LUIZ MELO ROMÃO

Classificação Global Hierárquica Multirrótulo da Função de Proteínas Utilizando Sistemas Classificadores

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

LUIZ MELO ROMÃO

Classificação Global Hierárquica Multirrótulo da Função de Proteínas Utilizando Sistemas Classificadores

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. Julio César Nievola

Curitiba
2012

Romão, Luiz Melo
Classificação Global Hierárquica Multirrótulo da Função de Proteínas Utilizando Sistemas Classificadores. Curitiba, 2012.

Tese - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. Sistemas Classificadores 2. Problemas de Classificação Hierárquica 3. Predição da Função de Proteínas I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e Tecnologia. Programa de Pós-Graduação em Informática II - t

Dedico a meus pais Geraldo e Wanda e a minha esposa Geisa, com muito amor.

Agradecimentos

Agradeço a Deus pela saúde e por ter me guiado para a conclusão desta Tese.

A minha esposa Geisa pelo carinho, apoio e compreensão nesta longa jornada.

Aos meus pais Geraldo e Wanda e meu irmão Wagner, pelos valiosos ensinamentos sendo fontes de inspiração pessoal.

Ao meu enteando Pedro Henrique e a toda minha família, sobrinhos, cunhada, avó, sogros, tios, tias, primos, primas, amigos e amigas pelo apoio e amizade.

Ao meu orientador Prof. Dr. Júlio César Nievola, pelos valiosos ensinamentos e pela total dedicação a mim prestada em todo o processo de doutoramento.

Ao "JavaMan" Prof. Me. Walter Coan pela ajuda dedicada no desenvolvimento do sistema.

E a todas as pessoas envolvidas nas entidades PucPr, Univille e Softville que confiaram no meu trabalho e pelo apoio moral e financeiro.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

ACS Anticipatory Classifier Systems

AG Algoritmos Genéticos

ALP Processo de Aprendizagem Antecipada

AUC Area Under the ROC

CCET Centro de Ciências Exatas e de Tecnologia

CLUS-HMC Hierarchical Multi-Label Classification

CLUS-HSC Hierarchical Single-Label Classification

CLUS-SC Single-Label Classification

DAG Directed Acyclic Graph

E Especificidade

EC Code Classificação Internacional das Enzimas

FN Falso Negativo

FP Falso Positivo

GMNB Global-Model Naive Bayes

FunCat MIPS Functional Catalogue

GO Ontologia Gênica

GPCR G-Protein-Coupled Receptors

HLCS Hierarchical Learning Classifier System

HLCS-Local Hierarchical Learning Classifier System - Local

HLCS-Tree Hierarchical Learning Classifier System - Tree

HLCS-DAG Hierarchical Learning Classifier System - DAG

HLCS-Multi Hierarchical Learning Classifier System - Multilabel

hmAnt-Miner Hierarchical Multi-Label Classification Ant-Miner

HMC Classificação Hierárquica Multirrótulo

HMC-CT HMC-Cross-Training

HMC-LP HMC-Label-Powerset

hR Revocação Hierárquica

hP Precisão Hierárquica

KDD Knowledge Discovery in Databases

LCS Sistemas Classificadores

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

P Precisão

PCT Predictive Clustering Trees

PPGIa Programa de Pós-Graduação em Informática

PR Precision-Recall

PUCPR Pontifícia Universidade Católica do Paraná

R Revocação

RE Rule Evolution

RIPPER Repeated Incremental Pruning to Produce Error Reduction

ROC Receiver Operating Characterisitics

SAI Sistema Imunológico Artificial

SC Sequential Covering

TA Taxa de Acerto

TE Taxa de Erro

TDIDT Top-down induction of decision trees

VN Verdadeiro Negativo

VP Verdadeiro Positivo

WEKA Waikato Environment for Knowledge Analysis

XCS eXtended Classifier System

ZCS Zeroth Level Classifier System

Resumo

Vários são os problemas que têm sido tratados pela bioinformática, entre estes, se destaca a predição de funções biológicas de proteínas. A complexidade deste tipo de aplicação vem da própria estrutura de organização da proteína que descreve estas funções utilizando uma hierarquia estruturada em árvores ou em grafos acíclicos dirigidos. O conceito do problema da classificação hierárquica pode ser ainda mais complexo nos casos onde, além das classes serem estruturadas em uma hierarquia, as instâncias podem ter suas classes associadas a dois ou mais caminhos na estrutura hierárquica, como é o caso das principais ontologias utilizadas atualmente para a predição de funções de proteína que são FunCat e Ontologia Gênica. Existem trabalhos envolvendo classificação hierárquica da função de proteínas, entretanto, nem todos levam em consideração a estrutura hierárquica das proteínas durante o desenvolvimento dos modelos de classificação, fato importante que deve ser usado para obter uma predição de melhor qualidade. Além disso, em muitos casos a condição multirrótulo também não é considerada. Nesta tese é apresentado o algoritmo Hierarchical Learning Classifier System Multilabel (HLCS-Multi) que exibe uma solução multirrótulo global para a classificação hierárquica da função de proteínas, respeitando a hierarquia das classes em todas as fases de desenvolvimento do modelo. O HLCS-Multi foi desenvolvido especificamente para trabalhar com bases hierárquicas e utiliza como modelo de desenvolvimento os Sistemas Classificadores (LCS), gerando seus resultados em um conjunto de regras no formato SE-ENTÃO, que são representações mais compreensíveis do que modelos como redes neurais, máquinas de vetor de suporte, entre outros. O modelo proposto foi analisado contra os algoritmos RIPPER, GMNB, hAnt-Miner e Clus-HMC através das medidas de revocação hierárquica, precisão hierárquica e curva PR com bons resultados.

