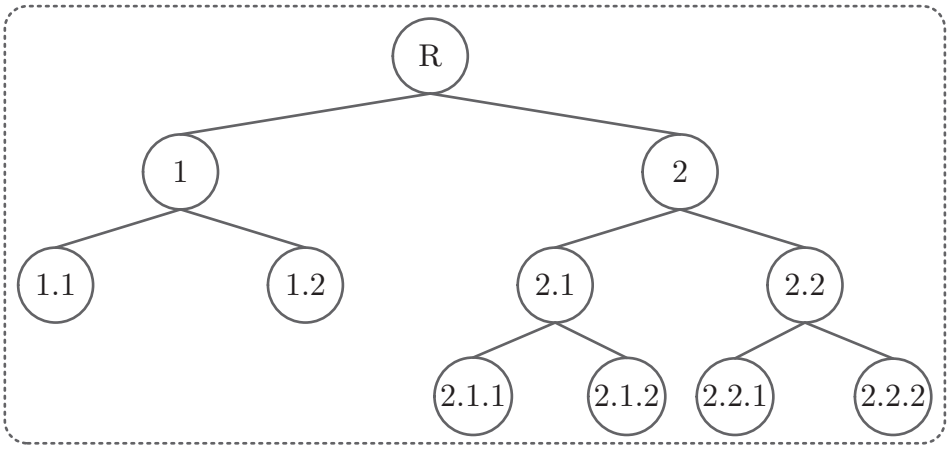


Figura 3.9: Exemplo de classificação hierárquica global (Silla e Freitas, 2011).

3.2.4 Padronização da terminologia para classificação hierárquica

Devido aos inúmeros trabalhos relacionados à classificação hierárquica e às inúmeras técnicas empregadas na busca de soluções para este tipo de problema de classificação, em muitos trabalhos existem diferenças de nomenclaturas para itens relacionados à classificação hierárquica que possuem o mesmo significado. Em seu trabalho, Silla e Freitas (2011) propõem uma unificação da terminologia e um conjunto de símbolos para sua representação, proposta esta que será descrita nas próximas subseções.

Categorização dos diferentes tipos de problemas de classificação hierárquica

O tipo de problema de classificação hierárquica é descrito por meio da tripla $\langle \Upsilon, \Psi, \Phi \rangle$, em que:

- Υ representa o tipo de hierarquia, podendo assumir os valores:
  - T (tree) quando as classes a serem preditas estão arranjadas em árvore;

  - D (DAG) indicando que as classes a serem preditas estão organizadas em um grafo direcionado;

- Ψ indica quando a instância a ser classificada pode ser associada a um único ou a vários caminhos na estrutura de classificação. Pode assumir os valores:

  - *Single Path of Labels* (SPL) quando a instância a ser classificada será associada a um único caminho dentro da hierarquia;
  - *Multiple Path of Labels* (MPL) quando a instância a ser classificada pode ser associada a mais do que um caminho dentro da hierarquia.

- Φ descreve a profundidade das anotações de classes nas instâncias, sendo:

  - *Full Depth* (FD) indicando que todas as instâncias estão anotadas com as classes em todos os níveis, do primeiro nível até os nós folhas da hierarquia;
  - *Parcial Depth* (PD) indicando que pelo menos uma ou mais instâncias possuem anotações parciais, por exemplo, o valor da classe associada a instância em um determinado nível (normalmente nos nós folha) é desconhecido.

Categorização dos diferentes tipos de algoritmos de classificação hierárquica

Um algoritmo para classificação hierárquica pode ser categorizado pela quádrupla $\langle \Delta, \Xi, \Omega, \Theta \rangle$, em que:

- Δ indica quando o algoritmo pode predizer classes em um único ou múltiplos caminhos dentro da hierarquia de classes. Este atributo pode assumir dois valores:

  - *Single Path Prediction* (SPP), indicando que o algoritmo pode associar a uma instância apenas um caminho na hierarquia de classes;
  - *Multiple Path Prediction* (MPP), para indicar que o algoritmo pode associar mais do que um caminho na hierarquia de classes;

- Ξ indica a profundidade do algoritmo. Pode assumir os valores:

  - MLNP que indica que o algoritmo sempre irá associar às instâncias classes presentes em nós folha da hierarquia;
  - NMLNP que indica que o algoritmo pode associar a uma instância classes em qualquer nível da hierarquia.

- **Ω** indica a estrutura da taxonomia de classes suportada pelo algoritmo, podendo assumir os valores:
  - T (tree), indicando que o algoritmo suporta taxonomia de classes representada em árvore;
  - D (DAG), indicando que o algoritmo suporta taxonomia de classes representada em DAG;
- **Θ** indica como o algoritmo explora a taxonomia de classes no processo de classificação, podendo ser:
  - *Local Classifier Per Node* (LCN), conforme demonstrado na Figura 3.6;
  - *Local Classifier Per Level* (LCL), representado na Figura 3.8;
  - *Local Classifier Per Parent Node* (LCPN), demonstrado na Figura 3.7;
  - *Global Classifier* (GC), para indicar que o algoritmo é um classificador global consoante a Figura 3.9.

3.2.5 Comparativo entre as diferentes abordagens existentes para classificação hierárquica

De acordo com Silla e Freitas (2011), pode-se comparar as abordagens para classificação hierárquica em termos de vantagens e desvantagens, conforme demonstrado na Tabela 3.1. Na tabela, as 3 linhas que se referem aos classificadores locais consideram apenas a sua fase de treinamento, ao passo que a linha subsequente considera a fase de teste das três abordagens baseadas em classificadores locais.

3.2.6 Avaliação do desempenho de um sistema classificador

A avaliação de um classificador consiste em aplicar métricas efetivas sobre os resultados obtidos na etapa de testes a fim de quantificar a capacidade de predição correta ou incorreta de um classificador.

A avaliação do desempenho de um modelo de classificação é baseada nas contagens de instâncias de teste corretamente e incorretamente preditas pelo modelo (Tan *et al.*, 2009). A taxa de acerto ou de erro calculada a partir do conjunto de teste também pode ser utilizada para comparar o desempenho relativo de diferentes classificadores em um mesmo domínio.

Tabela 3.1: Resumo das características das diferentes abordagens para classificação hierárquica de acordo com Silla e Freitas (2011).

Abordagem Hierárquica	Vantagens	Desvantagens
Classificação plana	Simplicidade	Ignora completamente a hierarquia de classes
LCN (treinamento)	Simplicidade; Multiclasse nativo;	Pode apresentar problemas de bloqueio; Propenso a inconsistência; Emprega um grande número de classificadores
LCPN (treinamento)	Simplicidade; Utiliza menos classificadores do que o LCN	Pode apresentar problemas de bloqueio; Propenso a inconsistência
LCL (treinamento)	Simplicidade; Emprega um menor número de classificadores	Propenso a inconsistência; O classificador por ter que discriminar entre um grande número de classes (em níveis profundos); Ignora a relação pai-filho durante o treinamento
Todos os classificadores locais	Preserva as restrições naturais nas associações das classes; Considera a hierarquia das classes durante as fases de teste e criação dos conjuntos de treinamento; Generalidade (pode ser usado com qualquer classificador base)	Pode apresentar problemas de bloqueio; Dependendo do tipo de problema, pode criar um conjunto muito complexo de classificadores em cascata, que por sua vez leva a um modelo de classificação complexo; Erros de classificação em uma determinada classe são propagados para todas as classes descendentes
Classificador Global	Preserva as restrições naturais nas associações das classes; Considera a hierarquia das classes durante as fases de treinamento e teste; Modelo único (embora complexo) de decisão	Requer a implementação de um classificador específico

Nos problemas de classificação binária utiliza-se como base uma matriz de confusão, em que as seguintes situações referentes à predição podem ocorrer:

- Verdadeiro Positivo (VP) - O exemplo é predito corretamente como pertencendo à classe positiva;
- Falso Positivo (FP) - O exemplo é predito como pertencendo à classe positiva, mas pertence à classe negativa;
- Verdadeiro Negativo (VN) - O exemplo é predito corretamente como pertencendo à classe negativa;
- Falso Negativo (FN) - O Exemplo é predito como pertencendo à classe negativa, mas pertence a classe positiva.

A Tabela 3.2 demonstra as quatro possibilidades de predição na classificação binária por meio da matriz de confusão em que se pode visualizar as correlações entre a classe predita e a classe verdadeira.

Tabela 3.2: Matriz de confusão para classificação binária.

		Classe Predita	
		Positiva	Negativa
Classe Verdadeira	Positiva	VP	FN
	Negativa	FP	VN

Assim, a Taxa de Acerto (TA) e a Taxa de Erro (TE) de um classificador são obtidas de acordo com a Equação 3.1 e a Equação 3.2 , respectivamente.

$$TA = \frac{|VN| + |VP|}{|VN| + |VP| + |FN| + |FP|} \tag{3.1}$$

$$TE = \frac{|FN| + |FP|}{|VN| + |VP| + |FN| + |FP|} = 1 - TA \tag{3.2}$$

Outras medidas são utilizadas além da taxa de acerto e de erro para avaliar a qualidade de um classificador, tais como Sensibilidade, Especificidade, Precisão e Medida-F.

A Sensibilidade (S) representada na Equação 3.3 mede a capacidade de se predizer uma classe positiva cuja predição está correta, ou seja, ela indica quantos exemplos positivos foram previstos do total de exemplos. Essa medida também é conhecida como Revocação.

$$S = \frac{|VP|}{|VP| + |FN|} \tag{3.3}$$

A Especificidade (E) representada na Equação 3.4 mede a capacidade de se predizer corretamente uma classe negativa, ou seja, quantos exemplos negativos foram preditos do total.

$$E = \frac{|VN|}{|VN| + |FP|} \tag{3.4}$$

A Precisão (P) representada na Equação 3.5 calcula a probabilidade de a predição positiva estar correta em relação a todas as amostras de exemplos.

$$P = \frac{|VP|}{|VP| + |FP|} \tag{3.5}$$

A Medida-F é obtida mediante da combinação da medida de Sensibilidade com a medida de Precisão. Nesta Equação utiliza-se uma constante β que geralmente recebe o valor 1, como mostra a Equação 3.6

$$Medida\text{-}F = \frac{(\beta^2 + 1) * P * S}{\beta^2 * P + S} \tag{3.6}$$

Avaliação de desempenho de um classificador hierárquico

Segundo Kiritchenko *et al.* (2005), as medidas de desempenho utilizadas na classificação plana não são aplicáveis à classificação hierárquica, pois não conseguem diferenciar os diferentes tipos de erros que nela podem ocorrer. As medidas baseadas em distância possibilitam tratar melhor o problema, no entanto elas têm alguns pontos fracos:

1. não são facilmente aplicáveis em modelos baseadas em DAG e multirrótulo;

2. não mudam de acordo com a profundidade da classificação. Uma classificação errada no primeiro nível em que a classe predita está em um nó irmão (*sibling*) da classe correta, e um mesmo erro que ocorra no décimo nível são considerados o mesmo tipo de erro (distância = 2), embora um erro no décimo nível é bem menos prejudicial que um erro no primeiro nível.

Ainda conforme Kiritchenko *et al.* (2005), é desejável que uma métrica de desempenho de um classificador hierárquico tenha as 3 características a seguir:

1. A medida deve considerar classificações parcialmente corretas, como, por exemplo, uma classificação incorreta na hierarquia da Figura 3.10, em que a classe predita para uma instância foi I quando a correta é G, deve ter uma penalização menor do que se para a mesma instância a classe predita for a classe D, uma vez que as classes G e I estão no mesmo subgrafo, o que não acontece com a classe D;

2. A medida deve penalizar erros de acordo com a distância:

   - a medida deve considerar melhor uma classificação parcialmente correta quando a classe predita está um nível acima da classe correta do que quando a classe predita está dois ou mais níveis acima. Por exemplo, predizer a classe E é melhor do que predizer a classe C para uma instância cuja classe correta é a classe G, considerando que a classe E está mais próxima da classe correta;
   - a medida deve considerar pior uma classificação parcialmente incorreta quando a classe predita está um nível abaixo se comparada a seu nó pai, por exemplo, predizer a classe F é pior do que predizer a classe C para uma instância cuja classe verdadeira é G;

3. A medida deve considerar mais graves os erros nos níveis mais altos da hierarquia, por exemplo, predizer a classe I quando a classe verdadeira é G é um erro menos grave do que predizer a classe B quando a classe verdadeira é a A.

Com base nestas premissas, Kiritchenko *et al.* (2005) propõe as medidas *Hierarchical Precision* (hP), *Hierarchical Recall* (hR) e a *Hierarchical F-measure* (hF).

Formalmente, na classificação multiclasse, para cada instância (d_i, C_i) classificada para um subconjunto de classes C'_i, estendem-se os subconjuntos C_i e C'_i com as suas correspondentes classes ancestrais criando os subconjuntos $\hat{C}_i = \{\bigcup_{ck \in C_i} Ancestrais(C_k)\}$ e

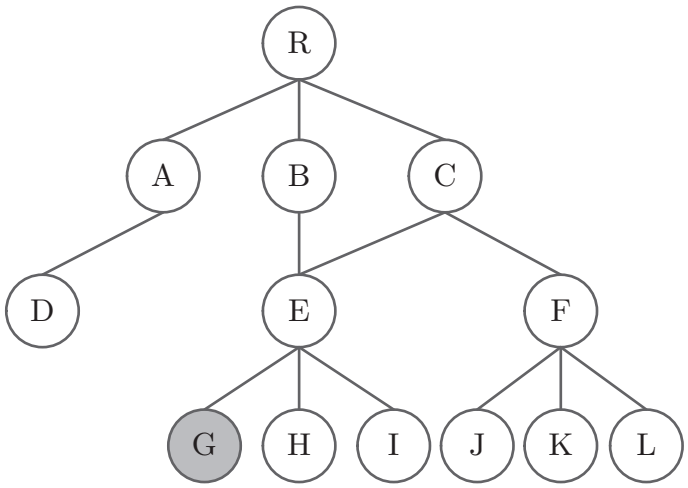


Figura 3.10: Exemplo de hierarquia de classes em forma de DAG. A elipse sólida do nó G representa a classe verdadeira de uma instância (Kiritchenko *et al.*, 2005)

$\hat{C}'_i = \{\bigcup_{ck \in C'_i} Ancestrais(C_k)\}$. As medidas hP e hR são calculadas pela Equação 3.7 e a Equação 3.8, respectivamente.

$$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} \tag{3.7}$$

$$hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|} \tag{3.8}$$

Para exemplificar, considere-se a hierarquia de classes da Figura 3.10. Suponha que a classe predita para uma determinada instância foi F quando a sua classe verdadeira é G. Para calcular as medidas hierárquicas, estende-se o conjunto das classes verdadeiras $C_i = \{G\}$ adicionando todos os ancestrais da classe G: $\hat{C}_i = \{B, C, E, G\}$. Também estende-se o conjunto das classes preditas $C'_i = \{F\}$ adicionando todos os ancestrais de F: $\hat{C}'_i = \{C, F\}$, então, pode-se verificar que a classe C é a única classe predita corretamente por meio do conjunto dado por $|\hat{C}_i \cap \hat{C}'_i| = 1$. Existem $|\hat{C}'_i| = 2$ classes preditas, então $hP = \frac{|\hat{C}_i \cap \hat{C}'_i|}{|\hat{C}'_i|} = \frac{1}{2}$, e existem $|\hat{C}_i| = 4$ classes verdadeiras, então $hR = \frac{|\hat{C}_i \cap \hat{C}'_i|}{|\hat{C}_i|} = \frac{1}{4}$.

A medida hF é obtida pela combinação das medidas hP e hR obtidas usando a Equação 3.9, sendo que usando $\beta = 1$ ter-se-a o mesmo peso para precisão e a sensibilidade.

$$hF\beta = \frac{(\beta^2 + 1) * hP * hR}{\beta^2 * hP + hR}, \beta \in [0, +\infty] \tag{3.9}$$

De acordo com Silla e Freitas (2011), embora nenhuma medida de desempenho usada para classificação hierárquica possa ser considerada a melhor em todos os cenários e aplicações possíveis da classificação hierárquica, a principal razão para se recomendar as medidas hP, hR e hF está no fato de que elas podem ser aplicadas em qualquer cenário da classificação hierárquica, como por exemplo: estruturada em árvore ou DAG, SPL ou MPL, MLNP ou NMLNP.

Curvas Precision-Recall

Em Vens *et al.* (2008), é proposto o uso das curvas precisão-revocação - PR - para avaliar o desempenho do Clus-HMC. As curvas PR refletem a precisão de um classificador em função da sua revocação. Conforme os autores, a utilização das curvas PR permite uma avaliação mais adequada para os problemas de classificação hierárquica multirrótulo pelo fato de que neste tipo de problema as classes individuais possuem um número reduzido de exemplos positivos em relação ao número total de exemplos. Esta característica é presente, por exemplo, nos problemas de classificação funcional dos genes em que apenas um número pequeno de genes possuem uma função em particular, fazendo com que o número de exemplos negativos supere o número de exemplos positivos para as classes. Em conjuntos de dados com estas características, é mais interessante saber o número de exemplos positivos para uma determinada classe do que o número de exemplos negativos.

As curvas PR representam o desempenho de um classificador ao longo de um limiar que assumindo os valores entre 0 e 1 (indicando a probabilidade de uma instância pertencer a uma determinada classe) permite a obtenção de resultados com melhor precisão ou melhor revocação. Após obterem-se os valores da precisão e da revocação ao longo dos limiares, uma curva é desenhada em um plano cartesiano com seus pontos.

Para se comparar classificadores em relação a todos os resultados, a medida da AUPRC é utilizada, sendo que quanto mais perto de 1 for o seu valor, melhor é o classificador. A vantagem da curva PR é permitir a avaliação de um classificador ao longo de seus vários resultados para o limiar estabelecido, além de permitir uma análise mais detalhada demonstrando em quais limiares o classificador obteve um melhor ou pior resultado.

A diferença fundamental entre a tradicional curva *Receiver Operating Characteristics* (ROC) e a curva PR está no fato de que a primeira representa como a quantidade de exemplos corretamente classificados como VP varia em relação à quantidade de exemplos incorretamente classificados como FN; já as curvas PR demonstram a proporção entre

o número de exemplos corretamente classificados como VP em relação ao número de exemplos corretamente classificados como FP. Como resultado, na análise visual do gráfico, na curva ROC os melhores resultados tendem à proximidade com o ponto (0,1) do gráfico ao passo que nas curvas PR o melhor resultado tende ao ponto (1,1).

Davis e Goadrich (2006) fazem uma revisão sobre as curvas ROC e as curvas PR, comparando-as e demonstrando a aplicabilidade das duas diferentes medidas. De maneira resumida, o autor demonstra que as curvas PR são recomendadas para se comparar resultados de diferentes classificadores quando o conjunto de dados utilizado possui desbalanceamento na distribuição das classes. O autor demonstra por meio de um exemplo o uso dos dois métodos para comparar dois algoritmos e como as curvas PR podem trazer a tona detalhes que as curvas ROC não apresentam quando se trata de conjuntos de dados com desbalanceamento na distribuição das classes. O exemplo dado pelo autor está reproduzido na Figura 3.11 em que se pode observar que usando as curvas ROC (à esquerda), a diferença no desempenho dos dois algoritmos é bem menos significativa do que na representação dos resultados da curva PR (à direita).

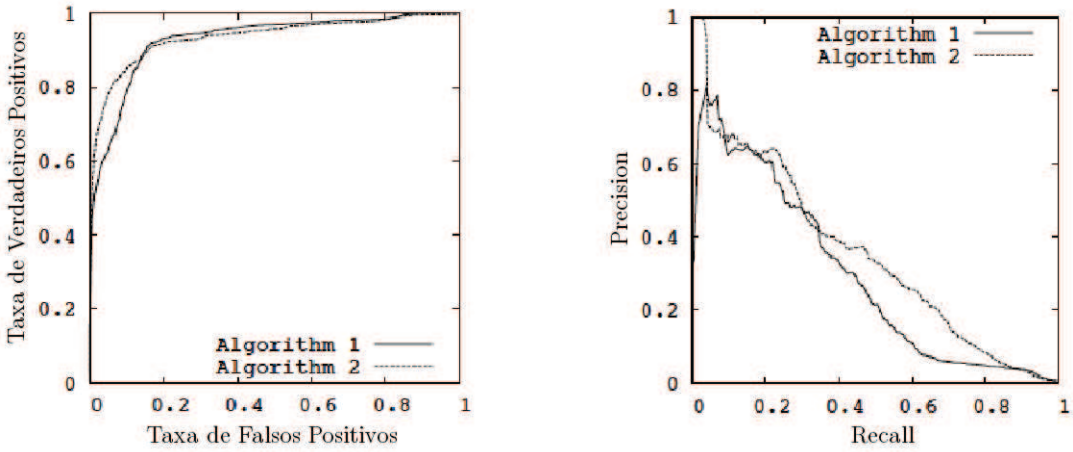


Figura 3.11: Diferenças entre as curvas ROC e PR (Davis e Goadrich, 2006).