Palavras-chave: Predição da Função de Proteínas, Classificação Hierárquica Multirrótulo, Sistemas Classificadores.

Abstract

There are several problems that have been dealt with bioinformatics. Among them is the prediction of biological functions proteins. The complexity of this type of application comes from the protein's organization structure that describe these functions using a structured hierarchy trees or directed acyclic graphs. The concept of hierarchical classification problem can be further complicated in cases where, besides classes are organized into a hierarchy, the instances may have their classes associated with two or more paths in the hierarchical structure, as is the case of the main ontologies used currently for predicting protein functions that are FunCat and Gene Ontology. There are some studies involving hierarchical classification of protein function. However, not everyone takes into account the hierarchical structure of the proteins during the development of the models, an important fact that should be used for a prediction of better quality. Furthermore, in many cases the multilabel condition is not considered. This thesis presents the algorithm Hierarchical Learning Classifier System Multilabel (Multi-HLCS) that shows a multilabel global solution to the hierarchical classification of protein function, respecting the hierarchy of classes at all stages of model development. The Multi-HLCS is developed specifically to work with hierarchical bases and uses as a model for the development the Learning Classifier Systems (LCS), generating its results on a set of rules in the form IF-THEN representations that are more understandable than models like neural networks , support vector machines, among others. The proposed model was analyzed against RIPPER , GMNB, Hant-Miner and Clus-HMC algorithms through hierarchical recall, hierarchical precision measures and curve PR with good results.

Keywords: Prediction of protein function, Hierarchical Multilabel Classification, Learning Classifier System.

Capítulo 1

Introdução

A bioinformática vem se tornando fundamental para a biologia no século 21. Com o projeto de sequenciamento do genoma, os bancos de dados biológicos têm sido sobrecarregados com a geração de terabytes de dados experimentais diariamente. De acordo com (JUNGCK et al., 2010), a aplicação de ferramentas baseadas em computador para armazenamento, distribuição e principalmente aprendizagem destes dados, se tornou importante para ajudar os pesquisadores a extrair informações específicas a respeito de processos biológicos.

Vários são os problemas que têm sido tratados pela bioinformática, entre estes, se destaca a predição de funções biológicas de proteínas. De acordo com (ALVES, 2010), a predição da função de proteínas, consiste em associar funções biológicas para novas sequências de proteínas. Este conhecimento pode auxiliar os pesquisadores num melhor entendimento de doenças, no desenvolvimento de fármacos, na medicina preventiva, entre outros.

Atualmente, de acordo com (KING; WISE; CLARE, 2004), as principais ferramentas de predição da função de proteínas são: FASTA (PEARSON; LIPMAN, 1988) e PSI-BLAST (ALTSCHUL et al., 1997). Estas ferramentas fazem suas predições baseadas no método da homologia. Neste método, uma nova sequência de aminoácidos é comparada com outras sequências em uma base de dados para se obter uma maior similaridade. Quando encontrada, a função da sequência mais similar é inferida para a nova. O problema deste método conforme (FRIEDBERG, 2006) é que, embora em geral as sequências similares de aminoácidos resultem 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 executar diferentes funções. Além disso, conforme (RIGDEN; MELLO, 2002), embora sequências diferentes sejam compatíveis com uma mesma estrutura, proteínas de enovelamentos diferentes podem ter a mesma função.

Com estes problemas, outras formas de predição da função de proteínas têm sido desenvolvidas e uma delas é através da mineração de dados. A mineração de dados é uma etapa de um processo maior denominado de descoberta do conhecimento em base de dados (KDD), do inglês *Knowledge Discovery in Databases*, caracterizado pela busca de padrões nos dados.

O KDD refere-se ao processo não trivial de identificação de novos padrões válidos, potencialmente úteis e compreensíveis em conjuntos de dados (FAYYAD et al., 1996). O KDD consiste no uso de métodos de várias áreas, principalmente, aprendizagem de máquina e estatística, para extrair conhecimento de um conjunto de dados do mundo real. O processo natural na descoberta do conhecimento envolve três etapas principais que são: o pré processamento ou preparação dos dados, a mineração dos dados e a etapa de pós processamento ou refinamento do conhecimento.

A etapa de mineração de dados é um campo interdisciplinar onde podem ser utilizados diferentes tipos de técnicas para a descoberta de padrões, como por exemplo, classificação, agrupamento, associação e regressão, sendo o foco deste trabalho o estudo do paradigma de classificação.

A tarefa de classificação está associada com a predição, ou seja, de acordo com o conhecimento prévio extraído de uma base de dados de treinamento, procura-se descobrir com qual das classes existentes, novos dados serão rotulados. Dado um conjunto de instâncias, cada instância de dados pertence a uma determinada classe que é indicada pelo valor de um atributo principal (FREITAS, 2002). Cada instância consiste portanto em duas partes: uma parte que contêm os atributos de previsão da instância e outra parte, que é o atributo principal, que identifica a classe da instância. Por exemplo, se o atributo principal indica se um animal é mamífero ou não, os atributos de previsão devem trazer informações relevantes como as características dos animais mamíferos.

Os dois principais tipos de classificação de dados são denominados de classificação plana e classificação hierárquica. A maioria dos trabalhos citados na literatura envolve o tipo de classificação plana, onde uma instância da base de treinamento está relacionada a apenas uma determinada classe em um único nível. Porém, existe um vasto número de problemas cujos dados estão dispostos em uma hierarquia, como por exemplo, a predição da função de proteínas em dados de bioinformática (FREITAS; CARVALHO, 2007).