Os conceitos de classificação hierárquica até aqui abordados, bem como as medidas de desempenho de classificadores hierárquicos, são utilizados neste trabalho para analisar o desempenho do classificador baseado em FCA proposto. A próxima Seção apresenta os principais trabalhos sobre classificação hierárquica.

3.3 Principais Trabalhos em Classificação Hierárquica

Existe um grande número de trabalhos publicados sobre problemas de classificação hierárquica utilizando diversas abordagens e técnicas diferentes. Em seu estudo e proposta de padronização de nomenclaturas Silla e Freitas (2011) elenca os principais trabalhos classificando-os de acordo com a terminologia proposta na Seção 3.2.4.

Para efeito deste trabalho, serão consideradas apenas as propostas que utilizam classificadores globais em estruturas de árvore ou DAG. Na Tabela 3.3, estão elencados os principais trabalhos que utilizam a abordagem de classificação global com estas características.

Tabela 3.3: Principais trabalhos em classificação hierárquica utilizando classificador global e classes em árvore ou DAG de acordo com Silla e Freitas (2011).

Abordagem (Θ)	Estrutura (Ω)	Lista de trabalhos
Classificador Global	Tree	(Labrou e Finin, 1999), (Wang *et al.*, 1999), (Wang e Senqiang, 2001), (Clare e King, 2003), (Blockeel *et al.*, 2006), (Cai e Hofmann, 2004), (Cai e Hofmann, 2007), (Dekel *et al.*, 2004a), (Dekel *et al.*, 2004b), (Peng e Choi, 2005), (Rousu *et al.*, 2005), (Rousu *et al.*, 2006), (Astikainen *et al.*, 2008), (Silla. e Freitas, 2009) e (Qiu *et al.*, 2009)
	DAG	(Kiritchenko *et al.*, 2005), (Kiritchenko *et al.*, 2006) , (Alves *et al.*, 2008), (Dimitrovski *et al.*, 2011), (Vens *et al.*, 2008), (Aleksovski *et al.*, 2009), (Otero *et al.*, 2009), (Wang *et al.*, 2009)

A Tabela 3.4 mostra os principais trabalhos envolvendo classificação hierárquica classificados conforme a nomenclatura proposta por Silla e Freitas (2011) para representar as características de cada algoritmo.

Tabela 3.4: Características dos principais trabalhos em classificação hierárquica utilizando classificador global e classes em árvore ou DAG de acordo com Silla e Freitas (2011), sendo (Δ) a abordagem, (Ω) a estrutura das classes, (Ξ) a cardinalidade da predição e (Θ) a profundidade da predição.

$\langle \Delta, \Xi, \Omega, \Theta \rangle$	Lista de trabalhos
⟨ GC, T, SPP, MLNP ⟩	(Qiu *et al.*, 2009)
⟨ GC, T, SPP, NMLNP ⟩	(Labrou e Finin, 1999) e (Silla. e Freitas, 2009)
⟨ GC, T, MPP, NMLNP ⟩	(Clare e King, 2003) , (Blockeel *et al.*, 2006), (Rousu *et al.*, 2005), (Rousu *et al.*, 2006), (Dimitrovski *et al.*, 2011) e (Aleksovski *et al.*, 2009)
⟨ GC, D, SPP, NMLNP ⟩	(Otero *et al.*, 2010)
⟨ GC, D, MPP, NMLNP ⟩	(Alves *et al.*, 2008), (Vens *et al.*, 2008).

3.4 Classificação Hierárquica Usando Classificadores Globais

Um dos primeiros trabalhos utilizando classificadores globais foi Rocchio (1971) no qual foi utilizada a ideia de agrupamentos de classes em que novos exemplos são associados à classe mais próxima com base em uma medida de distância entre o novo exemplo de teste e cada uma das classes. Um exemplo desta abordagem é proposto em Labrou e Finin (1999). Neste trabalho, o sistema classifica páginas web em um subconjunto de categorias hierárquicas definidas para classificação de páginas Web do Yahoo!. Este método é específico para mineração de texto e durante a fase de testes cada novo documento é comparado em termos e similaridade com as categorias.

Alguns tipos de classificadores globais são baseados na conversão do problema de classificação hierárquica em um problema de classificação multiclasse, como é o caso das propostas Kiritchenko *et al.* (2005) e Kiritchenko *et al.* (2006). Para possibilitar a predição de qualquer classe na hierarquia, a fase de treinamento é modificada para considerar todas as classes da hierarquia por meio da inclusão dos nós não-folha com a informação das classes ascendentes. Na fase de testes, o algoritmo não leva em consideração a hierarquia das classes, ou seja, tende a erros de inconsistência na predição. Por esta razão, os autores propõem uma fase de pós-processamento que avalia todas as predições a fim de garantir que as restrições hierárquicas estejam sendo respeitadas.

A proposta de Clare e King (2001) utiliza o algoritmo C4.5 baseado no método de árvore de decisão, para classificar um conjunto de dados de fenótipos catalogados no

modelo FunCat. A proposta utiliza uma abordagem de classificação plana e a predição das classes é feita em cada nível da hierarquia, independentemente dos outros níveis, numa sequência *top-down*. Os autores adaptaram o algoritmo C4.5 para suportar multirrótulo, modificando o cálculo de entropia utilizado para decidir qual atributo é selecionado para um determinado nó na árvore de decisão a ser construída. Para isso, foi utilizada uma fórmula de probabilidade para a escolha das possíveis classes de cada nó.

Posteriormente, em Clare e King (2003), a proposta anterior foi modificada para um modelo global de classificação hierárquica. Isto envolveu mudanças na leitura e armazenamento das classes, na forma de encontrar as melhores classes para representar um nó e na execução dos cálculos de entropia de forma ponderada. Este cálculo levou em conta o fato que as classes mais gerais tendem a ter uma entropia mais baixa se comparada com as classes mais específicas; no entanto, as classes mais específicas proporcionam um maior conhecimento sobre a função biológica de uma proteína, e, portantom a fórmula foi ponderada para valorizar estas classes.

Em Vens *et al.* (2008), os autores criaram um modelo de classificação hierárquica baseado no método de árvore de decisão para estruturas do tipo DAG. No trabalho, são abordadas três técnicas de classificação, e cada solução aborda o problema de classificação hierárquica de maneiras diferentes: o primeiro, chamado de CLUS-SC (*Single-Label Classification*), utiliza o método de classificação local e cria uma árvore de decisão para cada classe ignorando a hierarquia entre elas; o segundo, chamado de CLUS-HSC (*Hierarchical Single-Label Classification*), é uma variação do primeiro algoritmo, mas neste caso levando em conta a hierarquia na fase de treinamento; e o último, CLUS-HMC (*Hierarchical Multi-Label Classification*), utiliza o método de classificação global, levando em conta todas as classes na predição por meio de uma única árvore de decisão.

O modelo é definido com base no *framework Predictive Clustering Trees* (PCT), que define uma árvore de decisão como uma hierarquia de grupos, em que o nó raiz corresponde a um grupo contendo todos os dados, e que, de maneira recursiva, é particionado em grupos menores. Este *framework* foi construído utilizando o algoritmo *Top-down Induction of Decision Trees* (TDIDT). A ideia geral consiste em particionar recursivamente o conjunto de dados de forma a reduzir a variância dentro do agrupamento, maximizar a homogeneidade do agrupamento e melhorar o desempenho da predição.

A proposta inicial foi desenvolvida para estruturas no formato de árvore. De acordo com Vens *et al.* (2008), para a tarefa de classificação hierárquica multiclasse, as classes dos exemplos são representadas por meio de vetores com valores booleanos (o i-ésimo valor do vetor é 1 se o exemplo pertencer à classe ci, caso contrário será 0). A Figura 3.12 mostra um exemplo desta operação, em que o subconjunto de classes {1;2;2.2} indicadas

em negrito na hierarquia, é representado como um vetor v_i= [1;1;0;1;0].

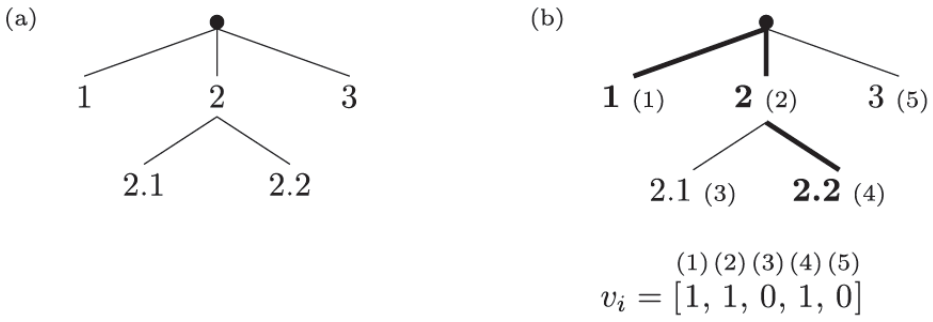


Figura 3.12: (a) Representação de uma pequena hierarquia. (b) Conjunto das classes {1,2,2.2} cujo caminho é indicado em negrito na hierarquia representado em um vetor.

Assim, a variância do conjunto de dados é calculada como sendo a distância média quadrada entre cada classe do exemplo v_i, e a média das classes do conjunto $\vec{v}$, como mostra a equação 3.10 retirada de Vens *et al.* (2008).

$$Var(S) = \frac{\sum_i d(v_i, \vec{v})^2}{|S|} \tag{3.10}$$

Para considerar a similaridade nos níveis da hierarquia, visto que há mais similaridade nos níveis mais altos do que nos mais baixos, os autores utilizaram a distância euclidiana ponderada, conforme mostra a Equação 3.11.

$$d(v_1, v_2) = \sqrt{\sum_i w(c_i) * (v_{1,i} - v_{2,i})^2} \tag{3.11}$$

Sendo $v_{k,i}$, o i-ésimo valor do vetor de classes v_k de uma instância x_k, e w(c) o peso da classe, o qual diminui em direção às classes folha da hierarquia, ou seja, ele diminui com a profundidade ($w(c) = w_0^{folha(c)}$, com $0 < w_0 < 1$).

Para trabalhar com estruturas em DAG, os autores propuseram uma modificação no cálculo do peso das classes. Como na estrutura em árvore, a variância é computada consoante a distância euclidiana ponderada entre os vetores de classe de acordo com a profundidade da classe na hierarquia, para a estrutura em DAG foi levado em consideração o fato de que a classe pode ter várias profundidades dependendo do caminho seguido do

nível superior até a classe.

Assim, de acordo com Vens *et al.* (2008), o peso ($w(c) = w_0^{folha(c)}$) calculado para a estrutura hierárquica em árvore foi reescrito utilizando uma relação de recorrência $w(c) = w_0 * w(pai(c))$ sendo $pai(c)$ a classe pai de c, e os pesos da classes de nível superior igual a w_0. Esta relação de recorrência generaliza a hierarquia em uma classe pode ter múltiplos pais pela agregação de $w(pai(c))$ no cálculo do peso.

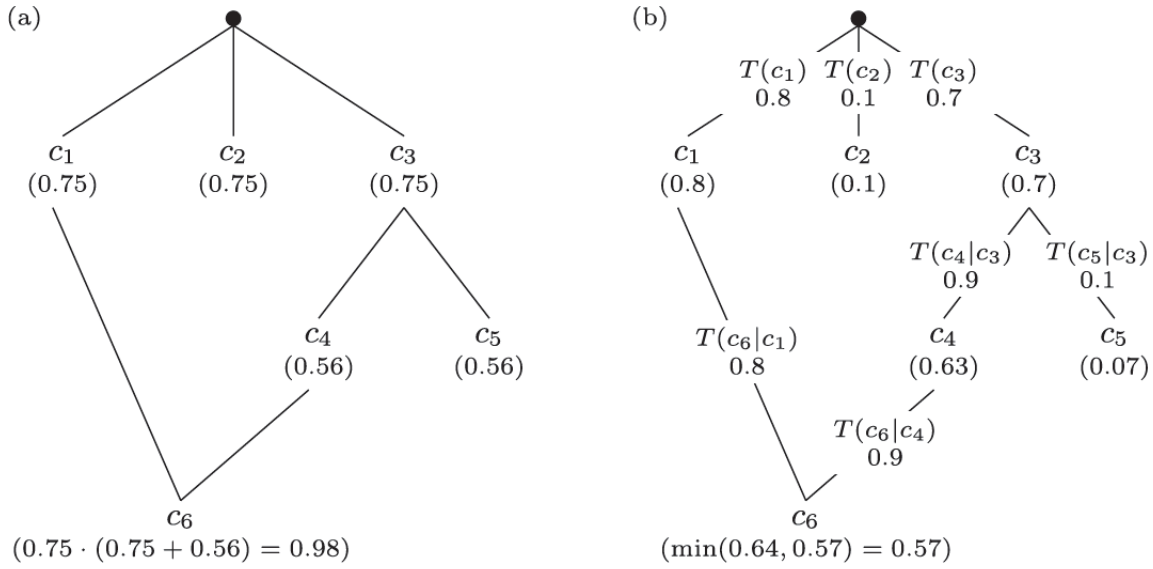


Figura 3.13: Exemplo da aplicação da hierarquia de classes estruturada em DAG desenvolvido por Vens *et al.* (2008)

A técnica Clus-HSC também foi modificada para suportar a estrutura do tipo DAG. Para isso, os autores fizeram alterações no cálculo de predição do modelo. Desse modo, ao invés de calcular a probabilidade condicional $P(c_j|pai(c))$ é calculada a probabilidade condicional $P(c_j|pai_j(c))$. Para fazer a predição, o produto da regra $P(c) = P(c_j|pai_j(c)) * P(pai(c))$, pode ser aplicado para cada classe pai $pai(c)$.

A Figura 3.13 mostra um exemplo da hierarquia de classe estruturada como DAG desenvolvido em Vens *et al.* (2008). Em (a) o peso de cada classe é computado pelo Clus-HMC utilizando o esquema de peso $w(c) = w_0 \sum_j w(paij(c))$ e $w_0 = 0.75$. Em (b), por meio da árvore construída pelo CLUS-HSC a probabilidade de predição de uma classe (c) é dada pela fórmula $P(c) = min_j P(c_j|pai_j(c)) * P(pai_j(c))$.

Os testes no trabalho de Vens *et al.* (2008) foram feitos com vinte e quatro bases de dados que descrevem diferentes aspectos dos genes no genoma do organismo *Saccharomyces cevisiae*. Estas bases de dados foram classificadas utilizando a descrição funcional baseada

no FunCat, que organiza os dados na forma de árvores, e na descrição funcional baseada no GO, que organiza os dados na forma de DAG. Os resultados foram avaliados por meio das medidas de precisão e revocação e pelas curvas PR. Conforme os autores, o Clus-HMC teve um desempenho preditivo melhor, se comparado com as outras técnicas, tanto para as estruturas de árvore como para DAG. Além disso, o tamanho da hierarquia de quando usado o Clus-HMC é muito menor.

Em Otero *et al.* (2010), os autores usam uma abordagem baseada em otimização por colônia de formigas para fazer a predição de classes para uma base estruturada de forma hierárquica. Os principais aspectos do algoritmo são:

- *Representação do problema:* O problema é mapeado para um grafo o qual é utilizado pelas formigas para busca das soluções. As formigas se movimentam de forma randômica em um grafo $G_c = (C, L)$, em que o conjunto C representa os vértices do grafo e L representa as arestas entre os vértices, na busca por soluções. O grafo G_c representa o espaço de busca;

- *Construção das soluções:* Cada formiga constrói incrementalmente soluções candidatas se movimentando pelo grafo G_c. Os vértices a serem visitados são definidos por meio de um método estocástico com base na quantidade de feromônio depositado em cada vértice e uma heurística associada ao problema. Cada solução candidata é representada como uma trilha no grafo G_c;

- *Comunicação indireta:* Após a criação de uma solução candidata a formiga avalia a solução para estabelecer quanto feromônio irá depositar na trilha. Em geral, a quantidade de feromônio depositada é proporcional à qualidade da solução. O depósito de feromônio aumenta a probabilidade de que os vértices/arestas serão usados por outras formigas em outras soluções candidatas.

O algoritmo proposto em Otero *et al.* (2010) foi testado e comparado aos algoritmos CLUS-SC e CLUS-HMC por meio de 16 bases de dados organizadas em árvore (bases de dados FunCat) e DAG (*Gene Ontology*) utilizando medidas baseadas em AUPRC. Os resultados apresentados pelo algoritmo proposto se mostraram competitivos e promissores.

Em Alves *et al.* (2008), autores construíram um modelo de classificação hierárquica denominado de *Multi-Label Hierarchical Classification with an Artificial Immune System* (MHC-AIS) o qual utiliza conceitos de um Sistema Imunológico Artificial (SAI) e gera regras no formato SE-ENTÃO. Os autores apresentam duas versões do MHC-AIS: global e local. Na versão local é construído um classificador para cada classe do domínio. Neste classificador, o consequente das regras descobertas apenas diferencia se um exemplo pode

ou não ser associado à classe para a qual aquele classificador foi treinado, ou seja, para cada nó da estrutura hierárquica. Já na versão global um único classificador é gerado para distinguir todas as classes do domínio da aplicação tratada. Nesta versão, uma ou mais classes são representadas no consequente.

Conforme Alves *et al.* (2008), a fase de treinamento do MHC-AIS é realizada por dois procedimentos principais, chamados de *Sequential Covering* (SC) e *Rule Evolution* (RE). O procedimento SC iterativamente chama o procedimento RE até quase todos os "antígenos" (exemplos) serem cobertos pelas regras geradas. O procedimento RE essencialmente evolui os "anticorpos" (regras de classificação) que são utilizados para classificar os "antígenos". Em seguida, o melhor "anticorpo" evoluído é adicionado ao conjunto de regras descobertas.

Após a criação de cada "anticorpo", uma medida de qualidade (*fitness*) é calculada levando em consideração o produto dos valores de sensibilidade e de especificidade de cada "anticorpo". O MHC-AIS mantém a consistência do conjunto hierárquico durante a construção do modelo global, conforme mostra a Equação 3.12.

$$fit(c^i_{k*}) = max[fit(c^i_{k*}), fit(c^i_k)], c^i_{k*} \in Ancestral(c^i_k) \tag{3.12}$$

Ou seja, se o *fitness* de algum ancestral da classe c^i_{k*} for menor do que o *fitness* da sua classe descendente c^i_k, então o *fitness* do c^i_k é atribuído para a classe ancestral.

Os dois modelos foram avaliados pelas medidas de sensibilidade, precisão e medida-F, e, apesar de ambos terem suas vantagens e desvantagens, o modelo global apresentou um melhor desempenho geral se comparado com o modelo local.

Outras duas abordagens foram publicadas sobre o tema classificação hierárquica durante o desenvolvimento deste trabalho: Bi e Kwok (2011) propuseram um método para classificação hierárquica que busca por meio de algoritmos gulosos encontrar o melhor subgrafo da árvore ou DAG para classificar exemplos; já Pugelj e Džeroski (2011) propuseram um método para classificação hierárquica baseado no *k-Nearest Neighbours Method*.

3.5 Considerações Finais

Neste capítulo, foram relacionados os principais trabalhos na atualidade ligados à classificação hierárquica de dados, trazendo elementos necessários para análise do modelo HMCS-FCA proposto em relação aos trabalhos que compõem o estado da arte no tema. Dentre os inúmeros trabalhos elencados, merecem destaque as propostas Clare e King (2001), Clare e King (2003), Vens *et al.* (2008) e Otero *et al.* (2010).

Capítulo

4

Modelo Proposto

4.1 Considerações Iniciais

A predição da função de proteínas possui muitas particularidades que precisam ser consideradas ao se desenvolver uma solução de classificação, principalmente por tratar-se de um problema de classificação hierárquica em que a taxonomia das classes está estruturada em forma de árvore ou DAG e o algoritmo deve considerar esta hierarquia no processo de classificação.

Atualmente, a maioria dos algoritmos de classificação desenvolvidos para suportar esse tipo de estrutura não avalia o modelo hierárquico como um todo (abordagem global), o que pode alterar o resultado preditivo das amostras. Também deve-se levar em consideração que o uso de ontologias para predição de funções de proteínas tem sido bastante frequente, e as ontologias FunCat e GO têm sido as preferidas pelos pesquisadores da área. Os termos do FunCat e da GO são estruturados na forma de árvore e DAG respectivamente, sendo que neste segundo tipo de ontologia um termo "filho", pode estar conectado a um ou mais termos "pais", tornando a solução ainda mais complexa. Outra questão importante diz respeito aos resultados obtidos pelo algoritmo, que devem ser compreensíveis o suficiente para que se possam expressar facilmente os conhecimentos alcançados.

4.2 Descrição do Modelo Proposto

A ideia central deste trabalho é criar um classificador capaz de ler uma base de treinamento com várias instâncias previamente classificadas de acordo com uma hierarquia de classes (taxonomia), e por meio das técnicas da FCA criar um modelo capaz de predizer as classes para novas instâncias não apresentadas ao classificador na fase de treinamento.

A FCA possui um grande potencial para representação de conhecimento quando estes estão organizados sob uma hierarquia explícita mediante o emprego da técnica de adição de conhecimento a um contexto abordada na Seção 2.2.6. Para a aplicação proposta, os conceitos formais calculados a partir de um contexto formal, criado com as informações das instâncias de treinamento e com a adição do conhecimento para incluir as informações da taxonomia de classes, permitem representar os dados que serão usados para indução sobre novos exemplos abstraindo a estrutura da sua taxonomia, possibilitando que o mesmo algoritmo seja empregado tanto para taxonomias representadas por árvores quanto para taxonomias representadas por meio de DAG. Este nível de abstração quanto à estrutura das taxonomias é um dos principais fatores que motivaram esta investigação.

Uma característica da FCA que impõe algumas limitações para seu uso extensivo é o fato de que o cálculo de todos os conceitos formais para um contexto pode apresentar um comportamento exponencial no seu pior caso (Kuznetsov, 2001). Embora este comportamento raramente aconteça, o custo computacional quando usa-se FCA para determinadas aplicações pode ser proibitivo. Como esta proposta é uma investigação inicial sobre o uso de FCA para classificação hierárquica, esta limitação será contornada neste trabalho com o uso de mecanismos de poda para reduzir o tamanho do modelo.

Considerando a classificação proposta por Silla e Freitas (2011), abordada na seção 3.2.4 (página 41) deste trabalho, o modelo proposto pode ser classificado pela quádrupla $\langle \Delta, \Xi, \Omega, \Theta \rangle$ que representa: se o classificador faz predições monorrótulo ou multirrótulo; se o classificador faz predições em qualquer nível ou apenas em nós folha da hierarquia; o tipo de taxonomia suportada; e a forma como o classificador explora a hierarquia, como:

- Δ: MPP - *Multiple Path Prediction*;
- Ξ: NMLNP - *Non-Mandatory Leaf Node Prediction*;
- Ω: T ou D - *Tree or* DAG;
- Θ: GC - *Global Classifier*;

O modelo proposto terá as etapas de pré-processamento de dados, treinamento do classificador, testes do classificador e avaliação dos resultados. Estas etapas serão descritas nas subseções 4.2.1, 4.2.2, 4.2.3 e 4.2.4 respectivamente.

4.2.1 Pré-processamento dos dados

Nesta etapa, as instâncias para treinamento do classificador serão selecionadas e algumas atividades de pré-processamento são realizadas a fim de adequar os dados brutos ao contexto do classificador proposto. As atividades de pré-processamento empregadas foram: transformação dos dados de multirrótulo para monorrótulo; substituição dos valores ausentes; remoção dos atributos sem influência; e a discretização dos dados. Estas atividades estão descritas na sequência.

Como a maioria das bases de dados hierárquicas multirrótulo disponíveis apresentam em suas anotações mais do que uma classe associada a cada exemplo, é necessário adequá-las para utilização dos algoritmos de discretização. O Algoritmo 1 representa como esta transformação é feita: nele, as funções $atts(I)$ e $class(I)$ separam em cada instância as informações de atributos e classes, e a Tabela 4.1 exemplifica como a transformação é feita sobre os dados, em que se pode observar que a instância i_1 (multirrótulo) deu origem às instâncias $i_{1.1}$, $i_{1.2}$ e $i_{1.3}$. Esta transformação apenas separou as associações multirrótulo das instâncias em associações monorrótulo com base na classe folha associada à instância e sem levar em consideração os ascendentes das classes.

Algoritmo 1 Transformação do conjunto de dados de hierárquico multirrótulo para hierárquico monorrótulo.

Input: Um conjunto de dados ARFF hierárquico multirrótulo H contendo um conjunto de instâncias I, em que cada instância I está associada a 1 ou mais classes;
Output: Um conjunto de dados ARFF hierárquico monorrótulo P;

1. $P := \emptyset$
2. **for all** $I \in H$ **do**
3. $\quad Iattr := attr(I)$
4. $\quad Iclass := class(I)$
5. $\quad$ **for all** $c \in Iclass$ **do**
6. $\quad\quad P := P \cup \{Iattr \cup c\}$
7. $\quad$ **end for**
8. **end for**
9. **return** P

A taxonomia hierárquica das classes será extraída dos arquivos de entrada no formato *Attribute-Relation File Format* (ARFF) hierárquicos e representada por meio de um reticulado de conceitos que pode representar as hierarquias de classes tanto em formato de

Tabela 4.1: Exemplo da transformação feita nos dados de multirrótulo para plano. A instância i_1 multirrótulo foi desdobrada nas instâncias $i_{1.1}$, $i_{1.2}$ e $i_{1.3}$.

Instância	$Atributo_1$	$Atributo_2$	$Atributo_3$	*Classes*
i_1	3	4	6	$c_1@c_2@c_3$
i_2	8	5	3	$c_2@c_4$
...				
$i_{1.1}$	3	4	6	c_1
$i_{1.2}$	3	4	6	c_2
$i_{1.3}$	3	4	6	c_3
$i_{2.1}$	8	5	3	c_2
$i_{2.2}$	8	5	3	c_4

árvore quanto em formato de DAG. O formato ARFF é um formato adotado pelo software *Waikato Environment for Knowledge Analysis* (WEKA) (Hall *et al.*, 2009) que será utilizado para pré-processamento dos dados.

A substituição dos valores ausentes em atributos, comumente encontrados nos conjuntos de dados utilizados para experimentos envolvendo a predição de proteínas foi feita pela substituição deles pela moda ou pela média dos valores presentes nas outras instâncias.

Em um conjunto de dados pode ocorrer a existência de atributos que não agregam nenhuma informação que diferencie uma instância das outras, como por exemplo um atributo com o mesmo valor para todas as instâncias. Estes atributos são removidos na etapa de pré-processamento.

Para encerrar o pré-processamento, é realizada a discretização dos dados a qual tem por objetivo transformar os atributos numéricos (contínuos) e nominais em atributos binários uma vez que isto é um pré-requisito para usá-los com a FCA. A discretização foi parametrizada para discretizar os atributos contínuos em 50 atributos discretos, bem como manter a frequência dos atributos equalizada. A discretização dos dados pode trazer com ela algumas questões que precisam ser melhor investigadas e tratadas no contexto deste trabalho. O alto número de atributos discretos (50) para representar cada atributo contínuo aqui proposto justifica-se pelo fato de que em conjuntos de dados com um número reduzido de atributos, após a sua discretização, muitas instâncias podem ficar iguais em termos de atributos. Se isto acontecer ter-se-ão no mesmo conjunto de dados, instâncias com os mesmos atributos associadas a classes diferentes, o que não é uma situação desejável, pois poderá confundir o classificador.

A Figura 4.1 demonstra as operações que serão realizadas nesta etapa. As entradas

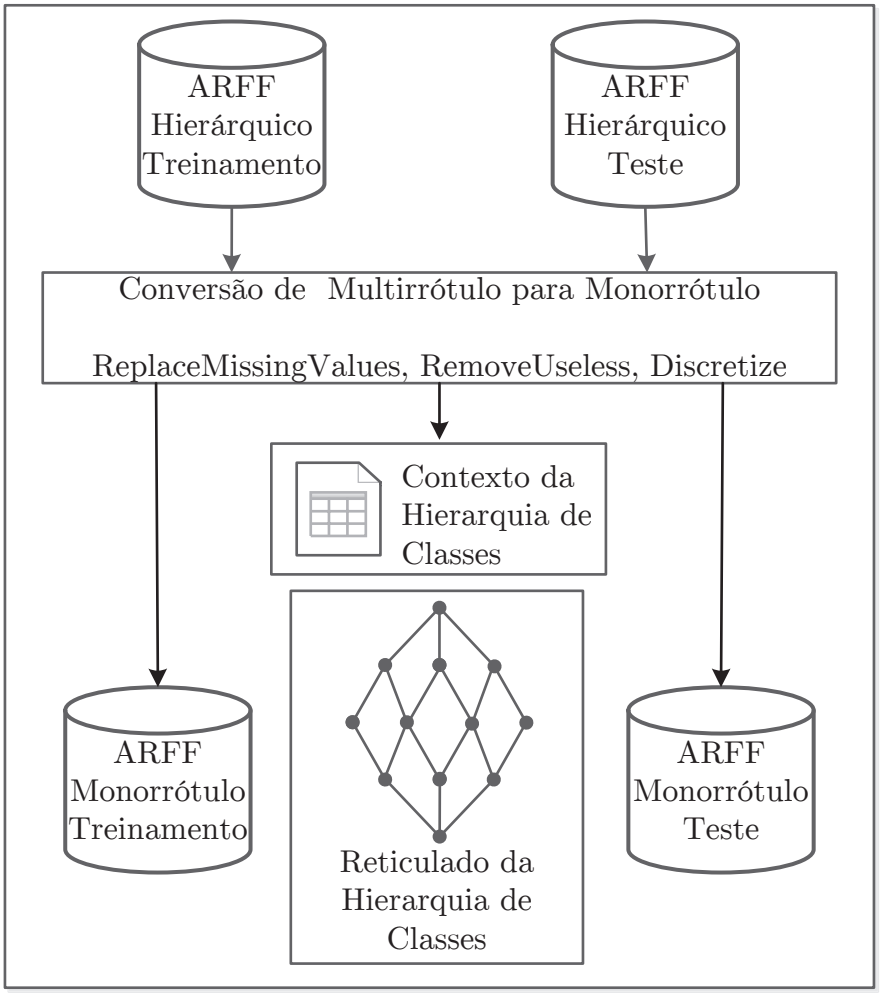


Figura 4.1: Operações realizadas na etapa de pré-processamento do modelo proposto.

são as bases de dados no formato ARFF hierárquico, já as saídas são as bases de dados discretizadas no formato ARFF plano e o reticulado representando a hierarquia de classes.

4.2.2 Treinamento do classificador

Esta etapa pode ser dividida nas subetapas:

1. Montagem de um contexto formal com as instâncias de treinamento;
2. Aumento do contexto com informações de fundo por meio dos métodos de ampliação de um contexto formal para adição das informações sobre a hierarquia das classes ao contexto formal. Aqui serão empregadas as técnicas de FCA descritas na Seção 2.2.6;

3. Cálculo do reticulado de conceitos ou apenas dos conceitos a partir do contexto ampliado.

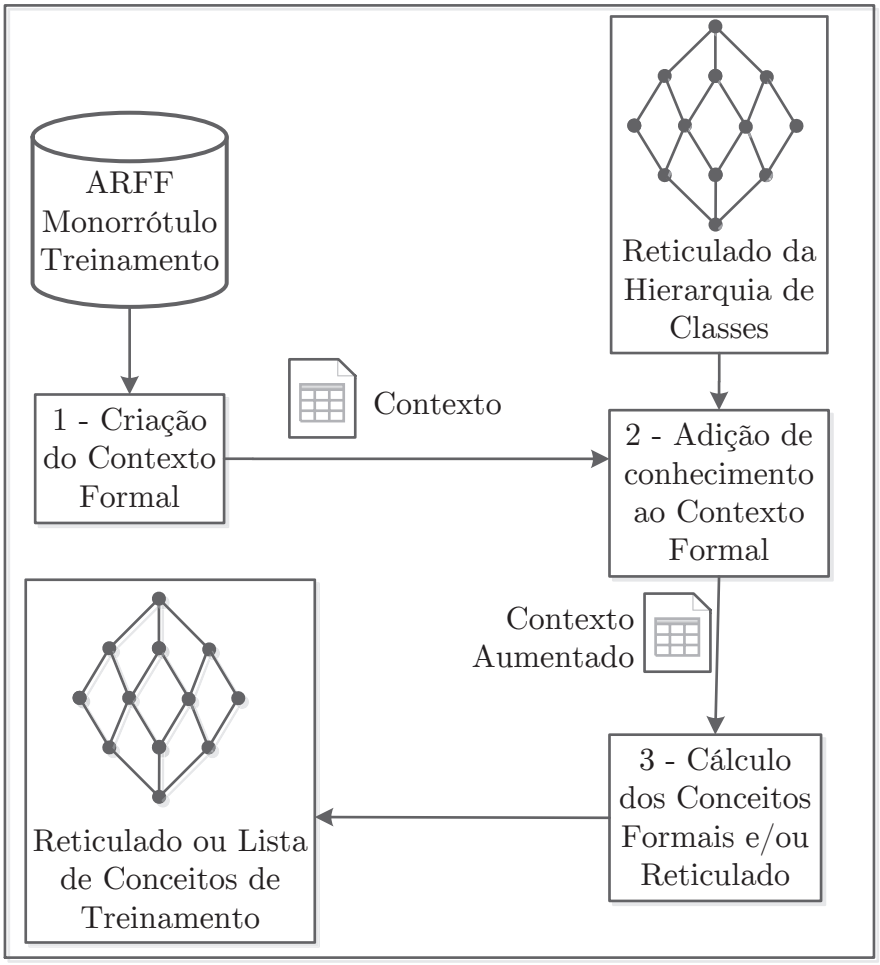


Figura 4.2: Operações realizadas na etapa de treinamento do modelo proposto.

A Figura 4.2 detalha a etapa de treinamento em que se tem como entradas a base de dados de treinamento ARFF plana e o reticulado que representa à hierarquia de classes obtidos na etapa de pré-processamento. A saída desta etapa é o reticulado ou lista de conceitos de treinamento calculado a partir do contexto formal criado com as instâncias de treinamento e a adição de conhecimento referente a hierarquia das classes. No modelo proposto não será feito o cálculo do reticulado com o estabelecimento de uma relação de ordem entre os conceitos por uma questão de economia de recursos computacionais. Todo o modelo proposto utiliza apenas a lista dos conceitos formais e o reticulado será utilizado no contexto deste trabalho apenas para facilitar a compreensão dos exemplos.

O Algoritmo 2 representa de forma detalhada a etapa de treinamento, considerando que $DTrain = \{d_1, d_2, ..., d_n\}$ é um conjunto contendo n instâncias de treinamento, e $d = (\{At_d\}, \{Cl_d\})$ uma instância tal que $d \in DTrain$, composta por um par de conjuntos de informações At_d que contêm seus os atributos, Cl_d que contém as suas classes, e TC que é uma taxonomia com a organização hierárquica das classes representada por meio de DAG ou árvore.

A fim de possibilitar o uso do classificador considerando apenas o cálculo de parte dos conceitos formais possíveis em um contexto, foi estabelecido um limiar denominado *maxAtt* nos algoritmos tendo como finalidade definir o número máximo de atributos que um conceito pode possuir para ser incluído na lista final de conceitos. O Algoritmo 3 detalha como o limiar *maxAtt* é utilizado para limitar o número de conceitos calculados mediante o descarte dos conceitos com número de atributos maior que o valor estabelecido.

Algoritmo 2 Etapa de treinamento do classificador.

Input: Um conjunto de instâncias de treinamento $DTrain$; uma taxonomia hierárquica das classes TC; um valor para limiar *maxAtt*;
Output: Um conjunto de conceitos formais CF;

1. Inicializa contexto formal Ctx aumentado para armazenar atributos e classes;
2. **for all** $d \in DTrain$ **do**
3. Adiciona um objeto a contexto Ctx contendo os atributos $\{At_d\}$ de d e as suas classes folhas $\{Cl_d\}$;
4. **for all** $c \in \{Cl_d\}$ **do**
5. Marca as classes intermediárias até o nó raiz da hierarquia de acordo com TC;
6. **end for**
7. **end for**
8. $CF := calculaConceitos(Ctx, maxAtt)$;
9. **return** CF;

Cada conceito encontrado nesta etapa é composto de um par $(extensão, intensão)$, sendo que a intensão, devido ao aumento do contexto formal com as informações das classes, conterá um subconjunto de atributos e um subconjunto de classes. Por meio da intensão do conceito, obtém-se a informação de que existe um grupo de instâncias que possui em comum um conjunto de atributos e pertencem a um conjunto de classes. A extensão fornece a informação a respeito de quais as instâncias da base de treinamento possuem o agrupamento atributo-classes presentes na intensão. A partir da extensão pode-se obter a informação de suporte deste conceito, ou seja, quantas instâncias na base de treinamento são cobertas por este conceito. A Figura 4.3 representa a estrutura interna de um conceito formal obtido na etapa de treinamento; na representação, a intensão armazena de forma binária uma associação entre atributos e classes e a extensão armazena de forma

Algoritmo 3 *calculaConceitos*(*Ctx*, *maxAtt*) Etapa de treinamento do classificador - detalhamento da utilização do limiar *maxAtt*.

Input: Um contexto formal *Ctx* e um valor *maxAtt* determinando o número máximo de atributos aceitos em um conceito formal;
Output: Um conjunto de conceitos formais *C*;

```
1. ...
2. Encontre um conceito C_k usando o contexto Ctx;
3. AtC_k := atributos (não classes) presentes na intensão de C_k;
4. if |AtC_k| <= maxAtt then
5.    C := C ∪ {C_k}
6. end if
7. ...
8. return C;
```

binária quais as instâncias do contexto de treinamento possuem o agrupamento presente na intensão, possibilitando o cálculo do suporte do agrupamento.

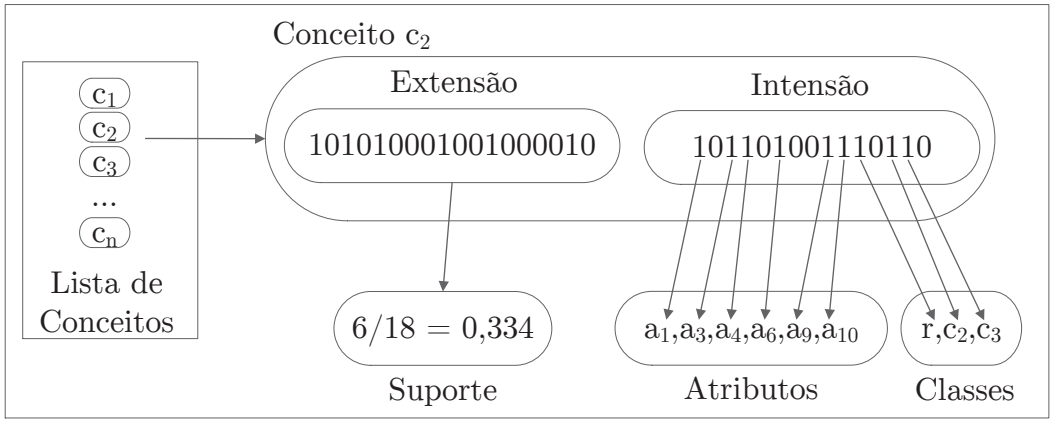


Figura 4.3: Estrutura interna de um conceito formal obtido na etapa de treinamento.

4.2.3 Testes do classificador

A Figura 4.4 representa a etapa de testes. As entradas para a etapa são os conceitos obtidos na etapa de treinamento e a base ARFF hierárquica contendo as instâncias de teste.