Problemas deste tipo, são problemas onde uma ou mais classes podem ser divididas em subclasses ou agrupadas em superclasses (CERRI et al., 2008). Nesse caso, as classes são dispostas em uma estrutura hierárquica, tal como uma árvore ou um grafo acíclico direcionado (DAG) do inglês *Directed Acyclic Graph*. A principal diferença entre a estrutura de árvore e a estrutura de DAG é que na estrutura de árvore cada nó de

classe, exceto o nó raiz, tem apenas um pai, enquanto que na estrutura de DAG cada nó de classe pode ter um ou mais nós pai.

O conceito do problema da classificação hierárquica pode ser ainda mais complexo nos casos onde, além das classes serem estruturadas em uma hierarquia, as instâncias podem ter suas classes associadas a dois ou mais caminhos na estrutura hierárquica, como é o caso das principais ontologias utilizadas atualmente para a predição de funções de proteína que são FunCat e Ontologia Gênica. Neste caso, conforme (CERRI; CARVALHO; FREITAS, 2011), o problema é conhecido como classificação hierárquica multirrótulo (HMC) do inglês *Hierarchical Multi-Label Classification*.

Existem trabalhos envolvendo classificação hierárquica da função de proteínas, entretanto, nem todos levam em consideração a estrutura hierárquica das proteínas durante o desenvolvimento dos modelos, fato importante que deve ser usado para uma predição de melhor qualidade. Além disso, em muitos casos a condição multirrótulo também não é considerada.

Nesta tese é apresentado o algoritmo Hierarchical Learning Classifier System Multilabel (HLCS-Multi) que exibe uma solução multirrótulo global para a classificação hierárquica da função de proteínas, respeitando a hierarquia das classes em todas as fases de desenvolvimento do modelo. O HLCS-Multi foi desenvolvido especificamente para trabalhar com bases hierárquicas e utiliza como modelo de desenvolvimento os Sistemas Classificadores (LCS), gerando seus resultados em um conjunto de regras no formato SE-ENTÃO, que são representações, de acordo com (FREITAS; WIESER; APWEILER, 2010) mais compreensíveis do que modelos como redes neurais, máquinas de vetor de suporte, entre outros.

1.1 Desafio

De acordo com (LESK, 2008), para desenvolver um fármaco contra uma determinada doença, é necessário selecionar uma proteína associada à doença, de forma que ele seja terapeuticamente interessante para afetar sua função e expressão. Com isso, as comunidades científica e médica, para terem sucesso em seus experimentos, são dependentes da qualidade dos bancos de dados utilizados.

Estes bancos de dados, são formados principalmente, por experimentos realizados com ferramentas como FASTA e PSI-BLAST, que, como já foi citado, trabalham utilizando o método de predição da função de proteínas baseado em homologia. Entretanto, análises feitas no mapeamento das funções de proteínas que utilizam o método da homologia, mostraram que em cerca de 40% dos casos, uma sequência não mostra similaridade

significativa com uma proteína já caracterizada (RIGDEN; MELLO, 2002). Como estas ferramentas não são sensíveis o suficiente para descobrir outras semelhanças entre estas proteínas, anotações errôneas podem estar sendo propagadas pelos bancos de dados na mesma velocidade com que novas sequências vem sendo analisadas.

Desta forma, conforme (RIGDEN; MELLO, 2002), o desenvolvimento de ferramentas que permitam a diminuição das falhas que levam a uma interpretação errada, refletirá diretamente na melhoria das bases de dados para análise de proteínas.

Com isso, o uso da mineração de dados, através do método de classificação por exemplo, pode auxiliar no processo de classificação funcional das proteínas. Estes algoritmos, ao invés de realizar a predição através da comparação de sequências, trabalham com informações de diversos tipos de elementos da proteína, como por exemplo: atributos de aminoácidos, estrutura, fusão de genes, proximidade de cromossomos, padrões filogenéticos, entre outros.

A complexidade deste tipo de aplicação vem da própria estrutura de organização da proteína. Atualmente, tem sido cada vez mais frequente o uso de ontologias para organização de funções de proteínas, que descrevem estas funções utilizando uma hierarquia estruturada em árvores ou em DAG.

Nas ontologias estruturadas em DAG, a abordagem que vem sendo mais utilizada pelos pesquisadores é a Ontologia Gênica (GO). De acordo com (ASHBURNER et al., 2000), a GO prevê uma ontologia de termos definidos que representam as propriedades do gene, formando um vocabulário de termos consistente e estruturado para descrever domínios chave da biologia molecular, incluindo atributos e produtos gênicos, assim como sequências biológicas. Estes termos podem ser anotados para sequências, genes ou produtos gênicos armazenados em bases de dados biológicos. A GO descreve atributos de produtos gênicos em três domínios disjuntos da biologia molecular: função molecular, processo biológico e componente celular.

A classificação hierárquica da função de uma proteína estruturada em um DAG é bastante complexa, pois um nó pode ter mais de um pai. Isto faz com se tenha um problema de classificação hierárquica multirrótulo e um modelo de predição para este tipo de cenário deve ser desenvolvido, preferencialmente, respeitando a hierarquia das classes durante todas as fases. Isso se deve ao fato que as informações dos antecedentes e descendentes de uma classe são importantes para uma correta predição da classe em questão. Outra dificuldade no processo preditivo, está diretamente relacionada com a profundidade da hierarquia. Normalmente, o desempenho preditivo diminui com o aumento da profundidade (especificidade), visto que a quantidade de exemplos mais específicos é menor, o que dificulta o processo de treinamento do modelo e predição da instância.