Nesta etapa, as instâncias separadas para o teste do classificador serão submetidas a ele a fim de predizer as suas classes. A instância a ser classificada é comparada a cada um dos conceitos formais do reticulado ou lista de conceitos. Em cada comparação é realizado o que se denomina "Teste de Cobertura" que consiste em verificar se a instância possui

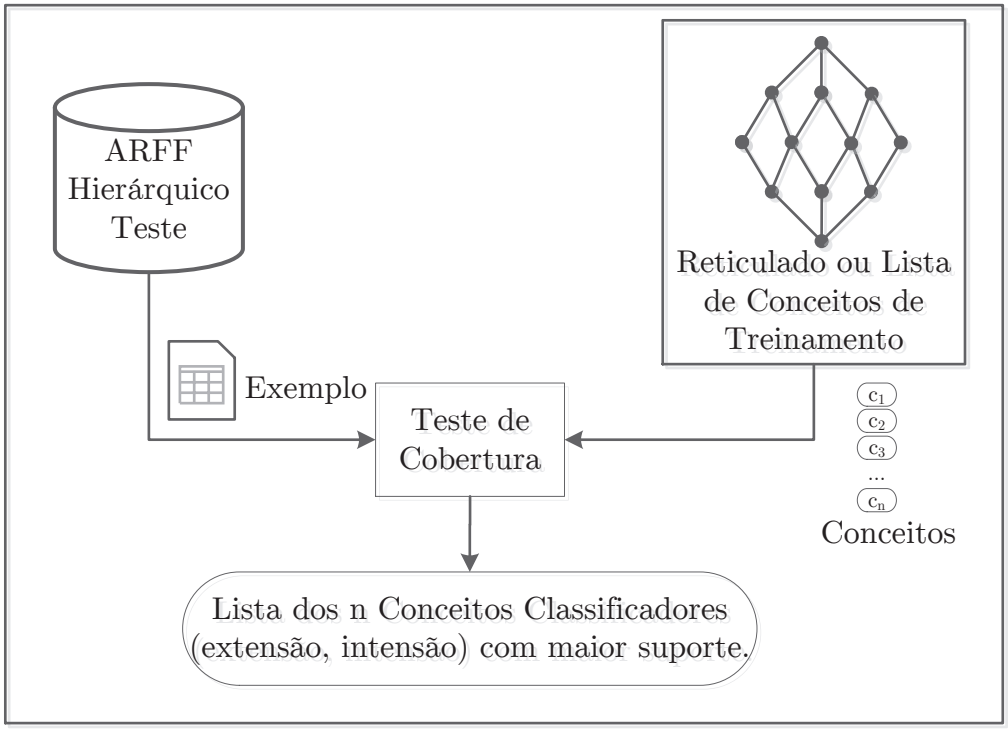


Figura 4.4: Operações realizadas na etapa de teste do modelo proposto.

todos os atributos do conceito formal que não representam informações sobre classes; em caso positivo, este conceito é selecionado para posteriormente ser utilizado para classificar a instância.

Como um exemplo em classificação pode conter um grande número de conceitos que o cobrem, fez-se necessário estabelecer um número máximo de conceitos que serão selecionados para classificar este exemplo e uma métrica para determinar os critérios para seleção dos melhores conceitos dentre os que cobrem o exemplo sendo classificado. Para limitar a quantidade de conceitos escolhidos para cada instância, estabeleceu-se um limiar denominado *nConcepts* e a métrica usada para comparar e escolher os melhores conceitos será o valor do seu suporte.

Ao final das comparações entre instâncias em classificação e conceitos formais obtidos na etapa de treinamento, cada instância terá uma lista de conceitos para sua classificação, e cada conceito selecionado terá uma lista de classes (em função da ampliação do contexto formal e adição do conhecimento sobre a taxonomia de classes realizada na etapa de treinamento), a qual conterá as classes preditas para a instância em classificação. A Figura 4.5 representa a estrutura da saída desta etapa e na sequência será usado um algoritmo para mostrar de forma mais detalhada como são realizados os testes.

Seja $DTest = \{d_1, d_2, ..., d_n\}$ um conjunto contendo n instâncias de teste, e seja $d = (\{At_d\}, \{Cl_d\})$ uma instância tal que $\{d \in DTest\}$, composta por dois subconjuntos de informações $\{At_d\}$ que contem seus os atributos e $\{Cl_d\}$ que contêm as suas classes, e $C = \{c_1, c_2, ..., c_n\}$ uma lista de conceitos formais obtidos na etapa de treinamento do classificador, e seja $c_i = (int_i, ext_i)$ composto de um par de valores representando sua intensão e sua extensão, o Algoritmo 4 representa a etapa de testes.

Algoritmo 4 Etapa de testes do classificador.

Input: Um conjunto de conceitos formais CF obtidos na etapa de treinamento (Algoritmo 2); Um parâmetro $nConcepts$ estabelecendo o número de conceitos que o algoritmo deverá selecionar para cada instância em classificação;

Output: Um conjunto de Resultados R contendo uma lista de conceitos selecionados para cada instância em classificação;

```
1.  R := ∅
2.  for all d ∈ DTrain do
3.     selcf = ∅;
4.     for all cf ∈ CF do
5.        if cobre(cf, d) then
6.           if |selcf| <= nConcepts then
7.              selcf := selcf ∪ {cf};
8.           else
9.              if (suporte(cf) > menor suporte em selcf) then
10.                selcf := selcf \ {conceito de menor suporte em selcf};
11.                selcf := selcf ∪ {cf};
12.             end if
13.          end if
14.       end if
15.    end for
16.    R := R ∪ {(d, selcf)};
17. end for
18. return R;
```

A função $cobre(cf, d)$ utilizada no Algoritmo 4 verifica se a instância d em classificação possui todos os atributos presentes na intensão de cada conceito formal; em caso afirmativo, diz-se que o conceito cf cobre a instância d que está sendo classificada.

A saída da etapa de teste representada pela variável R no Algoritmo 4 conterá uma lista de conceitos selecionados para classificar cada instância, e cada lista será ordenada em ordem decrescente de suporte dos conceitos. A Figura 4.5 representa a organização da saída da etapa de treinamento.

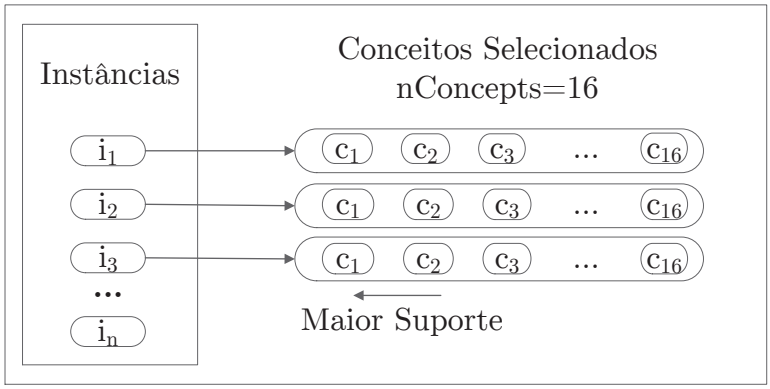


Figura 4.5: Representação da saída obtida na etapa de testes.

4.2.4 *Apuração dos resultados do classificador*

Nesta etapa, os dados referentes aos testes com a classificação das instâncias serão analisados a fim de estabelecer as medidas de desempenho do classificador. As medidas propostas para este modelo são a hP, hR e hF propostas por Kiritchenko *et al.* (2005) e descritas na Equação 3.7, Equação 3.8 e Equação 3.9 respectivamente, detalhadas na Seção 3.2.6.

A proposta inicial do classificador era produzir um único resultado representado por um par de valores para precisão e revocação; no entanto, após esta implementação inicial ele foi adaptado para possibilitar a obtenção de múltiplos resultados mediante o uso de um limiar que possibilita ao usuário obter resultados com maior precisão ou maior revocação.

Após a etapa de testes e a obtenção do conjunto de resultados R, as classes existentes nos conceitos selecionados para cada instância são a elas atribuídas e os resultados são totalizados para análise do desempenho do classificador. O limiar estabelecido pelo usuário indica quantos resultados (pares de valor para precisão e revocação) serão obtidos.

Para exemplificar, suponha-se que em um experimento foi utilizado o parâmetro $nConcepts = 16$, conforme o exemplo da Figura 4.5. A apuração de resultados iniciará com $limiar = 1$ e cada instância será classificada com as classes existentes no conceito formal que estiver na posição 1 da sua lista de conceitos, os resultados são comparados com as classes verdadeiras das instâncias e o primeiro par de valores para precisão e revocação é obtido. Na sequência, o limiar é incrementado passando para $limiar = 2$ e uma nova apuração é feita, sendo que agora as classes atribuídas às instâncias serão as classes que estiverem nas posições 1 e 2 de cada lista, e um novo par de resultados é apurado. Usando $limiar = 3$ serão utilizados os conceitos das posições 1,2 e 3 para cada instância e assim sucessiva-

mente possibilitando a obtenção de n pares de valores para precisão e revocação, no qual $n = limiar = nConcepts$.

Esta heurística proposta faz com que nos limiares iniciais os conceitos com maior suporte e por consequência menor número de atributos/classes sejam utilizados para classificar as instâncias, levando com maior precisão, mas com valores baixos para revocação. Conforme o limiar vai aumentando, um número maior de conceitos é usado cumulativamente para determinar as classes de cada instância, aumentando a revocação e diminuindo a precisão. Estes resultados podem então ser interpolados e representados por meio de uma curva PR (descrita na Seção 3.2) e para se obter um resultado que represente o desempenho do classificador considerando o conjunto de resultados obtidos, a área sob esta curva é calculada. O método proposto para representar os resultados é similar ao utilizado no Clus-HMC, descrito em Vens *et al.* (2008), o que facilitará a comparação dos resultados.

4.3 Demonstração do Modelo Proposto

Nesta seção, demonstrar-se-á de forma simplificada o método proposto para classificação por meio de um exemplo utilizando uma pequena base de dados hipotética a fim de facilitar a compreensão e a visualização dos dados.

Para o exemplo, considerar-se-á a existência de uma taxonomia hierárquica de classes representada por uma DAG, com um total de 6 classes (5 classes e a classe raiz) conforme a Figura 4.6.

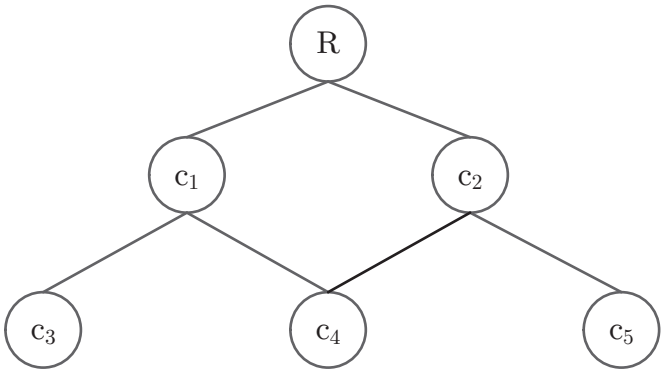


Figura 4.6: DAG representando a hierarquia das classes para o exemplo.

O conjunto de dados para o exemplo é composto de 10 instâncias com características que se enquadram no paradigma de classificação hierárquica multirrótulo, representado na

Tabela 4.2. É importante notar que de acordo com a taxonomia proposta, uma instância que pertence à classe c_3, como é o caso da instância 3, também pertence à classe c1 pelo fato de que a classe c_3 é uma classe filha da classe c_1. Nos dados de entrada as instâncias estão relacionadas apenas com as suas classes de maior profundidade na hierarquia. Outro ponto importante dos dados de exemplo é a instância 6, que está associada a duas classes (c_3 e c_5) que não possuem nenhum ancestral em comum exceto a classe raiz na hierarquia proposta, o que caracteriza um problema de classificação hierárquica multirrótulo. Na instância 4 tem-se uma instância que pertence à classe c_4 a qual possui dois nós pais na hierarquia (c_1 e c_2), o que caracteriza um problema de classificação com uma taxonomia representada em forma de DAG.

Tabela 4.2: Conjunto de dados para a demonstração do classificador.

id	a_1	a_2	a_3	a_4	a_5	Classes
1	5	0	1	1	1	c_1
2	1	5	3	2	0	c_2
3	5	1	5	3	3	c_3
4	5	4	1	5	3	c_4
5	2	5	0	2	2	c_5
6	4	4	4	0	4	$c_3@c_5$
7	5	4	4	1	4	$c_2@c_3$
8	3	4	1	5	2	c_2
9	1	5	1	4	5	c_4
10	5	3	4	2	5	$c_3@c_5$

O conjunto de dados foi particionado em dois subconjuntos, o primeiro com 7 instâncias será utilizado para treinamento do classificador e está representado na Tabela 4.3. O segundo subconjunto composto por 3 instâncias será usado para os testes e está representado na Tabela 4.4.

Tabela 4.3: Dados para a etapa de treinamento com os atributos contínuos.

id	a_1	a_2	a_3	a_4	a_5	Classes
1	5	0	1	1	1	c_1
2	1	5	3	2	0	c_2
3	5	1	5	3	3	c_3
4	5	4	1	5	3	c_4
5	2	5	0	2	2	c_5
6	4	4	4	0	4	$c_3@c_5$
7	5	4	4	1	3	$c_2@c_3$

O próximo passo é a transformação dos dados do formato hierárquico para plano a fim de possibilitar a discretização dos atributos por meio da ferramenta WEKA. O processo

Tabela 4.4: Dados para etapa de testes com os atributos contínuos.

id	a_1	a_2	a_3	a_4	a_5	Classes
8	3	4	1	5	2	c_2
9	1	5	3	4	5	c_4
10	5	3	4	2	5	c_3@c_5

é realizado para as partições de treinamento e teste conforme descrito demonstrado no Algoritmo 1 e representado na Tabela 4.1.

Com os dados representados pelo do formato plano, a discretização é realizada. A fim de que os mesmos parâmetros de discretização sejam aplicados sobre ambas as partições de treinamento e teste, as partições foram unidas em uma única partição para este processo. Após a discretização dos atributos todas as informações passam a ter uma representação discreta, conforme demonstra a Tabela 4.5. Note-se que cada atributo contínuo foi desdobrado em 3 atributos discretos e binários, por exemplo, o atributo a_1 foi representado pelos atributos a_{11}, a_{12} e a_{13} discretos. A discretização dos dados é uma etapa necessária para o modelo proposto, pois a FCA não suporta atributos contínuos em seus contextos.

Outra etapa também já representada na Tabela 4.5 é o aumento do contexto formal realizado mediante a adição das informações referentes à taxonomia das classes. Como no conjunto de dados de entrada têm-se apenas as informações referentes às classes folhas com as quais as instâncias estão associadas, o processo envolve alguns tratamentos dos dados, tais como: (i) representar computacionalmente a hierarquia das classes - isto pode ser feito com a criação da matriz de adjacências da árvore ou DAG da taxonomia, sendo então necessário ainda encontrar os fechamentos transitivos na matriz que podem ser obtidos pelo do algoritmo de Warshall (Cormen *et al.*, 2001); (ii) representar todas as classes como atributos adicionais no contexto formal e (iii) para cada instância/objeto do contexto, adicionar as informações referentes às classes folhas e também os fechamentos transitivos.

Os dados da Tabela 4.5 estão agora no formato de um contexto formal, possibilitando o uso dos algoritmos de FCA para o cálculo dos conceitos formais e reticulado. A Figura 4.7 demonstra o reticulado gerado a partir do contexto formal da Tabela 4.5, sendo que o atributo *id* foi desconsiderado e também apenas as instâncias com *id* de 1 a 7 escolhidas para treinamento foram utilizadas. Na representação, a intensão de cada conceito formal está descrita na primeira linha de cada caixa de texto que acompanha os conceitos formais, e a extensão na segunda linha está representada por um único valor numérico que é a contagem de exemplos que possuem o agrupamento presente na intensão; dividindo o valor da extensão pelo número total de exemplos usados para treinamento obter-se-á o

Tabela 4.5: Dados com os atributos discretizados e com adição das informações referentes a taxonomia das classes. As instâncias de 1 a 7 serão utilizadas para treinamento e de 8-10 para testes.

id	a_{11}	a_{12}	a_{13}	a_{21}	a_{22}	a_{23}	a_{31}	a_{32}	a_{33}	a_{41}	a_{42}	a_{43}	a_{51}	a_{52}	a_{53}	r	c_1	c_2	c_3	c_4	c_5
1	0	0	1	1	0	0	1	0	0	1	0	0	1	0	0	1	1	0	0	0	0
2	1	0	0	0	0	1	0	1	0	0	1	0	1	0	0	1	0	1	0	0	0
3	0	0	1	1	0	0	0	0	1	0	1	0	0	1	0	1	1	0	1	0	0
4	0	0	1	0	0	1	1	0	0	0	0	1	0	1	0	1	1	1	0	1	0
5	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	1	0	1	0	0	1
6	0	0	1	0	0	1	0	0	1	1	0	0	0	0	1	1	1	1	1	0	1
7	0	0	1	0	0	1	0	0	1	1	0	0	0	1	0	1	1	1	1	0	0
8	0	1	0	0	0	1	1	0	0	0	0	1	0	1	0	1	0	1	0	0	0
9	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1	1	1	1	0	1	0
10	0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	1	1	1	1	0	1

suporte de cada conceito formal.

O reticulado contém os conceitos formais que serão utilizados para classificar novas instâncias ainda não apresentadas ao classificador na etapa de treinamento. Na representação do reticulado estão todos os conceitos possíveis para o contexto formal, no entanto para se utilizar um conceito para predizer as classes de um instância este conceito deverá gerar predições que tenham alguma significância, assim alguns conceitos do reticulado serão excluídos da etapa de treinamento por não apresentarem informações relevantes. No exemplo, o conceito que representa o topo do reticulado $(\{r\}, 7)$, alguns outros conceitos dos primeiros níveis como $(\{a_{24}, r\}, 3)$, $(\{a_{51}, r\}, 2)$, $(\{a_{52}, r\}, 3)$, $(\{a_{31}, r\}, 3)$, $(\{a_{24}, a_{52}, r\}, 3)$ não têm utilidade no contexto de predição hierárquica, pois sempre associarão a instância em classificação com a classe raiz da taxonomia.

Após obtenção dos conceitos formais e o reticulado na etapa de treinamento, passar-se-á à etapa de testes na qual submeter-se-ão ao classificador instâncias não utilizadas no processo de treinamento a fim de obterm-se as predições das classes para estas instâncias. Esta etapa acontece conforme definido no Algoritmo 4.

Analisem-se os testes que serão feitos sobre a instância $id = 8$ a qual possui na Tabela 4.5 o conjunto de atributos $\{a_{12}, a_{23}, a_{31}, a_{43}, a_{52}\}$. Ao procurarem-se nos conceitos do reticulado da Figura 4.7 os conceitos que cobrem este exemplo encontrar-se-ão os conceitos $(\{a_{23}, c_2, r\}, 5)$ e $(\{a_{23}, c_2, c_5, r\}, 2)$ - considerando apenas os conceitos que possuem classes diferentes da classe raiz. O processo é realizado para cada instância a ser classificada e em experimento com bases de dados reais, o número de conceitos que cobrirão um exemplo pode ser bastante alto, sendo necessário estabelecer um limite de conceitos que serão selecionados (parâmetro *nConcepts* no Algoritmo 4), bem como ao se escolher um novo conceito deve ser verificado se não existem outros conceitos de maior suporte já

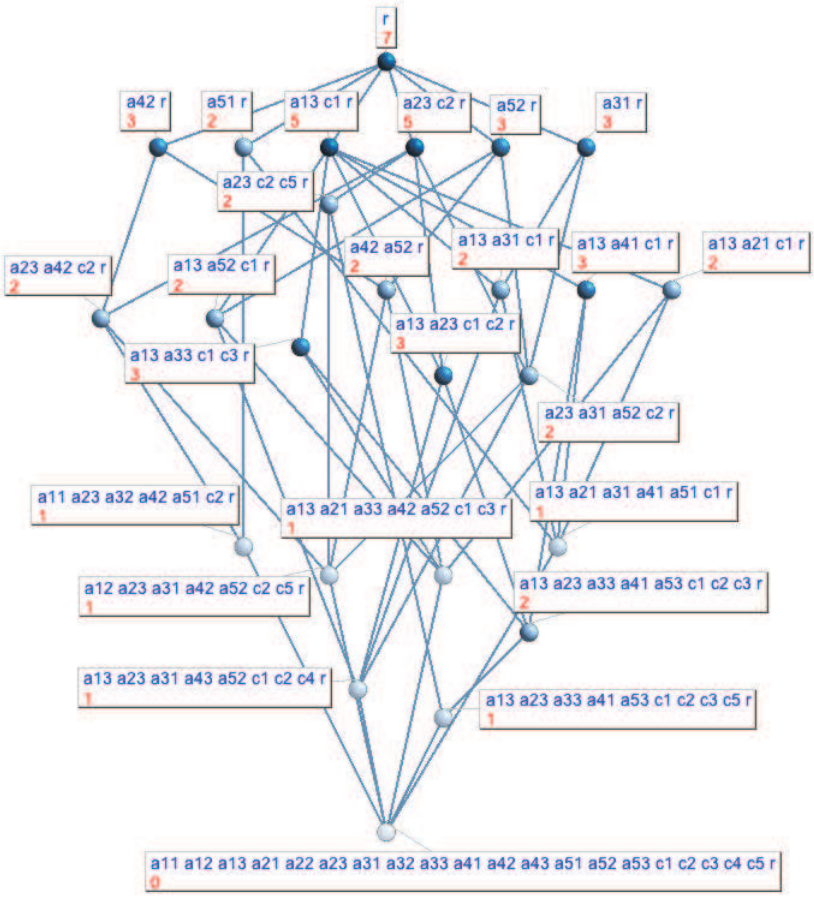


Figura 4.7: Reticulado (de treinamento) contendo os conceitos que serão utilizados para classificar novos exemplos.

selecionados associados às mesmas classes.

Ao final da etapa de testes, ter-se-ão os conceitos formais selecionados para cada instância a ser classificada. A Tabela 4.6 apresenta os conceitos selecionados para as instâncias da partição de testes. Neste exemplo, para facilitar a compreensão, foram selecionados apenas 2 conceitos para cada exemplo ($nConcepts = 2$).

Tabela 4.6: Conceitos selecionados para as instâncias da partição de teste.

id	Conceito Selecionado Limiar 1	Conceito Selecionado Limiar 2
8	($\{a_{23}$,c_2,r$\}$,5)	($\{a_{23}$,c_2,c_5,r$\}$,2)
9	($\{a_{13}$,c_1,r$\}$,5)	($\{a_{13}$,c_3,c_1,r$\}$,2)
10	($\{a_{13}$,c_1,r$\}$,5)	($\{a_{13}$,c_3,c_1,r$\}$,2)

Após a conclusão da etapa de testes, a última tarefa a ser realizada é a apuração dos resultados obtidos. Neste processo, os conceitos selecionados para classificar uma instância

são ordenados de acordo com o seu suporte a fim de que os conceitos com maior suporte sejam utilizados nos limiares iniciais obtendo uma maior precisão nas predições. A Tabela 4.7 demonstra a apuração dos acertos no limiar 1. Para se obter os resultados no primeiro limiar (*limiar* = 1), todas as instâncias da partição de testes são classificadas com as classes que estão no conceito formal de maior suporte (primeiro conceito da lista ordenada de conceitos selecionados para a instância).

Tabela 4.7: Análise dos resultados para o limiar 1. (*)A classe root não é considerada nas contagens dos Verdadeiros Positivos (VP) e Falsos Positivos (FP).

id	Conceitos do limiar 1	Classes Preditas	Classes Verdadeiras	VP*	FP*
8	($\{a_{23},c_2,$r$\}$,5)	$\{c_2,$r$\}$	$\{c_2,$r$\}$	1	0
9	($\{a_{13},c_1,$r$\}$,5)	$\{c_1,$r$\}$	$\{c_4,c_2,c_1,$r$\}$	1	0
10	($\{a_{13},c_1,$r$\}$,5)	$\{c_1,$r$\}$	$\{c_5,c_3,c_2,c_1,$r$\}$	1	0

A Tabela 4.8 representa a apuração dos resultados no segundo limiar (*limiar* = 2), neste limiar as classes preditas para as instâncias serão as classes presentes no conceito selecionado para o limiar 1 mais as classes presentes no conceito selecionado para o limiar 2. Em exemplos reais com um grande número de limiares, as predições em cada limiar também serão cumulativas da mesma forma que neste exemplo.

Tabela 4.8: Análise dos resultados para o limiar 2. No limiar 2 as classes preditas são adicionadas às preditas no limiar 1. (*)A classe root não é considerada nas contagens dos Verdadeiros Positivos (VP) e Falsos Positivos (FP).

id	Conceitos dos limiares 1 e 2	Classes Preditas	Classes Verdadeiras	VP*	FP*
8	($\{a_{23},c_2,$r$\}$,5), ($\{a_{23},c_2,c_5,$r$\}$,2)	$\{c_2,c_5,$r$\}$	$\{c_2,$r$\}$	1	1
9	($\{a_{13},c_1,$r$\}$,5), ($\{a_{13},c_3,c_1,$r$\}$,2)	$\{c_3,c_1,$r$\}$	$\{c_4,c_2,c_1,$r$\}$	1	1
10	($\{a_{13},c_1,$r$\}$,5),($\{a_{13},c_3,c_1,$r$\}$,2)	$\{c_3,c_1,$r$\}$	$\{c_5,c_3,c_2,c_1,$r$\}$	2	0

Como resultado final para o processo classificatório ter-se-ão os pares de valores para precisão e revocação obtidos para cada limiar proposto. A Tabela 4.9 sumariza os resultados obtidos, sendo $\sum VP$ a soma de todos os valores de VP obtidos para as instâncias no limiar, $\sum FP$ a soma de todos os valores de FP obtidos para as instâncias no limiar e $\sum Pos$ a soma do número de classes verdadeiras em cada instância. Em cada limiar, o valor final para precisão é obtido pela Equação 3.7 e o valor final para revocação é obtido com a Equação 3.8, de acordo com a proposta do Kiritchenko *et al.* (2005) detalhada na Seção 3.2.6 deste trabalho.

Após a obtenção dos pares de valores para precisão e revocação referentes aos limiares estabelecidos pode-se representá-los por meio da curva PR e calcular-se a área sob a curva

Tabela 4.9: Resultados finais por limiar.

Limiar	$\sum VP$	$\sum FP$	$\sum Pos$	Precision	Recall
1	3	0	8	1,000	0,375
2	4	2	8	0,666	0,500

a fim de obter uma medida única do desempenho do classificador. A Figura 4.8 apresenta a curva e a medida da área sob a curva para os dados do exemplo demonstrado.

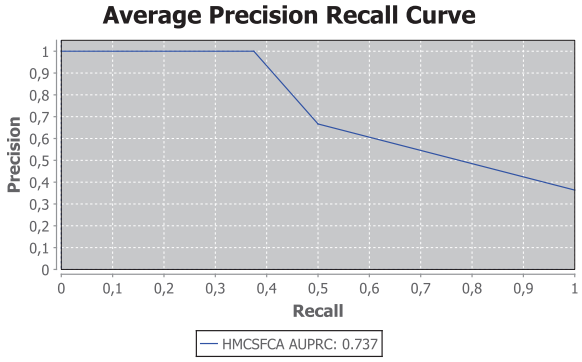


Figura 4.8: Gráfico representando a curva PR e a medida AUPRC para o exemplo.

Os pontos referentes aos extremos são adicionados no processo interpolação para se obter uma curva completa e o método para interpolação utilizado é o proposto por Davis e Goadrich (2006) equivalente ao método utilizado em Vens *et al.* (2008). A AUPRC obtida no exemplo foi de 0,737 sendo 1 o resultado para um classificador com 100% de acerto.

4.4 Considerações Finais

Neste capítulo, apresentou-se o modelo proposto neste trabalho para classificação hierárquica multirrótulo. Por tratar-se de um modelo que emprega técnicas que não são tradicionais dentro do processo de mineração de dados, como por exemplo a FCA, a representação do modelo precisa apresentar um maior nível de detalhamento para facilitar a sua compreensão. Neste sentido, também foi criado um exemplo com um pequeno conjunto de dados hipotéticos para representar as tarefas realizadas em cada etapa do classificador proposto. No próximo capítulo serão apresentados os experimentos realizados e os resultados obtidos.

Capítulo

5

Experimentos e Resultados Obtidos

Neste capítulo serão demonstrados os resultados obtidos mediante o uso do modelo proposto. A Seção 5.1 apresenta as bases de dados selecionadas para os experimentos. A Seção 5.2 mostra os resultados dos experimentos e a Seção 5.3 detalha os testes estatísticos aplicados sobre os resultados. Finalizando o Capítulo a Seção 5.4 apresenta a análise dos resultados apresentados.

5.1 Bases de Dados

Em Vens *et al.* (2008), foram criadas e disponibilizadas publicamente para *download* 24 bases de dados que foram utilizadas no desenvolvimento do Clus. As bases foram organizadas tendo como referência o modelo clássico de organismos biológicos *Saccharomyces cerevisiae* também conhecido como *Yeast*, as quais foram utilizadas previamente em Clare e King (2003). Todas as bases são hierárquicas multirrótulo, ou seja, cada instância é rotulada como pertencente a mais de uma classe em diferentes ramos da taxonomia e as classes estão organizadas em uma estrutura hierárquica na forma de um árvore ou DAG. A Tabela 5.1 apresenta as informações gerais sobre as 24 bases de dados que podem ser obtidas no site do projeto Clus por meio do endereço http://dtai.cs.kuleuven.be/clus/hmcdatasets já divididas em 3 partições: treinamento, validação e testes.

Para realização dos experimentos com o modelo proposto foram selecionadas 8 bases de dados dentre as 24 disponibilizadas pelo projeto do Clus, sendo 4 bases de dados

Tabela 5.1: Informações gerais sobre as 24 bases de dados utilizadas no projeto Clus (Vens *et al.*, 2008).

	FunCat	Gene Ontology
Versão do esquema	2.1 (01/09/2007)	1.2 (04/11/2007)
Anotações Yeast	16/03/2007	07/04/2007
Total classes	1362	22960
Média de classes por base de dados	492 (6 níveis)	3997 (14 níveis)
Média de classes por exemplo	8,8 (3,2 nós folhas)	35 (5 nós folhas)

com taxonomia em DAG (GO): spo_GO, gasch2_GO, cellcycle_GO, church_GO, e as suas respectivas 4 bases de dados com taxonomia em árvore (FunCat): spo_FUN, gasch2_FUN, cellcycle_FUN, church_FUN.

As características das bases de dados selecionadas são apresentadas na Tabela 5.2 e na Tabela 5.3 para as taxonomias em DAG e árvore, respectivamente.

Tabela 5.2: Características das bases de dados com taxonomia em DAG utilizadas para os testes do classificador proposto.

Base de dados	Partição	Amostras	Atributos	Classes na Taxonomia
spo_GO	Treinamento	1597	80	4113
	Validação	837		
	Teste	1263		
gasch2_GO	Treinamento	1636	52	4132
	Validação	849		
	Teste	1288		
cellcycle_GO	Treinamento	1625	77	4119
	Validação	848		
	Teste	1278		
church_GO	Treinamento	1627	27	4119
	Validação	844		
	Teste	1278		

Os conjuntos de dados selecionados podem ser classificados quanto ao tipo de problema de classificação de acordo com a terminologia proposta por Silla e Freitas (2011), abordada na seção 3.2.4 (página 40), por meio da tripla $\langle \Upsilon, \Psi, \Phi \rangle$ representada para cada conjunto de dados na Tabela 5.4.

Uma característica que difere os problemas de classificação hierárquica multirrótulo de

Tabela 5.3: Características das bases de dados com taxonomia em árvore utilizadas para os testes do classificador proposto.

Base de dados	Partição	Amostras	Atributos	Classes na Taxonomia
spo_FUN	Treinamento	1600	80	500
	Validação	837		
	Teste	1266		
gasch2_FUN	Treinamento	1639	52	500
	Validação	849		
	Teste	1291		
cellcycle_FUN	Treinamento	1628	77	500
	Validação	848		
	Teste	1281		
church_FUN	Treinamento	1630	27	500
	Validação	844		
	Teste	1281		

Tabela 5.4: Classificação dos problemas de classificação hierárquica multirrótulo de acordo com a terminologia proposta por Silla e Freitas (2011), abordada na seção 3.2.4.

Base de dados	Υ	Ψ	Φ
spo_GO	D	MPL	PD
gasch2_GO	D	MPL	PD
cellcycle_GO	D	MPL	PD
church_GO	D	MPL	PD
spo_FUN	T	MPL	PD
gasch2_FUN	T	MPL	PD
cellcycle_FUN	T	MPL	PD
church_FUN	T	MPL	PD

outros problemas de classificação com uso de taxonomias de classes é a grande quantidade de rótulos associados a cada exemplo. A Tabela 5.5 apresenta o número médio de rótulos associados a cada instância, sendo que esta contagem foi feita nas partições de testes para cada conjunto de dados.

Para os experimentos foram ignoradas as partições separadas para validação uma vez que o classificador proposto necessita apenas da base de treinamento para construção do classificador e a base de testes para verificar o seu desempenho. A opção pelo descarte desta

Tabela 5.5: Número médio de rótulos associados para os exemplos no conjunto de dados de treinamento.

	Média de classes por exemplo	
Base de Dados	Nós folha da taxonomia	Todos os nós da taxonomia
spo_GO	5,08	25,18
gasch2_GO	5,08	25,15
cellcycle_GO	5,09	25,17
church_GO	4,56	23,02
spo_FUN	3,28	8,93
gasch2_FUN	3,26	8,88
cellcycle_FUN	3,27	8,91
church_FUN	3,87	10,29

partição e o não aproveitamento destes exemplos para treinamento foi feita para facilitar a comparação de resultados, pois a maior parte dos trabalhos em classificação hierárquica utilizam estas bases de dados com o mesmo particionamento nos experimentos.

Os conjuntos de dados selecionados para os experimentos foram pré-processados. Foi realizada a conversão dos conjuntos de dados de hierárquicos para planos, conforme proposto no Algoritmo 1. Posteriormente, foi realizada a substituição dos valores ausentes mediante o uso da ferramenta WEKA aplicando o filtro para pré-processamento de atributos não supervisionado *ReplaceMissingValues* a fim de substituir os valores desconhecidos. Para remoção dos atributos que não agregam informações relevantes ao contexto foi aplicado o filtro *RemoveUseless* de pré-processamento não supervisionado por meio da ferramenta WEKA, utilizando como variância máxima permitida o valor padrão (99%).

Para encerrar o pré-processamento, foi realizada a discretização dos dados pela ferramenta WEKA com o filtro *Discretize* de pré-processamento não supervisionado de atributos, parametrizado para por padrão discretizar os atributos contínuos em 50 atributos discretos, bem como manter a frequência dos atributos equalizada.

Na etapa de treinamento, um contexto formal é criado para cada conjunto de dados utilizando as instâncias da partição de treinamento pré-processadas, na qual cada exemplo do conjunto de dados é mapeado como um objeto do contexto e cada atributo é mapeado como um atributo do contexto, conforme descrito na Seção 4.2.2.

No mapeamento de cada exemplo para objeto do contexto, é realizado o processo de agrupamento dos exemplos do conjunto de dados plano que pertencem ao mesmo exemplo

do conjunto de dados hierárquico, fazendo o processo inverso ao de transformação do conjunto de dados de hierárquico para plano executado na etapa de pré-processamento; assim, o número de objetos do contexto é igual ao número de exemplos do conjunto de dados original. Esta reversão foi realizada, pois a transformação de hierárquica para plana foi motivada pela necessidade de discretizar os dados na etapa de pré-processamento e a maioria dos algoritmos para discretização não suportam como dados de entrada exemplos multirrótulo com taxonomia hierárquica de classes.

Para finalizar a criação do contexto formal a partir do conjunto de dados pré-processados é feito o processo de aumento do contexto no qual as informações sobre a hierarquia das classes são adicionadas como atributos para cada exemplo no contexto, este processo foi detalhado na Seção 2.2.6;

A Tabela 5.6 apresenta as características básicas dos contextos formais criados no início da etapa de treinamento, sendo o número de objetos igual ao número de exemplos do conjunto de dados, o número de atributos igual ao número de atributos do conjunto de dados após a sua discretização acrescido do número de classes presentes na hierarquia, e a densidade do contexto indica a proporção entre a presença e ausência dos atributos no contexto.

Tabela 5.6: Características dos contextos formais gerados na etapa de treinamento do classificador.

	Contexto Formal		
Base de dados	Objetos	Atributos	Densidade
spo_GO	1.597	7.491	0,014
gasch2_GO	1.636	6.725	0,011
cellcycle_GO	1.625	7.969	0,013
church_GO	1.118	4.850	0,010
spo_FUN	1.600	3.881	0,023
gasch2_FUN	1.639	3.100	0,019
cellcycle_FUN	1.628	4.350	0,019
church_FUN	1.337	1.127	0,031

O cálculo dos conceitos na etapa de treinamento, conforme proposto neste trabalho, tem como requisito a necessidade de se fazer uma poda no computo deles mediante a análise do número de atributos presentes em sua intensão, e como os algoritmos existentes para cálculo dos conceitos não apresentam esta funcionalidade, foi desenvolvido um algoritmo para o cálculo dos conceitos formais com esta característica.

Para cada um dos 8 conjuntos de dados selecionados foram realizados 5 experimentos com valores distintos para o limiar *maxAtt* = $\{4, 6, 8, 12, \infty\}$ que estabelece o número mínimo de atributos não classes que um conceito deve conter na sua intensão para que seja selecionado. A Tabela 5.7 apresenta o número de conceitos formais obtidos para cada conjunto de dados nos diferentes limiares propostos.

Tabela 5.7: Número total de conceitos formais obtidos em cada experimento realizado.

	Limiar maxAtt				
Base de dados	4	6	8	12	∞
spo_GO	233.422	734.452	1.539.344	2.858.650	3.002.653
gasch2_GO	1.127.505	1.127.762	1.127.770	1.127.770	1.127.770
cellcycle_GO	2.157.077	2.774.977	3.891.234	5.169.850	5.210.024
church_GO	1.238.420	1.980.071	2.292.633	2.455.218	2.467.384
spo_FUN	7.228	27.789	102.828	448.953	569.504
gasch2_FUN	310.552	310.570	310.575	310.575	310.575
cellcycle_FUN	513.924	533.224	546.251	551.551	551.699
church_FUN	30.723	130.859	332.834	549.215	617.038

Após o cálculo dos conceitos, a etapa de testes foi realizada, conforme proposto no Algoritmo 4, e os resultados obtidos são apresentados na Seção 5.2 a seguir.

5.2 Resultados Obtidos

A Tabela 5.8 apresenta os resultados obtidos em cada experimento do HMCS-FCA com os diferentes valores para o limiar *maxAtt* e os resultados obtidos utilizando o Clus-HMC. Os valores apresentados representam a AUPRC obtida para cada experimento.

A Tabela 5.9 sumariza os resultados apresentado o melhor resultado obtido pelo HMCS-FCA e o melhor resultado entre o HMCS-FCA e o Clus-HMC. Nos casos em que houve empate entre os resultados do HMCS-FCA, foi escolhido o resultado com o menor *maxAtt*, considerando que quanto menor o limiar, menor será o custo computacional para se obter o resultado. Neste comparativo pode-se observar que o melhor desempenho do HMCS-FCA se deu nos valores mais baixos do limiar *maxAtt*.

Os gráficos apresentados nas Figuras 5.1a, 5.1b, 5.1c e 5.1d mostram o comparativo entre as curvas PR do melhor resultado obtido pelo HMCS-FCA e o resultado do Clus-HMC para as bases de dados com taxonomia representada por meio de DAG. Os gráficos

Tabela 5.8: Resultados obtidos nos experimentos. Os valores representam a AUPRC.

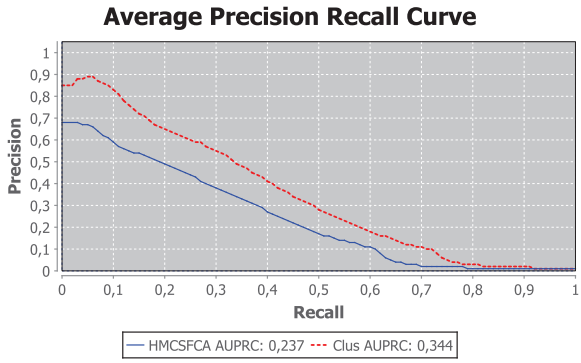
	HMCS-FCA com Limiar maxAtt					Clus-HMC
Base de dados	4	6	8	12	∞	
spo_GO	0,237	0,232	0,232	0,232	0,232	0,354
gasch2_GO	0,248	0,249	0,249	0,249	0,249	0,362
cellcycle_GO	0,240	0,241	0,241	0,240	0,240	0,331
church_GO	0,246	0,246	0,246	0,246	0,246	0,342
spo_FUN	0,155	0,151	0,150	0,152	0,152	0,181
gasch2_FUN	0,160	0,160	0,160	0,160	0,160	0,185
cellcycle_FUN	0,159	0,159	0,159	0,159	0,159	0,173
church_FUN	0,171	0,170	0,169	0,169	0,169	0,168

Tabela 5.9: Comparação entre os valores obtidos para AUPRC para os classificadores HMCS-FCA e Clus-HMC. O símbolo $\bigoplus$ indica o melhor resultado obtido pelo HMCS-FCA nos diversos limiares escolhidos para *maxAtt* e o símbolo $\bigotimes$ indica o melhor resultado entre os dois classificadores.