Além disso, um fator importante para a predição da função de proteínas de acordo com (FREITAS; WIESER; APWEILER, 2010), é que o modelo de predição desenvolvido, além de ser eficaz, deve ser compreensível o bastante para que os pesquisadores consigam analisar e confiar nas predições recebidas.

1.2 Objetivos

1.2.1 Objetivo Geral

O objetivo principal deste trabalho é desenvolver e avaliar um algoritmo para a classificação global hierárquica multirrótulo da função de proteínas, baseado no modelo dos Sistemas Classificadores.

1.2.2 Objetivos Específicos

Os objetivos específicos são:

- Estudar as características de um conjunto de base de dados de proteínas disponíveis publicamente;
- Avaliar as formas de hierarquia pelas quais as proteínas são classificadas;
- Desenvolver e implementar um Sistema Classificador Global Hierárquico Multirrótulo para a função de proteínas;
- Analisar e implementar formas para avaliar o sistema criado;
- Avaliar os resultados obtidos do algoritmo proposto em comparação com outros modelos.

1.3 Solução Proposta

O algoritmo Hierarchical Learning Classifier System Multilabel (HLCS-Multi) proposto neste trabalho, apresenta uma solução global hierárquica multirrótulo para a classificação da função de proteínas respeitando a hierarquia das classes em todas as fases de desenvolvimento do sistema. É desenvolvido especificamente para trabalhar com bases hierárquicas e utiliza como modelo de desenvolvimento os Sistemas Classificadores (LCS). O HLCS-Multi é o primeiro modelo baseado em Sistemas Classificadores para a predição de problemas hierárquicos multirrótulo.

Para trabalhar com dados hierárquicos, o HLCS-Multi apresenta um componente específico para esta tarefa que é o componente de avaliação dos classificadores. Este componente tem a função de analisar as predições dos classificadores considerando a hierarquia das classes. Além do componente de avaliação, a arquitetura HLCS-Multi consiste dos seguintes módulos: população de classificadores, componente AG, componente de desempenho e componente de cessão de créditos, que interagem entre si.

Em geral, o sistema HLCS-Multi cria sua população de classificadores através da análise dos exemplos do ambiente, que representam os dados de treinamento. A medida que o HLCS-Multi interage com o ambiente, por meio dos componentes de desempenho e cessão de créditos, os classificadores são avaliados e recebem um retorno na forma de uma recompensa que impulsiona o processo de aprendizagem. Este processo de aprendizagem tem a intervenção do componente de avaliação, que ajuda no cálculo da recompensa avaliando as predições do sistema conforme a sua hierarquia. No fim desta etapa, os classificadores passam por um mecanismo de evolução através do componente AG, que utiliza algoritmos genéticos como base para a melhoria do conhecimento atual do sistema.

A questão multirrótulo é trabalhada durante todas as etapas de treinamento e teste do modelo por meio de um processo de decomposição das instâncias e na forma de análise das predições.

O modelo proposto foi analisado contra os algoritmos RIPPER (COHEN, 1995), GMNB (SILLA; FREITAS, 2009), hAnt-Miner (OTERO; FREITAS; JOHNSON, 2009) e Clus-HMC (VENS et al., 2008) através das medidas de revocação hierárquica, precisão hierárquica e curva PR com bons resultados.

1.4 Organização

Esta tese está organizada em seis capítulos. O Capítulo 2 descreve os conceitos sobre classificação hierárquica multirrótulo, sistemas classificadores e proteínas que proporcionam o embasamento teórico, bem como a contextualização em que esse trabalho se insere. O Capítulo 3 apresenta os trabalhos correlatos referentes ao tema desta proposta de tese. O Capítulo 4 descreve a proposta de tese, mostrando a evolução do sistema até se chegar no modelo HLCS-Multi. O Capítulo 5 mostra os resultados obtidos com o algoritmo proposto. O Capítulo 6 relata as conclusões deste trabalho.

Capítulo 2

Fundamentos

Neste capítulo são apresentados os principais conceitos sobre os assuntos que serão utilizados neste trabalho. Na Seção 2.1 são abordados os conceitos sobre classificação de dados, problemas de classificação hierárquica multirrótulo e medidas para avaliação dos algoritmos de classificação. Na Seção 2.2 é descrito o funcionamento dos Sistemas Classificadores assim como seus componentes. Na Seção 2.3 os principais conceitos sobre proteínas, bem como a estrutura hierárquica das ontologias ligadas as proteínas são apresentados.

2.1 Classificação de Dados

A classificação é um dos problemas mais importantes da aprendizagem de máquina e da mineração de dados (FREITAS; CARVALHO, 2007). De acordo com (TAN; STEINBACH; KUMAR, 2005), classificação é a tarefa de aprender uma função (f) que mapeie um conjunto de atributos (x) a uma classe de rótulos predefinida (y).

A técnica de classificação é uma abordagem sistemática para a construção de modelos de classificação a partir de um conjunto de dados de entrada. As técnicas mais utilizadas para classificação de dados são: árvores de decisão, redes neurais artificiais, naïve bayes e máquinas de vetores de suporte. A técnica utilizada neste trabalho e que será discutida adiante, chama-se Sistemas Classificadores. Cada técnica utiliza um algoritmo de aprendizagem para identificar um modelo que melhor se adapte a relação entre o conjunto de atributos e rótulo da classe dos dados de entrada (TAN; STEINBACH; KUMAR, 2005). O modelo de classificação desenvolvido deve ser capaz de prever com precisão rótulos de classes de registros desconhecidos.