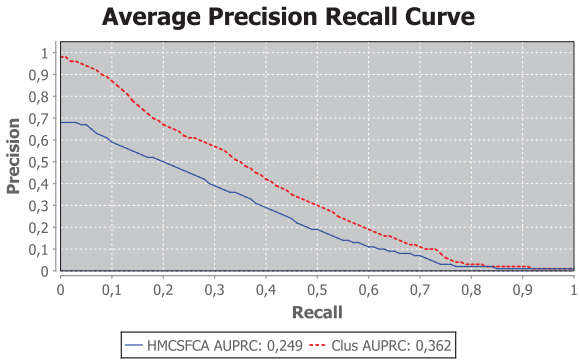
	HMCS-FCA com Limiar maxAtt					Clus-HMC
Base de dados	4	6	8	12	∞	
spo_GO	$\bigoplus$					$\bigotimes$
gasch2_GO		$\bigoplus$				$\bigotimes$
cellcycle_GO		$\bigoplus$				$\bigotimes$
church_GO	$\bigoplus$					$\bigotimes$
spo_FUN	$\bigoplus$					$\bigotimes$
gasch2_FUN	$\bigoplus$					$\bigotimes$
cellcycle_FUN	$\bigoplus$					$\bigotimes$
church_FUN	$\bigoplus$ $\bigotimes$					

apresentados nas Figuras 5.1e, 5.1f, 5.1g e 5.1h, mostram o comparativo entre as Curvas Precision-Recall do melhor resultado obtido pelo HMCS-FCA e o resultado do Clus-HMC para as bases de dados com taxonomia representada por árvores. As curvas referentes aos experimentos nos quais foi obtido um valor menor para AUC não apresentam diferenças significativas quanto a sua distribuição.

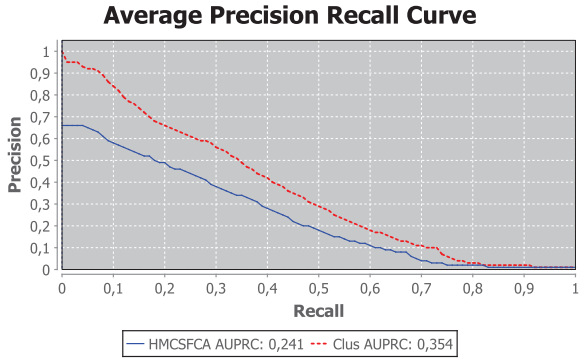
No Apêndice A todas as curvas obtidas são apresentadas agrupadas por base de dados ao longo dos diferentes valores do limiar *maxAtt*.



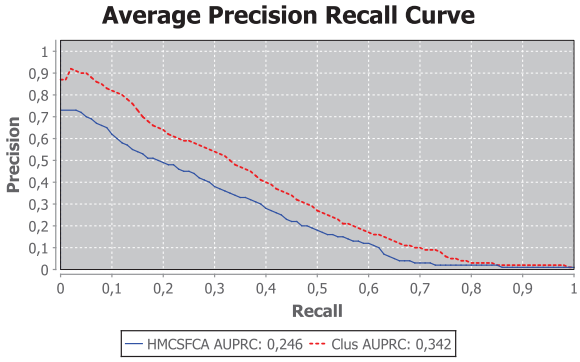
(a) Base de dados spo_GO *maxAtt* = 4.



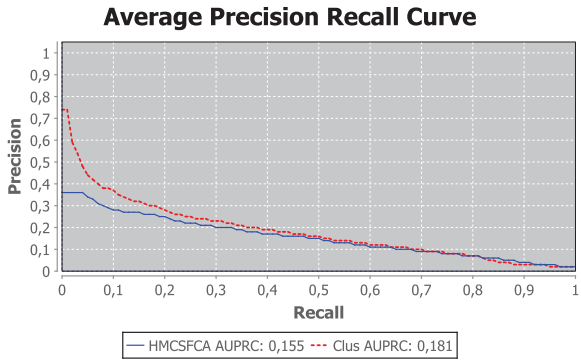
(b) Base de dados gasch2_GO *maxAtt* = 6.



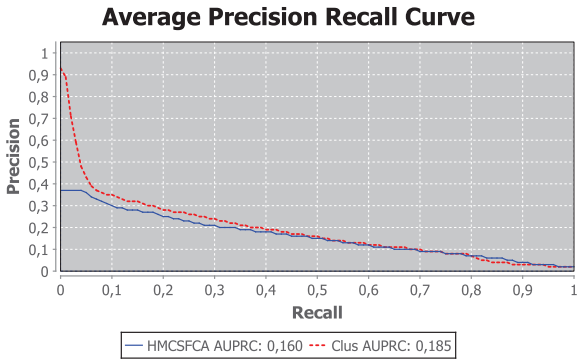
(c) Base de dados cellcycle_GO *maxAtt* = 6.



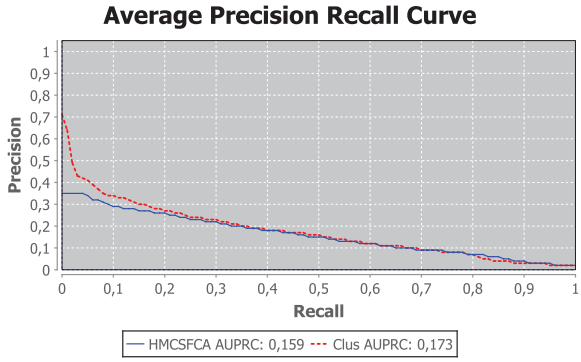
(d) Base de dados church_GO *maxAtt* = 4.



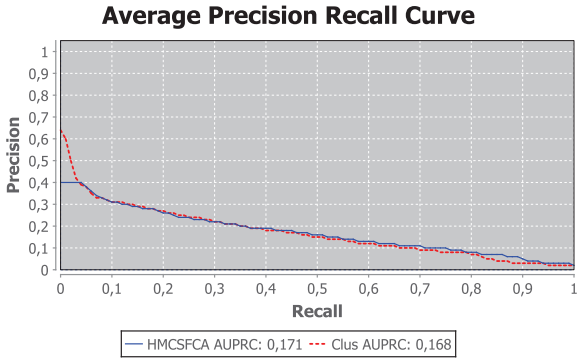
(e) Base de dados spo_FUN *maxAtt* = 4.



(f) Base de dados gasch2_FUN *maxAtt* = 4.



(g) Base de dados cellcycle_FUN *maxAtt* = 4.



(h) Base de dados church_FUN *maxAtt* = 4.

Figura 5.1: Curvas PR contendo o melhor resultado em cada base de dados selecionada.

5.3 Testes Estatísticos

Após obtidos os resultados dos experimentos, um teste estatístico foi realizado para comparar os resultados do HMCS-FCA com os resultados do Clus-HMC. A fim de comparar os resultados obtidos em cada um dos 40 experimentos realizados foi selecionado o teste estatístico não paramétrico de Mann Whitney (Mann e Whitney, 1947), também conhecido como teste U e em algumas literaturas (Montegomery e Runger, 2009), descrito como teste da soma dos postos de Wilcoxon (Wilcoxon, 1945).

O teste U avalia quando a mediana de dois exemplos de dados é a mesma. A hipótese nula é de que as medianas dos dois exemplos são as mesmas e o número de elementos presentes nos dois exemplos (n1 e n2) não precisa ser o mesmo. No teste U, resultados com nível de significância 0,05 podem também ser obtidos para n1 e n2 iguais a 10 ou menores tornando o teste mais fácil de usar do que ferramentas que exigem uma grande quantidade de elementos presentes nos exemplos para se obter um mínimo útil de significância (Eberhart e Shi, 2007).

Para os testes foram selecionados conjuntos contendo 15 pontos extraídos de intervalos prefixados ao longo da curva Precision-Recall para cada classificador e comparados ao pares usando em cada ponto as medidas hP, hR e hF. Dessa forma, para cada experimento foram realizados seis testes representados por:

- t_1: $hP_{HMCS-FCA}$ x $hP_{Clus-HMC}$ obtendo os pontos em intervalos fixos no eixo y (Precision) do gráfico;
- t_2: $hR_{HMCS-FCA}$ x $hR_{Clus-HMC}$ obtendo os pontos em intervalos fixos no eixo y (Precision) do gráfico;
- t_3: $hF_{HMCS-FCA}$ x $hF_{Clus-HMC}$ obtendo os pontos em intervalos fixos no eixo y (Precision) do gráfico;
- t_4: $hP_{HMCS-FCA}$ x $hP_{Clus-HMC}$ obtendo os pontos em intervalos fixos no eixo x (Recall) do gráfico;
- t_5: $hR_{HMCS-FCA}$ x $hR_{Clus-HMC}$ obtendo os pontos em intervalos fixos no eixo x (Recall) do gráfico;
- t_6: $hF_{HMCS-FCA}$ x $hF_{Clus-HMC}$ obtendo os pontos em intervalos fixos no eixo x (Recall) do gráfico.

A Tabela 5.10 apresenta os resultados dos testes estatísticos t_1, t_2, t_3 realizados comparando os resultados obtidos entre o HMCS-FCA e o Clus-HMC usando o teste U. Para obtenção dos valores críticos foi utilizado o nível de significância de 5% ($\alpha = 0,05$). Na grande maioria das comparações, o resultado obtido foi em favor da hipótese nula $H_0 : \mu_1 = \mu_2$ indicando que os resultados possuem a mesma mediana.

Para os testes t_4, t_5, t_6, o resultado apontou em todos os casos que a hipótese nula $H_0 : \mu_1 = \mu_2$, que indica que os resultados possuem a mesma mediana, deve ser aceita em todos os os casos.

Tabela 5.10: Resultados dos testes estatísticos t_1, t_2, t_3 sobre os resultados obtidos. O valor em branco indica que a hipótese nula $H_0 : \mu_1 = \mu_2$ deve ser aceita, já o valor R indica que a hipótese nula deve ser rejeitada e a hipótese alternativa $H_1 : \mu_1 \neq \mu_2$ deve ser aceita.

	Limiar maxAtt														
	4			6			8			12			∞		
Base de dados	t_1	t_2	t_3	t_1	t_2	t_3	t_1	t_2	t_3	t_1	t_2	t_3	t_1	t_2	t_3
spo_GO															
gasch2_GO			R			R			R			R			R
cellcycle_GO															
church_GO															
spo_FUN	R			R			R			R			R		
gasch2_FUN															
cellcycle_FUN	R			R			R			R			R		
church_FUN															

5.4 Análise dos Resultados Obtidos

Os resultados obtidos nos experimentos realizados com o HMCS-FCA permitem as seguintes conclusões:

- O classificador proposto não apresentou um comportamento padrão quanto ao número de conceitos formais encontrados em relação a limiar *maxAtt* estabelecido (Tabela 5.7). Em algumas bases o número de conceitos formais encontrados para com *maxAtt* = 4 foi baixo e apresentou crescimento na medida em que o limiar foi aumentando, mas em outras o número de conceitos encontrados para *maxAtt* = 4 ficou muito próximo do número de conceitos encontrados no cálculo de todos os conceitos (*maxAtt* = ∞). Embora este comportamento tenha sido mais acentuado nas

bases de dados com taxonomia em árvore, em que o número de conceitos tende a ser menor em função do menor número de classes e arestas na taxonomia. Pode-se perceber que este comportamento está ligado ao fato de que o número de possíveis conceitos formais em um contexto está diretamente relacionado à natureza dos dados, e dependerá diretamente das combinações existentes entre objetos e atributos deste contexto. Face ao exposto, o limiar na forma como foi estabelecido *maxAtt* não está sendo eficaz para reduzir o número de conceitos e o tamanho/complexidade do modelo;

- Os resultados obtidos para a AUPRC (Tabela 5.8) não apresentaram melhorias significativas em função do aumento do limiar *maxAtt* e em alguns casos apresentaram um pequeno decréscimo. Este resultado deu-se em função da utilização do suporte de cada conceito formal como métrica utilizada para ordenar os conceitos que serão utilizados para classificar cada instância. Como estão sendo selecionados os conceitos de maior suporte que cobrem a instância, é natural que quanto menor o número de atributos estabelecido pelo limiar *maxAtt* em um conceito, maior seja o seu suporte. Assim, pode-se concluir que empregando apenas o suporte como métrica para a escolha dos conceitos que serão usados para classificar uma instância, não se terão melhorias significativas nos resultados ao se incrementar o limiar *maxAtt*. Este comportamento provavelmente não será o mesmo se uma métrica diferente for adotada;

- Na comparação dos resultados com o Clus-HMC, o HMCS-FCA apresentou resultados competitivos tanto para bases de dados com taxonomia em DAG quanto para as com taxonomia em árvore. É possível observar na comparação das curvas nas Figuras 5.1a, 5.1b, 5.1c, 5.1d, 5.1e, 5.1f, 5.1g e 5.1h, que o Clus-HMC apresenta melhores resultados principalmente quando uma maior precisão é requerida. Embora a diferença seja mais acentuada nas bases de dados com taxonomia em DAG, ela também está presente nas bases de dados com taxonomia em árvore. Uma diferença nas características internas dos classificadores pode explicar esta diferença. O Clus-HMC não faz predição para todas as instâncias da partição de testes ao longo de toda a curva de resultados, sendo que nos resultados de maior precisão, o classificador atribui classes apenas para um número reduzido de instâncias com base em um fator de certeza em relação às suas predições. O HMCS-FCA tem um comportamento diferente pois sempre atribuirá classes a todas as instâncias em cada ponto obtido ao longo da curva. Esta diferença de comportamento faz com que o HMCS-FCA ao predizer classes para todas as instâncias obtenha uma quantidade maior de falsos positivos (FP) que afeta diretamente o resultado da precisão. Embora do ponto de

vista de avaliação do resultado final de um classificador o mais importante é a sua capacidade de acerto, pois na prática não existe vantagem em predizer classes para todas as instâncias uma vez que as predições possuem uma baixa taxa de acerto, é uma diferença que precisa ser considerada na comparação dos dois classificadores em questão sob a óptica de se propor melhorias no classificador proposto;

- Na comparação estatística dos resultados com o Clus-HMC, o HMCS-FCA, apesar de o teste proposto ser bastante flexível; de maneira geral, não foram encontradas diferenças significativas do ponto de vista estatístico entre os resultados comparados.

5.5 Considerações finais

Este capítulo apresentou os resultados dos experimentos realizados com o modelo proposto neste trabalho. Foram realizados experimentos com 4 bases de dados da Ontologia Gênica e e 4 bases de dados da FunCat. Os resultados foram comparados ao Clus-MHC por meio da medida AUPRC e também com o teste U aplicado aos pares sobre os resultados dos dois classificadores em cada base de dados.

Capítulo

6

Conclusão

Neste trabalho, foi apresentado o HMCS-FCA, um modelo para classificação hierárquica multirrótulo baseado em FCA. O modelo usa a abordagem de classificação global suportando taxonomias de classes organizadas em árvore ou DAG.

Nos problemas de classificação hierárquica multirrótulo, as classes estão organizadas em uma taxonomia hierárquica por meio de uma relação de ordem do tipo "É-UM", podendo uma classe ser superclasse ou subclasse de outra. Se a taxonomia for uma DAG a complexidade aumenta pelo fato de que uma subclasse pode conter mais do que uma superclasse. Além desta complexidade criada pela presença da taxonomia hierárquica de classes, ainda é preciso observar que uma instância pode estar associada a mais do que uma classe (em ramo diferentes) na hierarquia por se tratar de um problema de classificação hierárquica multirrótulo.

Assim, é fato que os problemas de classificação hierárquica apresentam um grau de complexidade maior quando comparados à classificação plana e têm sido objetos de muitos estudos e pesquisas e a maioria dos trabalhos estão voltados para problemas de classificação com taxonomia de classes organizadas em árvore e utilizando outras abordagens de classificação diferentes da abordagem global. Conforme a revisão teórica realizada neste trabalho, a abordagem de classificação global tem por vantagens a preservação das restrições naturais nas associações entre as classes e o fato de levar em consideração a hierarquia completa das classes durante as fases de treinamento e testes.

6.1 Contribuições do Trabalho

Como principais contribuições apresentadas pelo presente trabalho, pode-se elencar:

- O desenvolvimento de um modelo para classificação hierárquica multirrótulo utilizando técnicas de FCA é considerado a maior contribuição do presente trabalho. As técnicas da FCA permitem a representação e a manipulação de informações hierarquicamente organizadas, e ainda não haviam sido empregadas neste tipo de problema. Os resultados obtidos no modelo proposto são promissores e abrem novas frentes de pesquisa;

- A proposta de representação dos dados provenientes de bases de dados hierárquicas multirrótulo para um contexto formal da FCA, utilizando a técnica de adição de conhecimento ao contexto formal para adicionar as informações sobre as classes de cada instância, repeitando a ordem determinada pela taxonomia hierárquica das classes, é uma contribuição importante deste trabalho e que pode servir de referência para outras pesquisas utilizando a FCA para classificação hierárquica multirrótulo;

- A capacidade de abstração do modelo proposto quanto ao formato da taxonomia hierárquica de classes utilizada. A organização das informações em um contexto formal possibilita a criação de uma representação das informações capaz de representar problemas de classificação hierárquica com taxonomia em árvore ou DAG sem necessitar de tratamentos diferenciados de acordo com a taxonomia. No modelo apresentado, exceto na etapa de pré-processamento em que para a entrada de dados faz-se necessário diferenciar os conjuntos de dados de acordo com sua taxonomia de classes. Em todas as outras etapas os algoritmos empregados no processo de classificação são os mesmos independente da taxonomia de classes usada no conjunto de dados;

- O modelo implementado é uma contribuição deste estudo uma vez que, exceto as funcionalidades de pré-processamento importadas da ferramenta WEKA, todo o modelo foi programado neste trabalho. O algoritmo para cálculo dos conceitos formais permitindo o uso dos limiares também é uma contribuição deste trabalho que será melhor explorada futuramente no contexto da FCA;

- Em menor escala, a revisão teórica e bibliográfica em relação às duas áreas interseccionadas neste trabalho: FCA e classificação hierárquica multirrótulo, poderá ser utilizada para novas pesquisas e trabalhos relacionados aos assuntos.

6.2 Trabalhos Futuros

Como sugestão de continuidade, poderá ser investigado o uso de outras heurísticas para selecionar os melhores conceitos que serão utilizados para classificar um exemplo na etapa de testes. Neste trabalho, os critérios de seleção e ordenação foram a cobertura e o suporte do conceito respectivamente; no entanto outras métricas podem ser combinadas, pois os conceitos gerados têm em sua intensão uma regra do tipo Atributos → Classes. Outras informações a respeito desta regra, como por exemplo a sua confiança, ou medidas e técnicas já conhecidas que empregaram o uso de regras de associação para classificação, podem ser utilizadas para melhor selecionar os conceitos na etapa de testes.

De acordo com os resultados obtidos neste trabalho, o desempenho do classificador não apresentou melhoria considerável ao se aumentar o limiar *maxAtt*, que por consequência leva a um aumento do número de conceitos formais calculados. Isto demonstra que para os conjuntos de dados utilizados e considerando a cobertura e o suporte como critérios de seleção e ordenação respectivamente, os conceitos mais significativos estão mais próximos ao topo do reticulado. Usando este mesmo modelo, um critério de parada para o limiar *maxAtt* poderá ser definido mediante o uso da partição de validação da base de dados, tornando o modelo independente de um limiar estabelecido estaticamente pelo usuário.

Ainda na etapa de testes, uma função de aproximação poderá ser considerada na comparação do exemplo com os conceitos formais; nesta proposta os conceitos formais selecionados foram os conceitos que cobriam cada exemplo - todos os atributos presentes na intensão do conceito estavam presentes no exemplo. Aqui poderão ser estabelecidas outras medidas de similaridade para este processo, sendo inclusive possível se estabelecer pesos diferenciados para os atributos.

Outro ponto a ser explorado é a ordem que o reticulado impõe aos conceitos formais quando calculado. Neste trabalho, por uma questão de simplificação e capacidade de processamento, optou-se por utilizar os conceitos de maneira independente e não calcular o reticulado de conceitos. Novos trabalhos poderão explorar o uso das informações do reticulado para seleção dos conceitos formais que serão utilizados para classificar uma instância. Uma forma de utilizar estas informações pode ser o uso das relações de “pai”/“filho”, existentes entre os conceitos no reticulado, como critério para seleção dos conceitos classificadores.

6.3 Considerações Finais

O HMCS-FCA e os resultados apresentados demonstram que o modelo baseado em FCA pode ser utilizado para classificação hierárquica multirrótulo, e por se tratar de uma abordagem que utiliza uma combinação de técnicas não tradicionalmente empregadas em mineração de dados, abre inúmeras possibilidades de emprego de novas técnicas e heurísticas para melhorar os seus resultados.

Referências Bibliográficas

Aleksovski, D., Kocev, D., e S, D. (2009). Evaluation of distance measures for hierarchical multi-label classification in functional genomics. In *Proceedings of the 1st workshop on learning from multi-label data (MLD)*.