A Figura 2.1 mostra uma abordagem geral para a construção de um modelo de classificação. Primeiro, um conjunto de treinamento, consistindo de instâncias, cujos

rótulos de classe são conhecidos devem ser fornecidos. O conjunto de treinamento é usado para construir o modelo de classificação, que é posteriormente aplicado ao conjunto de teste, que consiste em instâncias com rótulos de classe desconhecida.

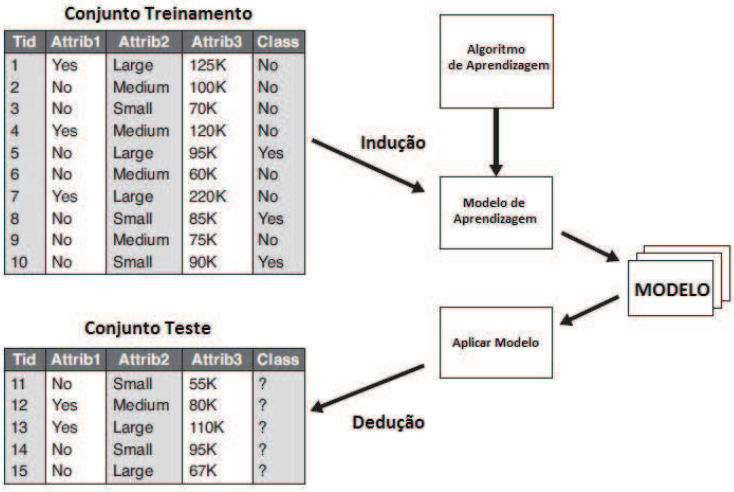

Figura 2.1: Abordagem geral para a construção de um modelo de classificação - Adaptado (TAN; STEINBACH; KUMAR, 2005)

Os dois principais tipos de classificação de dados são denominados de classificação plana e classificação hierárquica.

A classificação plana é mais simples e é utilizada pela maioria dos problemas citados na literatura. Nesta classificação, cada instância da base de dados está diretamente associada a apenas uma classe. Na classificação hierárquica, que é o modelo utilizado neste trabalho, todas as classes estão dispostas em uma estrutura hierárquica e cada instância pode estar associada a duas os mais classes, o que torna a criação do modelo de classificação ainda mais complexo. Na seção a seguir, este modelo será melhor detalhado.

2.1.1 Classificação Hierárquica de Dados

Como dito anteriormente, a grande maioria dos problemas de classificação descritos na literatura diz respeito a problemas de classificação não hierárquica, em que cada instância é associada a uma classe pertencente a um conjunto finito de classes, não considerando assim relacionamentos hierárquicos (CERRI; CARVALHO; FREITAS, 2011). Entretanto, muitos problemas reais apresentam cenários de classificação mais complexos. Casos como categorização de textos, ferramentas de busca, predição da função de proteínas, aplicações de livraria digital, entre outros, são problemas em que os rótulos das classes são estruturados de forma hierárquica. Problemas deste tipo são problemas onde uma ou mais classes podem ser divididas em subclasses ou agrupadas em superclasses (CERRI et

al., 2008). Nesse caso, as classes são dispostas em uma estrutura hierárquica, tal como uma árvore ou um grafo acíclico direcionado (DAG) do inglês *Directed Acyclic Graph*. A principal diferença entre a estrutura de árvore e a estrutura de DAG é que na estrutura de árvore cada nó de classe, exceto o nó raiz, tem apenas um pai, enquanto que na estrutura de DAG cada nó de classe pode ter um ou mais nós pai, como mostra a Figura 2.2.

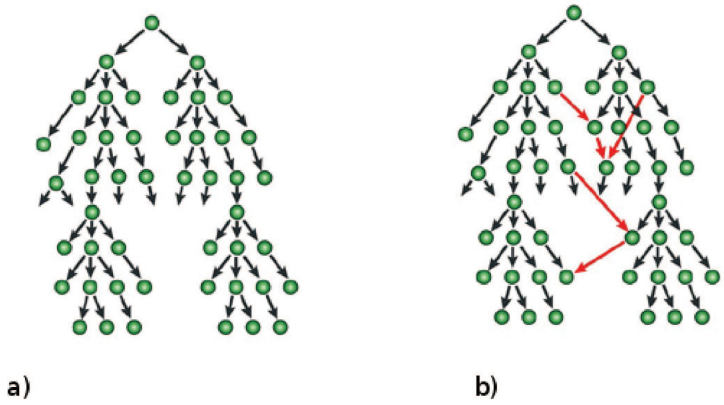

Figura 2.2: Exemplo de uma Estrutura em Árvore (a) e uma Estrutura em DAG (b) Adaptado de (BARD; RHEE, 2004)

O conceito de classificação hierárquica pode ser visto como um tipo específico de problema de classificação estruturada, em que a saída do algoritmo de classificação é definida por uma taxonomia de classes. Os conceitos utilizados neste trabalho a respeito de classificação hierárquica, seguem os princípios apresentados no trabalho de (SILLA; FREITAS, 2011).