Altschul, S. F., Madden, T. L., Schaffer, A. A., Zhang, J., Zhang, Z., Miller, W., e Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic acids research*, **25**(17), 3389.

Alves, R. T., Delgado, M. R., e Freitas, A. A. (2008). Multi-label hierarchical classification of protein functions with artificial immune systems. In *Proceedings of the 3rd Brazilian symposium on Bioinformatics: Advances in Bioinformatics and Computational Biology*, BSB '08, p. 1–12, Berlin, Heidelberg. Springer-Verlag.

Andrews, S. (2009). In-close, a fast algorithm for computing formal concepts. In *International Conference on Conceptual Structures (ICCS)*.

Ashburner, M., Ball, C. A., Blake, J. A., Botstein, D., Butler, H., Cherry, J. M., Davis, A. P., Dolinski, K., Dwight, S. S., Eppig, J. T., e et al. (2000). Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nature Genetics*, **25**(1), 25–29.

Astikainen, K., Holm, L., Pitkanen, E., Szedmak, S., e Rousu, J. (2008). Towards structured output prediction of enzyme function. *BMC Proceedings*, **2**(Suppl 4), S2.

Belohlavek, R. e Vychodil, V. (2009). Formal concept analysis with background knowledge: attribute priorities. *Trans. Sys. Man Cyber Part C*, **39**, 399–409.

Belohlávek, R. e Vychodil, V. (2010). Background knowledge in formal concept analysis: constraints via closure operators. In *SAC*, p. 1113–1114.

Bi, W. e Kwok, J. (2011). Multi-label classification on tree- and dag-structured hierarchies. In L. Getoor e T. Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, p. 17–24, New York, NY, USA. ACM.

Blockeel, H., Schietgat, L., Struyf, J., Džeroski, S., e Clare, A. (2006). Decision trees for hierarchical multilabel classification: a case study in functional genomics. In *Proceedings of the 10th European conference on Principle and Practice of Knowledge Discovery in Databases*, PKDD'06, p. 18–29, Berlin, Heidelberg. Springer-Verlag.

Cai, L. e Hofmann, T. (2004). Hierarchical document categorization with support vector machines. In *CIKM'04* , p. 78–87.

Cai, L. e Hofmann, T. (2007). Exploiting known taxonomies in learning overlapping concepts. In *Proceedings of the 20th international joint conference on Artifical intelligence*, IJCAI'07, p. 714–719, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Campbell, M., Ferreira, H., e Carlini, C. (2001). *Bioquímica*. Artmed Editora.

Carpineto, C. e Romano, G. (1993). Galois: An order-theoretic approach to conceptual clustering. In *ICML*, p. 33–40.

Carpineto, C. e Romano, G. (1996). A lattice conceptual clustering system and its application to browsing retrieval. *Machine Learning*, **24**(2), 95–122.

Carpineto, C. e Romano, G. (2004). *Formal concept analysis - mathematical foundations-Concept Data Analysis: Theory and Applications*. Wiley.

Cellier, P., Ferré, S., Ridoux, O., e Ducassé, M. (2007). A parameterized algorithm for exploring concept lattices. In *Proceedings of the 5th international conference on Formal concept analysis*, ICFCA'07, p. 114–129, Berlin, Heidelberg. Springer-Verlag.

Cerri, R. (2010). *Técnicas de Classificação Hierárquica Multirrótulo*. Master's thesis, Universidade de São Paulo.

Clare, A. e King, R. D. (2001). Knowledge discovery in multi-label phenotype data. In *In: Lecture Notes in Computer Science*, p. 42–53. Springer.

Clare, A. e King, R. D. (2003). Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, **19**, 42–49.

Cormen, T. H., Stein, C., Rivest, R. L., e Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

Davis, J. e Goadrich, M. (2006). The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, ICML '06, p. 233–240, New York, NY, USA. ACM.

Dekel, O., Keshet, J., e Singer, Y. (2004a). Large Margin Hierarchical Classification. In *Proceedings of the 21st International Conference on Machine Learning (ICML-2004)*.

Dekel, O., Keshet, J., e Singer, Y. (2004b). An online algorithm for hierarchical phoneme classification. In S. Bengio e H. Bourlard, editors, *MLMI*, volume 3361 of *Lecture Notes in Computer Science*, p. 146–158. Springer.

Dimitrovski, I., Kocev, D., Loskovska, S., e Dzeroski, S. (2011). Hierarchical annotation of medical images. *Pattern Recognition*, **44**(10-11), 2436–2449.

Eberhart, R. C. e Shi, Y. (2007). *Computational intelligence - concepts to implementations*. Elsevier.

Freitas, A. e Carvalho, A. C. (2007). A tutorial on hierarchical classification with applications in bioinformatics.

Freitas, A. A. (2002). *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Friedberg, I. (2006). Automated protein function prediction - the genomic challenge. *Briefings in Bioinformatics*, **7**(3), 225–242.

Ganter, B. (1984). Two basic algorithms in concept analysis. FB4–Preprint 831, TH Darmstadt.

Ganter, B. (1999). Attribute exploration with background knowledge. *Theor. Comput. Sci.*, **217**(2), 215–233.

Ganter, B. e Wille, R. (1999). *Formal concept analysis - mathematical foundations*. Springer.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., e Witten, I. H. (2009). The weka data mining software: an update. *SIGKDD Explor. Newsl.*, **11**(1), 10–18.

King, R. D., Wise, P. H., e Clare, A. (2004). Confirmation of data mining based predictions of protein function. *Bioinformatics*, **20**(7), 1110–1118.

Kiritchenko, S., Matwin, S., e Famili, A. F. (2005). Functional annotation of genes using hierarchical text categorization. In *in Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (held at ISMB-05*.

Kiritchenko, S., Matwin, S., Nock, R., e Famili, A. F. (2006). Learning and evaluation in the presence of class hierarchies: Application to text categorization. In *Canadian Conference on AI*, p. 395–406.

Krajca, P. e Vychodil, V. (2009). Distributed algorithm for computing formal concepts using map-reduce framework. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis: Advances in Intelligent Data Analysis VIII*, IDA '09, p. 333–344, Berlin, Heidelberg. Springer-Verlag.

Krajca, P., Outrata, J., e Vychodil, V. (2010). Advances in algorithms based on cbo. In *CLA*, p. 325–337.

Kuznetsov, S. (2001). On computing the size of a lattice and related decision problems. *Order*, **18**, 313–321.

Kuznetsov, S. O. e Obiedkov, S. (2002). Comparing performance of algorithms for generating concept lattices. *Journal of Experimental and Theoretical Artificial Intelligence*, **14**, 189–216.

Labrou, Y. e Finin, T. (1999). Yahoo! as an ontology: using yahoo! categories to describe documents. In *Proceedings of the eighth international conference on Information and knowledge management*, CIKM '99, p. 180–187, New York, NY, USA. ACM.

Lesk, A. M. (2008). *Introdução à Bioinformática*. Artmed, Porto Alegre.

Liquière, M. e Nguifo, E. M. (1990). Legal: un système d'apprentissage de concepts à partir d'exemples. In *Journées Françaises sur l'Apprentissage*.

Mann, H. B. e Whitney, D. R. (1947). On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics*, **18**(1), 50–60.

Mello, L. e Rigden, D. J. (2002). Anotação funcional e computacional de proteínas. *Revista Biotecnologia Ciencia & Desenvolvimento*, p. 64–70.

Montegomery, D. C. e Runger, G. C. (2009). *Estatística Aplicada e Probabilidade para Engenheiros*. LTC, 4nd edition.

Nguifo, E. M. (1994). Galois lattice: A framework for concept learning-design, evaluation and refinement. In *ICTAI*, p. 461–467.

Oosthuizen, G. D. (1988). *The use of a lattice in knowledge processing*. Ph.D. thesis, Glasgow, Scotland, UK, UK. UMI Order No. GAXD-96805.

Otero, F. E. (2010). *New ant colony optimisation algorithms for hierarquical classification of protein functions*. Ph.D. thesis, Universidade de Kent.

Otero, F. E. B., Freitas, A. A., e Johnson, C. G. (2009). A hierarchical classification ant colony algorithm for predicting gene ontology terms. *Evolutionary Computation Machine Learning and Data Mining in Bioinformatics*, p. 68–79.

Otero, F. E. B., Freitas, A. A., e Johnson, C. G. (2010). A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing*, p. 165–181.

Pearson, W. R. e Lipman, D. J. (1988). Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, **85**(8), 2444–2448.

Peng, X. e Choi, B. (2005). Document classifications based on word semantic hierarchies. In *Artificial Intelligence and Applications'05*, p. 362–367.

Poelmans, J., Elzinga, P., Viaene, S., e Dedene, G. (2010). Formal concept analysis in knowledge discovery: a survey. In *Proceedings of the 18th international conference on Conceptual structures: from information to intelligence*, ICCS'10, p. 139–153, Berlin, Heidelberg. Springer-Verlag.

Pugelj, M. e Džeroski, S. (2011). Predicting structured outputs k-nearest neighbours method. In *Proceedings of the 14th international conference on Discovery science*, DS'11, p. 262–276, Berlin, Heidelberg. Springer-Verlag.

Qiu, X., Gao, W., e Huang, X. (2009). Hierarchical multi-class text categorization with global margin maximization. In *Proceedings of the ACL-IJCNLP 2009 Conference*, p. 165–168. Association for Computational Linguistics.

Rocchio, J. (1971). Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System - Experiments in Automatic Document Processing*, p. 313–323. Prentice Hall.

Rousu, J., Saunders, C., Szedmak, S., e Shawe-Taylor, J. (2005). Learning hierarchical multi-category text classification models. In *Proceedings of the 22nd international conference on Machine learning*, ICML '05, p. 744–751, New York, NY, USA. ACM.

Rousu, J., Saunders, C., Szedmak, S., e Shawe-Taylor, J. (2006). Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, **7**(3), 1601–1626.

Ruepp, A. e et al (2004). The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic Acids Res*, **32**(18), 5539–5545.

Sahami, M. (1995). Learning classification rules using lattices (extended abstract). In *ECML*, p. 343–346.

Silla., C. N. e Freitas, A. A. (2009). A global-model naive bayes approach to the hierarchical prediction of protein functions. In *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, ICDM '09, p. 992–997, Washington, DC, USA. IEEE Computer Society.

Silla, Jr., C. N. e Freitas, A. A. (2011). A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, **22**, 31–72.

Silva, J. (2007). *Algoritmos de Classificação baseados em Análise Formal de Conceitos*. Master's thesis, Universidade Federal de Minas Gerais.

Silva, J., Zarate, L., e Vieira, N. (2006). Formal concept analysis for data mining: Theoretical and practical approaches. *International Conference on Engineering of Intelligent Systems*.

Strok, F. e Neznanov, A. (2010). Comparing and analyzing the computational complexity of FCA algorithms. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '10, p. 417–420, New York, NY, USA. ACM.

Sugumaran, V. (2007). *Intelligent Information Technologies: Concepts, Methodologies, Tools and Applications*. IGI Publishing, Hershey, PA, USA.

Sullivan, R. (2012). *Introduction to Data Mining for the Life Sciences*. Humana Press.

Tan, P.-N., Steinbach, M., e Kumar, V. (2009). *Introdução ao Data Mining - Mineração de Dados*. Ciência Moderna, Rio de Janeiro.

Tsoumakas, G. e Katakis, I. (2007). Multi-label classification: An overview. *IJDWM*, **3**(3), 1–13.

Tsoumakas, G., Katakis, I., e Vlahavas, I. P. (2010). Mining multi-label data. In *Data Mining and Knowledge Discovery Handbook*, p. 667–685.

Vens, C., Struyf, J., Schietgat, L., Džeroski, S., e Blockeel, H. (2008). Decision trees for hierarchical multi-label classification. *Mach. Learn.*, **73**, 185–214.

Wang, J., Shen, X., e Pan, W. (2009). On large margin hierarchical classification with multiple paths. *J Am Stat Assoc*, **104**(487), 1213.

Wang, K. e Senqiang, K. W. (2001). Hierarchical classification of real life documents. In *In Proceedings of the 1st SIAM International Conference on Data Mining*.

Wang, K., Zhou, S., e Liew, S. C. (1999). Building hierarchical classifiers using class proximity. In *Proceedings of the 25th International Conference on Very Large Data Bases*, VLDB '99, p. 363–374, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wilcoxon, F. (1945). Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, **1**(6), 80–83.

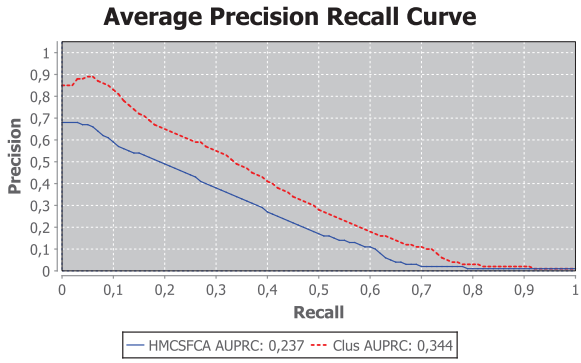
Wille, R. (1982). Restructuring lattice theory: an approach based on hierarchies of concepts. In I. Rival, editor, *Ordered sets*, p. 445–470, Dordrecht–Boston. Reidel.

Wu, F., Zhang, J., e Honavar, V. (2005). Learning classifiers using hierarchically structured class taxonomies. In *SARA*, p. 313–320.

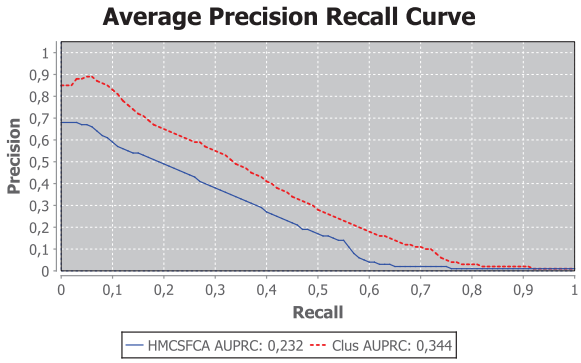
Apêndice

A

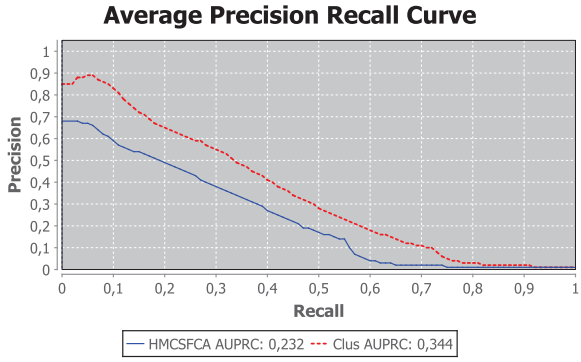
Curvas comparativas: HMCS-FCA e Clus-HMC



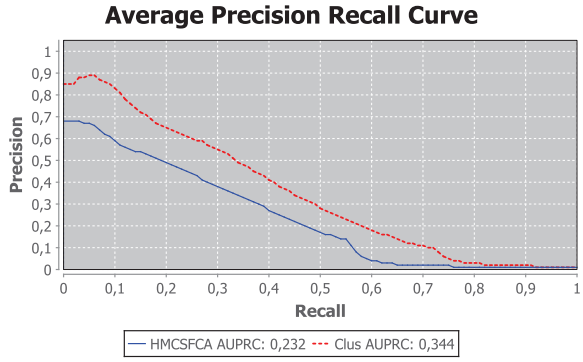
(a) *maxAtt* = 4.



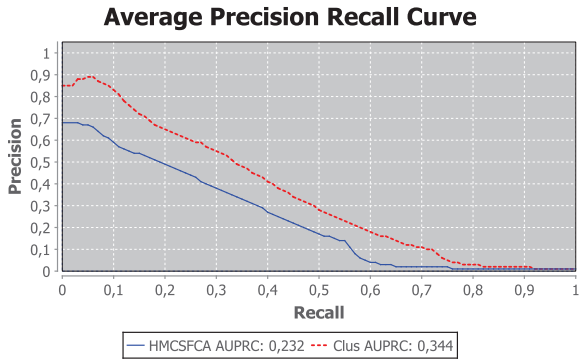
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

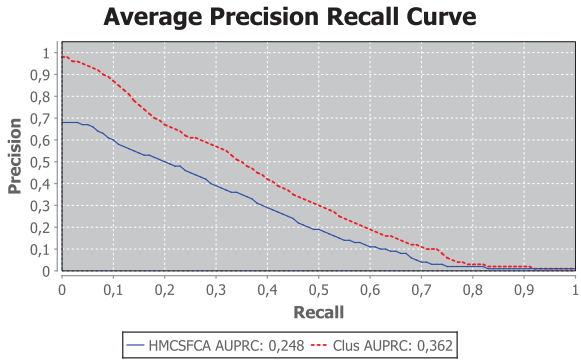


(d) *maxAtt* = 12.

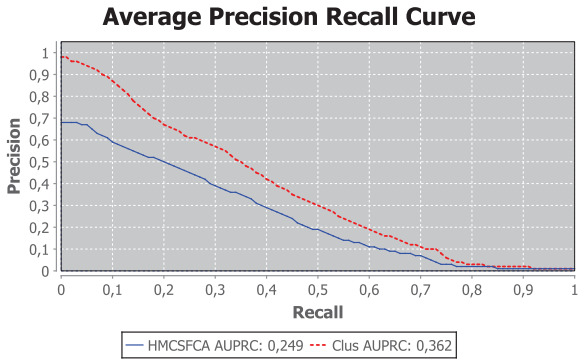


(e) *maxAtt* = ∞.

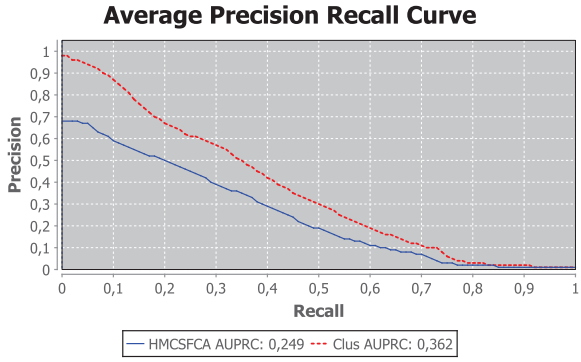
Figura A.1: Curvas PR comparativas para a base de dados spo_GO.



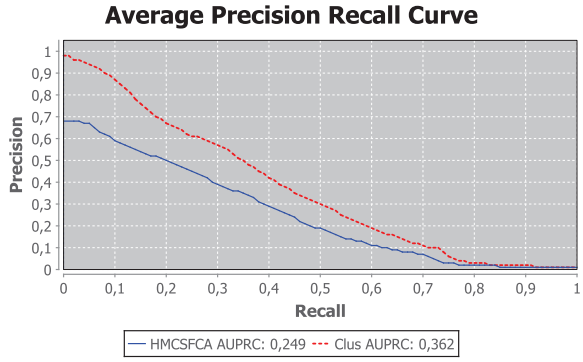
(a) *maxAtt* = 4.



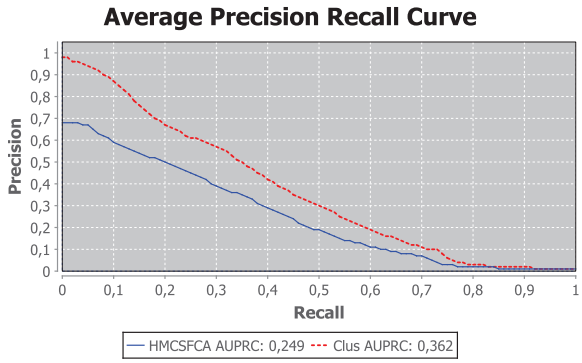
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

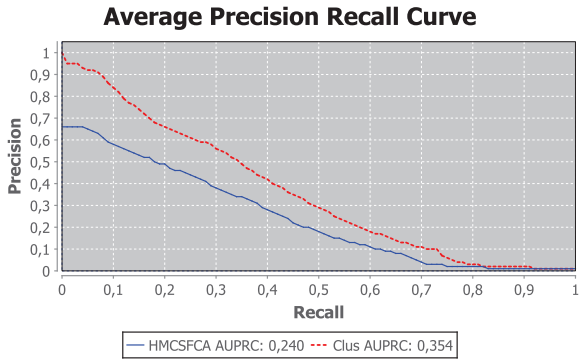


(d) *maxAtt* = 12.