Uma taxonomia de classes foi explorada em (WU; ZHANG; HONAVAR, 2005), como um conceito hierárquico definido sobre um conjunto parcialmente estabelecido $(C, \prec)$, onde C é um conjunto finito que enumera todos os conceitos de uma classe no domínio da aplicação e a relação $\prec$ representa um relacionamento do tipo "*É-UM*". Assim, este relacionamento é definido como uma relação assimétrica, anti-reflexiva e transitiva, como segue:

- Somente o maior elemento "R" é a raiz da árvore.

- $\forall c_i, c_j \in C$, se $c_i \prec c_j$ então $c_j \not\prec c_i$.

- $\forall c_i \in C$, $c_i \not\prec c_i$.

- $\forall c_i, c_j, c_k \in C$, $c_i \prec c_j$ e $c_j \prec c_k$ implica $c_i \prec c_k$.

Portanto, seguindo os conceitos apresentados, qualquer problema de classificação com uma estrutura de classes que satisfaça as quatro propriedades mencionadas da hierarquia "*É-UM*", pode ser considerado como um problema de classificação hierárquica. Esta definição originalmente proposta para taxonomia de classe para estruturas em árvores, também pode ser definida como taxonomia de classe para estruturas DAG, sendo que a diferença entre estas estruturas deve-se a uma maior complexidade das soluções para DAG.

Os problemas que envolvem soluções de classificação hierárquica são bastante específicos e é possível diferenciá-los por meio de alguns critérios (SILLA; FREITAS, 2011):

- Estrutura que o algoritmo pode manipular;
- Profundidade de predição do algoritmo;
- Cardinalidade da predição do algoritmo;
- Tipo de abordagem do algoritmo.

Cada uma destas propriedades serão detalhadas a seguir.

Estrutura que o algoritmo pode manipular

A estrutura dos dados em problemas hierárquicos pode estar disposta de duas formas: em árvore ou em DAG, como mostram as Figuras 2.3 e 2.4, respectivamente. Nas figuras, cada nó representa uma classe identificada por um valor e as arestas representam as relações entre a classe ancestral e a classe descendente. A principal diferença entre a estrutura de árvore e a estrutura de DAG é que na estrutura de árvore cada nó de classe, exceto o nó raiz, tem apenas um pai, enquanto que na estrutura de DAG cada nó de classe pode ter um ou mais nós pai.

Profundidade de predição do algoritmo

Esta característica leva em consideração a profundidade com que a classificação é realizada. Neste caso, esta característica pode ser separadas em duas categorias: predição obrigatória em nós-folha e predição opcional em nós-folha.

A primeira categoria refere-se que todos as instâncias devem ser associados com classes representadas por nós-folha. Neste caso, ao predizer uma classe no nível nó-folha,

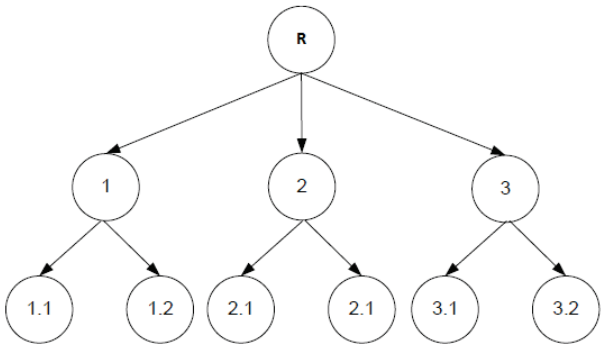

Figura 2.3: Exemplo de hierarquia de classe estruturada em árvore

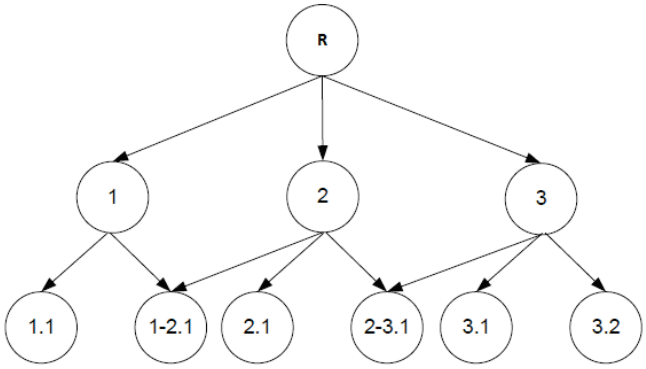

Figura 2.4: Exemplo de hierarquia de classe estruturada em DAG

o algoritmo também estará predizendo as classes nos níveis acima. Como mostra a Figura 2.5, somente as classes denominadas 1.1, 1.2, 2.1, 2.2, 3.1 e 3.2 podem ser preditas.

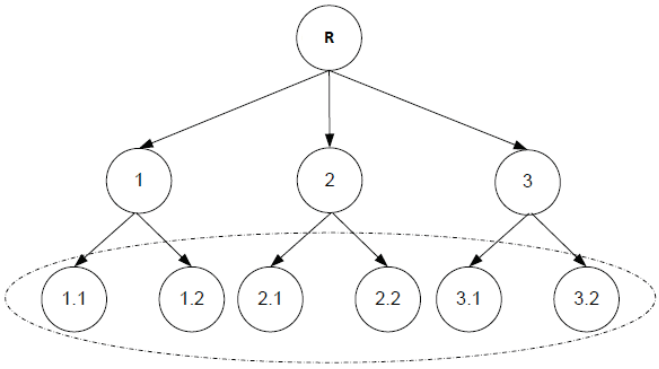

Figura 2.5: Exemplo de predição obrigatória em nós-folha

Já na categoria predição opcional em nós-folha, o algoritmo pode finalizar a predição em qualquer nó em qualquer nível da hierarquia, como mostra a Figura 2.6.