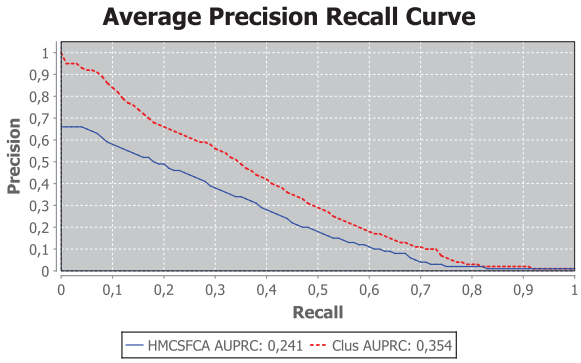


(e) *maxAtt* = ∞.

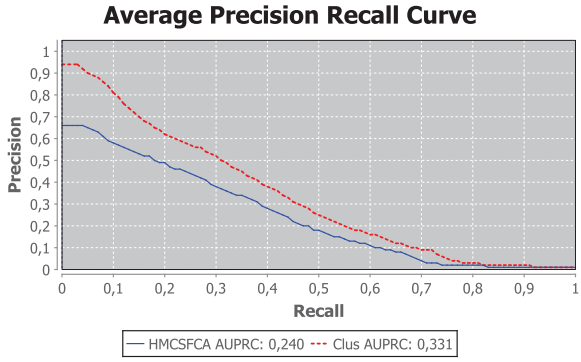
Figura A.2: Curvas PR comparativas para a base de dados gasch2_GO.



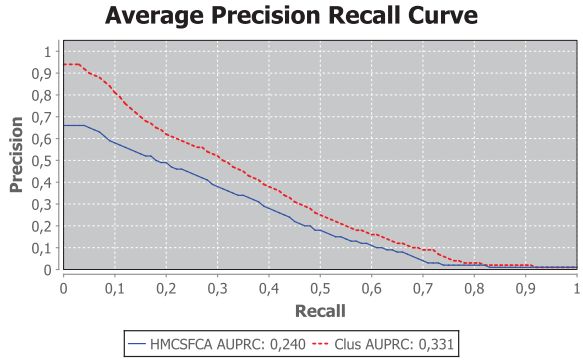
(a) *maxAtt* = 4.



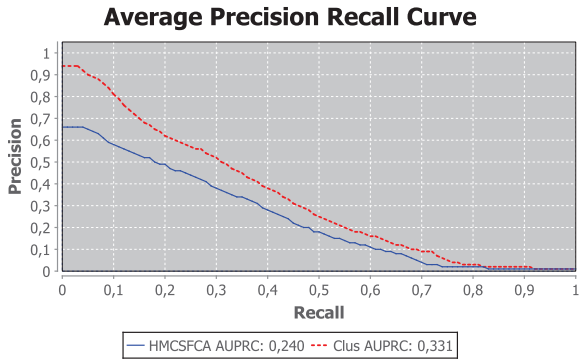
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

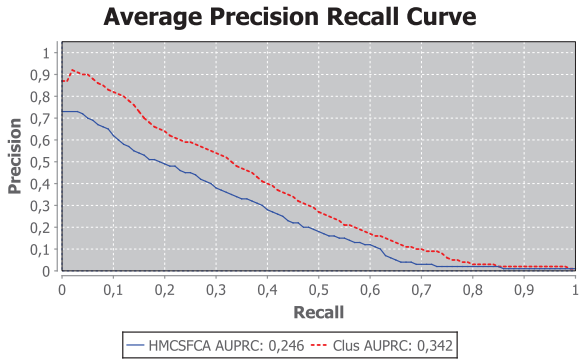


(d) *maxAtt* = 12.

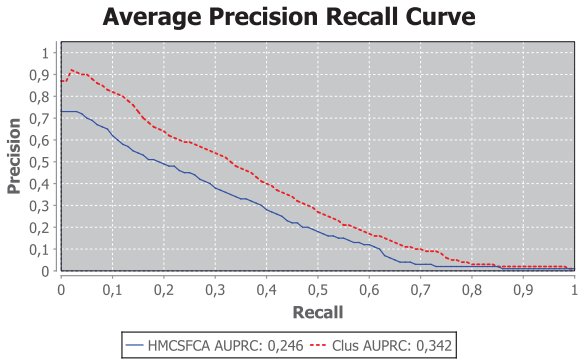


(e) *maxAtt* = ∞.

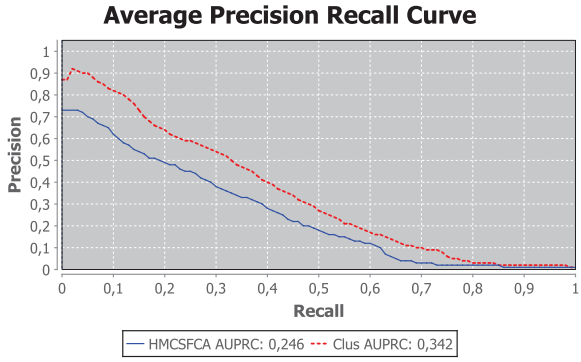
Figura A.3: Curvas PR comparativas para a base de dados cellcycle_GO.



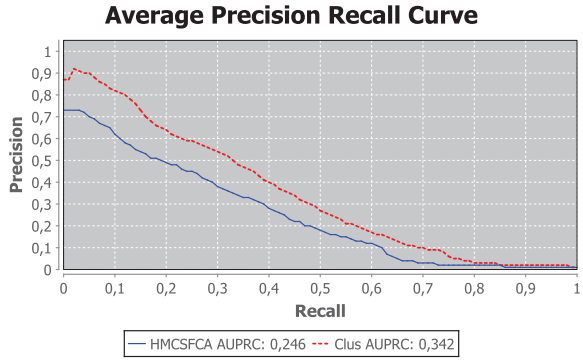
(a) *maxAtt* = 4.



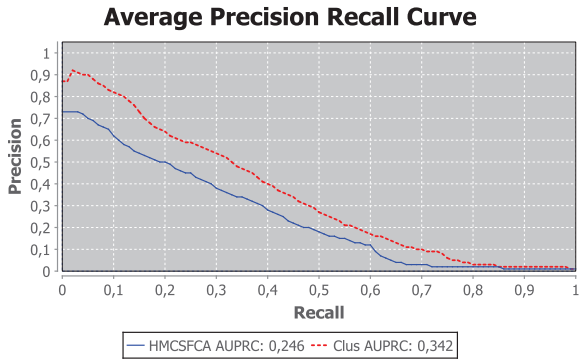
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

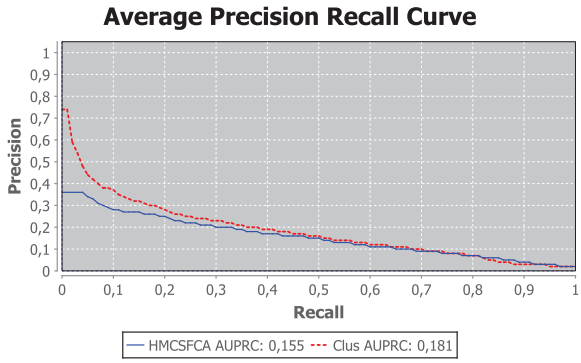


(d) *maxAtt* = 12.

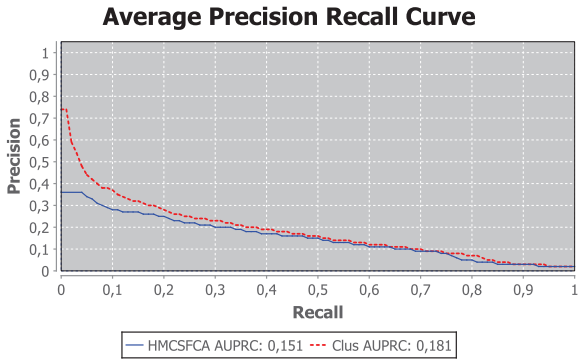


(e) *maxAtt* = ∞.

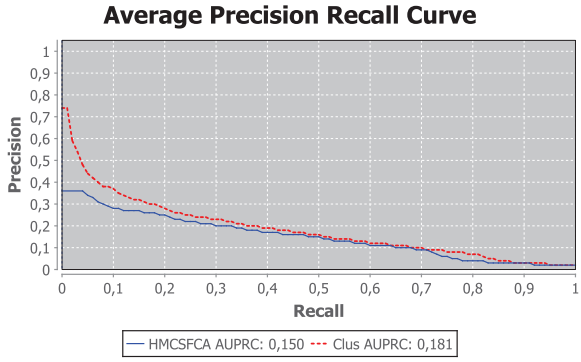
Figura A.4: Curvas PR comparativas para a base de dados church_GO.



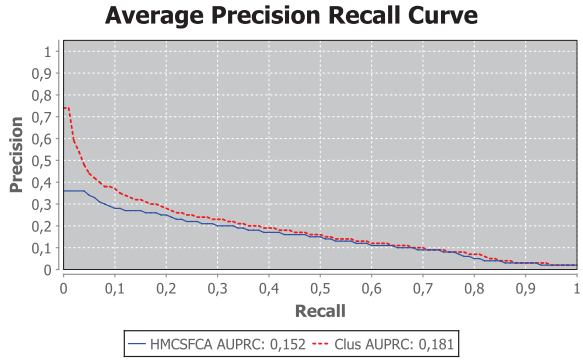
(a) *maxAtt* = 4.



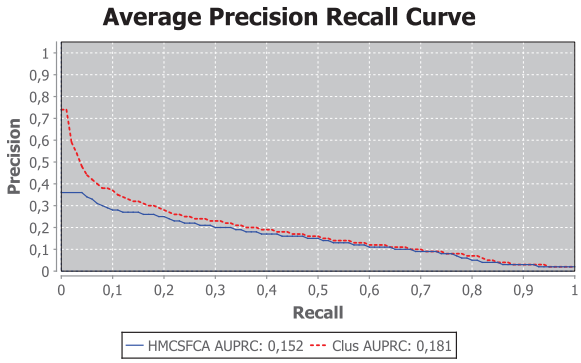
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

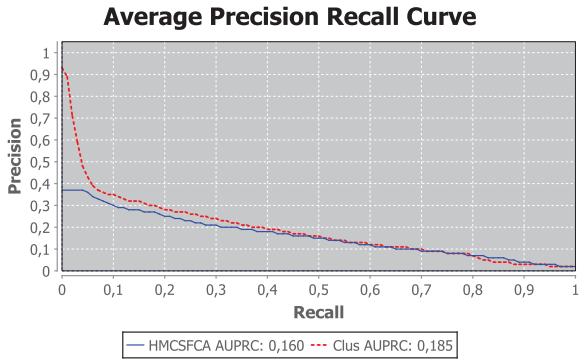


(d) *maxAtt* = 12.

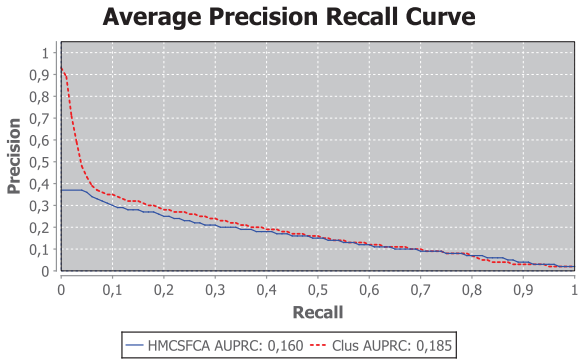


(e) *maxAtt* = ∞.

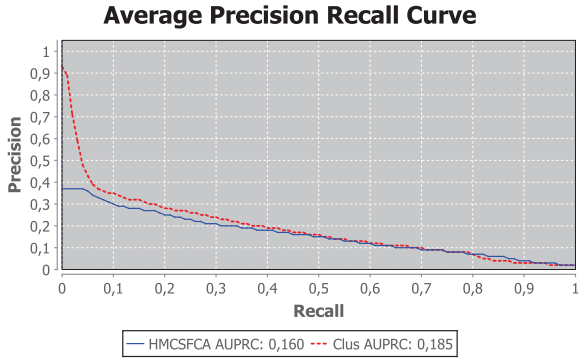
Figura A.5: Curvas PR comparativas para a base de dados spo_FUN.



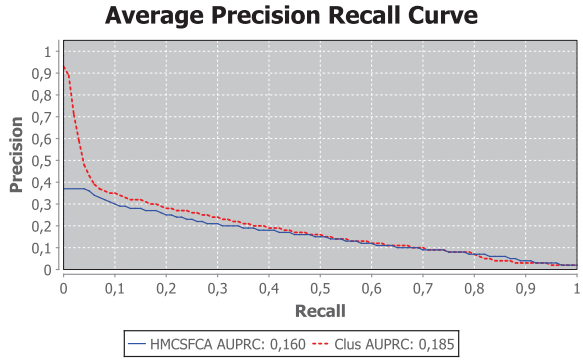
(a) *maxAtt* = 4.



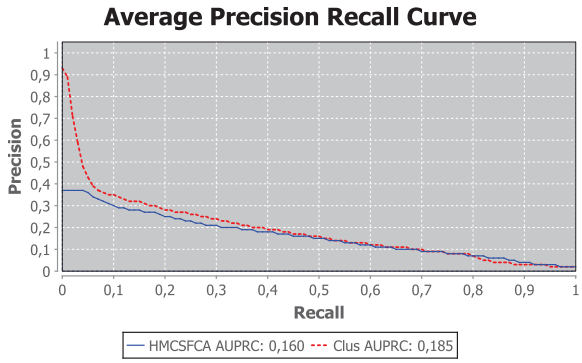
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

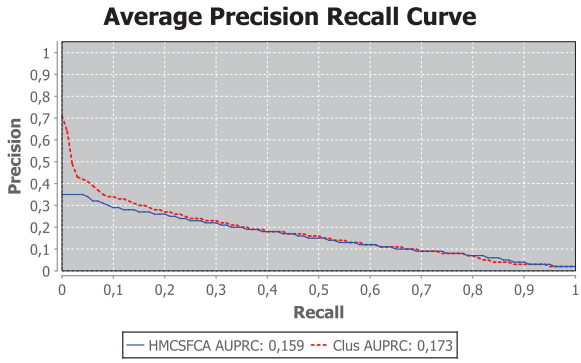


(d) *maxAtt* = 12.

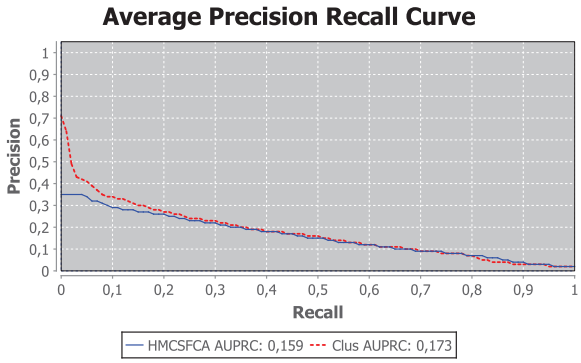


(e) *maxAtt* = ∞.

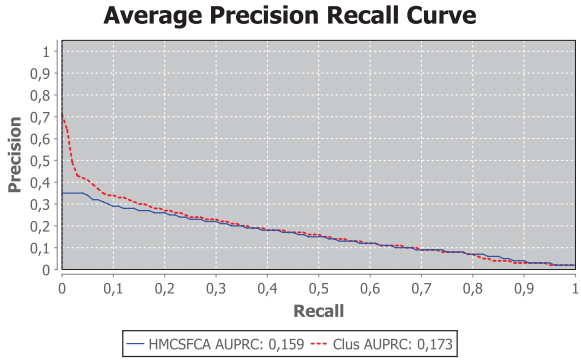
Figura A.6: Curvas PR comparativas para a base de dados gasch2_FUN.



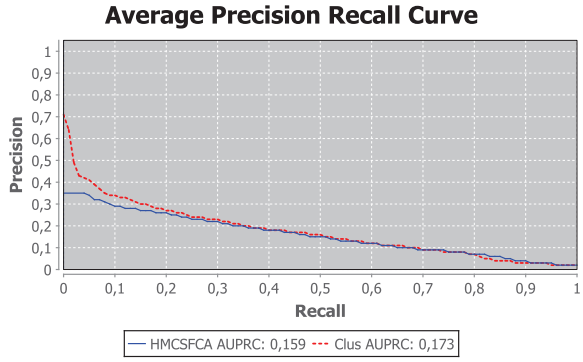
(a) *maxAtt* = 4.



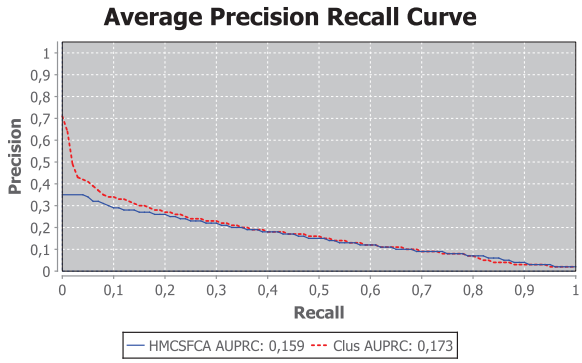
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.

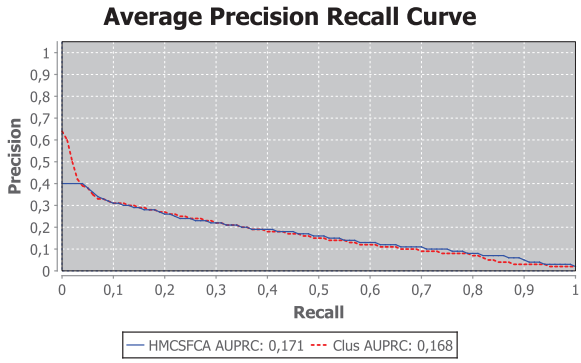


(d) *maxAtt* = 12.

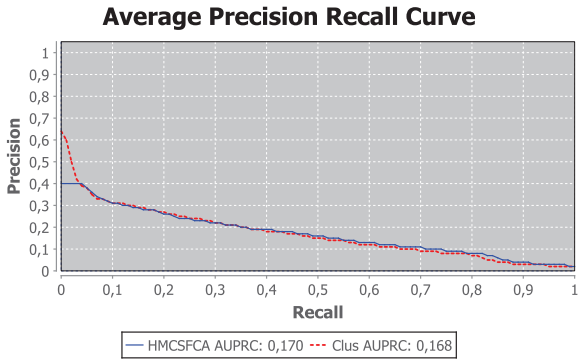


(e) *maxAtt* = ∞.

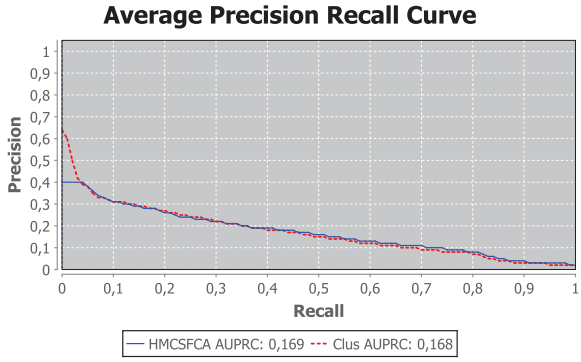
Figura A.7: Curvas PR comparativas para a base de dados cellcycle_FUN.



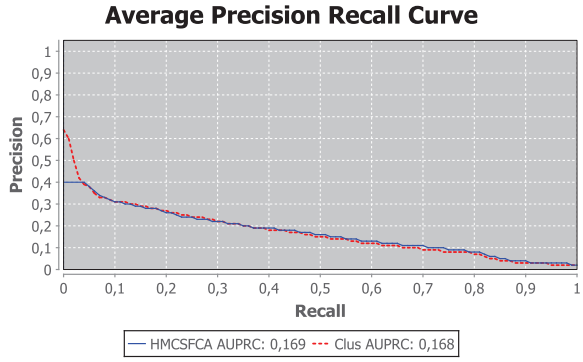
(a) *maxAtt* = 4.



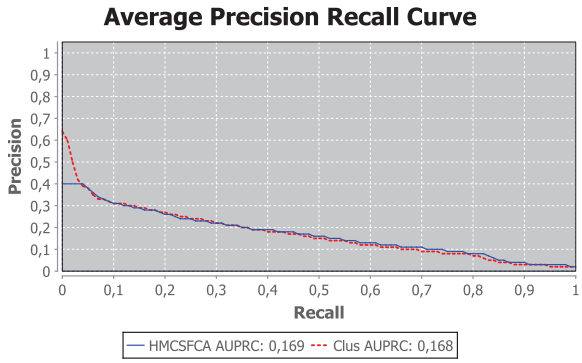
(b) *maxAtt* = 6.



(c) *maxAtt* = 8.



(d) *maxAtt* = 12.



(e) *maxAtt* = ∞.

Figura A.8: Curvas PR comparativas para a base de dados church_FUN.