Cardinalidade da predição do algoritmo

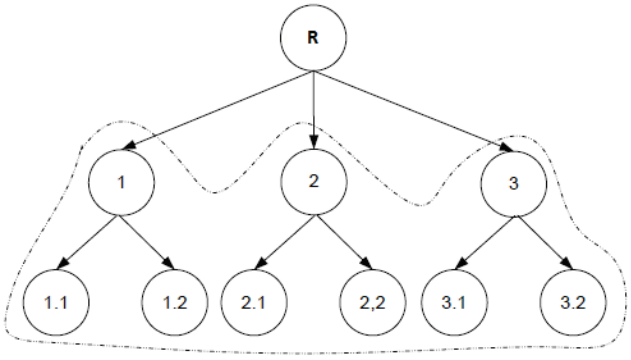

Figura 2.6: Exemplo de predição opcional em nós-folha

Esta característica indica se o algoritmo pode predizer classes em apenas um caminho ou em múltiplos caminhos diferentes na hierarquia. Este critério pode assumir dois valores: Predição de Caminho Simples e Predição de Caminhos Múltiplos.

A Predição de Caminho Simples indica que o algoritmo pode atribuir a cada instância de dados, no máximo, um caminho de rótulos previstos. A Predição de Caminhos Múltiplos indica que o algoritmo pode, potencialmente, atribuir a cada instância de dados múltiplos caminhos de rótulos previstos.

Tipo de abordagem do algoritmo

Para explorar os problemas de classificação hierárquica, alguns algoritmos têm sido propostos, e estes podem ser divididos em três abordagens principais: algoritmos de classificação plana, algoritmos de classificação local, divididos em local por nó, local por nó pai e local por nível, e algoritmos de classificação global (SILLA; FREITAS, 2011).

A classificação plana é a abordagem mais simples para se lidar com problemas de classificação hierárquica. Consiste em ignorar completamente a hierarquia de classe, geralmente prevendo apenas a classe dos nós-folha. 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 folhas), sem explorar informações sobre as relações pais-filhos presentes na hierarquia de classes. A Figura 2.7 exemplifica esta abordagem.

A abordagem de classificação local por nó é a mais utilizada na literatura e consiste em treinar um classificador binário para cada nó da hierarquia de classe como mostra a Figura 2.8. Neste caso é necessário utilizar N classificadores locais independentes, um

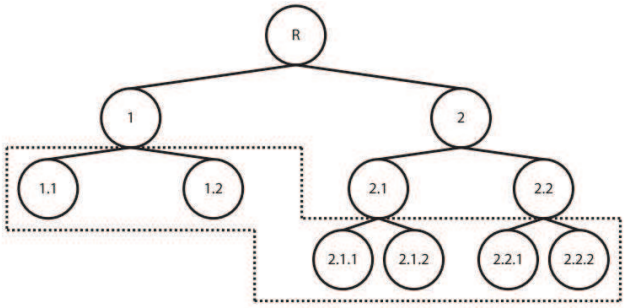

Figura 2.7: Exemplo de classificação plana (SILLA; FREITAS, 2011)

para cada classe exceto o nó raiz. Com isso, a quantidade de classificadores a serem treinados pode ser muito grande em situações onde existam muitas classes. Além disso, da mesma forma que ocorre na abordagem plana, essa técnica pode apresentar resultados incoerentes, pois não há nenhuma garantia que a hierarquia de classes será respeitada.

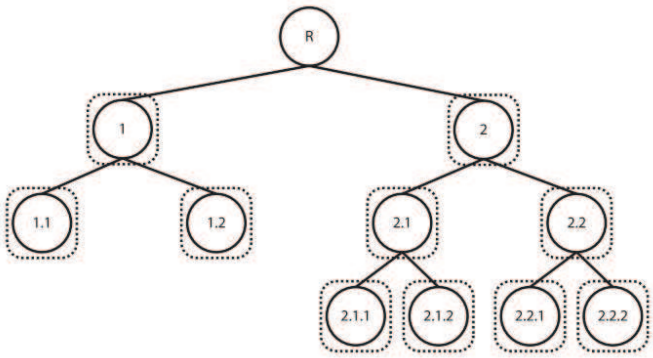

Figura 2.8: Exemplo de classificação local por nó (SILLA; FREITAS, 2011)

Na abordagem de classificação local por nó pai, para cada nó pai na hierarquia de classe um classificador multirrótulo é treinado para distinguir entre seus nós filhos. Considerando que a classificação ocorra de cima para baixo, suponha, de acordo com a Figura 2.9, 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.

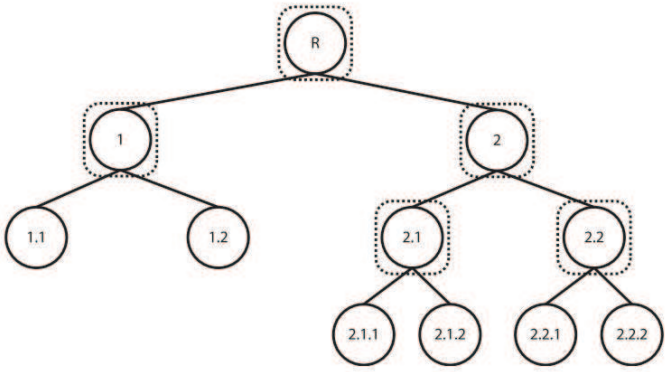

Figura 2.9: Exemplo de classificação local por nó pai (SILLA; FREITAS, 2011)

A abordagem de classificação local por nível é a menos utilizada na literatura e consiste em um classificador multirrótulo para cada nível da hierarquia de classes, conforme mostra o exemplo da Figura 2.10.

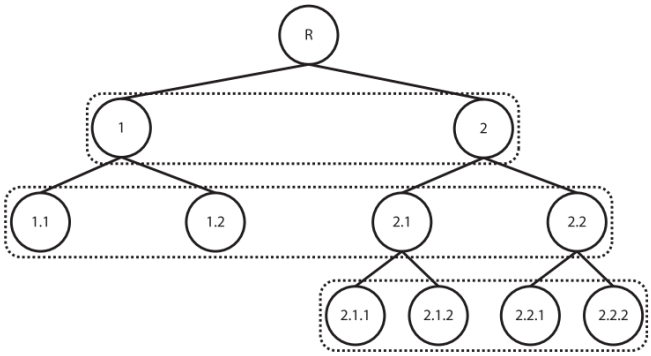

Figura 2.10: Exemplo de classificação local por nível(SILLA; FREITAS, 2011)

Esta técnica pode ser usada em estruturas em árvore e em DAG. Em um DAG a aplicação desta técnica é mais complexa, visto que pode existir mais de um caminho nesse tipo de estrutura. Assim, uma classe pode pertencer a mais de um nível na hierarquia o que pode trazer redundância entre os classificadores.

Na última abordagem chamada de classificação global 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 2.11.

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

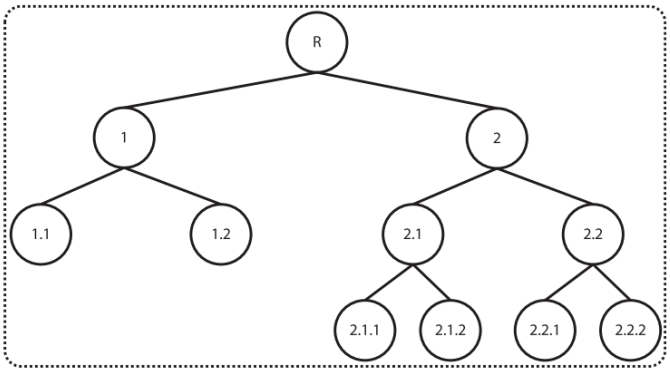

Figura 2.11: Exemplo de classificação global (SILLA; FREITAS, 2011)

2.1.2 Problemas de Classificação Hierárquica Multirrótulo

O conceito de problemas de classificação hierárquica pode ser ainda mais complexo nos casos onde, além das classes serem estruturadas em uma hierarquia, as instâncias podem ter suas classes associadas a dois ou mais caminhos na estrutura hierárquica. Neste caso, conforme (CERRI; CARVALHO; FREITAS, 2011), estes problemas são conhecidos como problemas de classificação hierárquica multirrótulo (HMC).

De acordo com (BLOCKEEL et al., 2006), um problema de HMC pode ser definido da seguinte forma:

Dado: um espaço de exemplos X; uma hierarquia de classes $(C, \leq_h)$, no qual C é um conjunto de classe e $\leq_h$ é uma ordem parcial representando o relacionamento de superclasse (para todo $c_1, c_2 \in C : c_1 \leq_h c_2$ se e somente se c_1 é uma superclasse de c_2); um conjunto T de exemplos (x_i, S_i) com $x_i \in X$ e $S_i \subseteq C$ tal que $c \in S_i \Rightarrow \forall \acute{c} \leq_h c : \acute{c} \in S_i$; um critério de qualidade q que recompense modelos com alta acurácia preditiva e baixa complexidade.

Encontrar: uma função $f : X \rightarrow 2^C$, na qual 2^C é o conjunto potência de C tal que $c \in f(x) \Rightarrow \forall \acute{c} \leq_h c : \acute{c} \in f(x)$ e f maximiza q.

Seguindo esta definição, e de acordo com as características definidas em (SILLA; FREITAS, 2011) para identificar os diferentes tipos de algoritmos de classificação hierárquica, neste trabalho, quando uma solução é dita hierárquica multirrótulo indica que o algoritmo pode, potencialmente, prever múltiplos caminhos de dados de classes.

Conforme (CARVALHO; FREITAS, 2009), existem basicamente duas formas de se tratar problemas de classificação multirrótulo: abordagem independente de algoritmo e abordagem dependente de algoritmo. Estas abordagens foram definidas para problemas não hierárquicos, mas podem ser adaptadas para problemas hierárquicos.

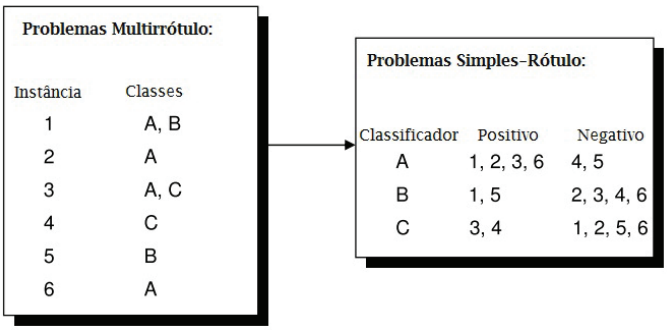

Figura 2.12: Transformação baseada nos rótulos de classes. Adaptado de (CARVALHO; FREITAS, 2009)

Abordagem Independente de Algoritmo

A abordagem independente de algoritmo utiliza algoritmos tradicionais de classificação para tratar problemas multirrótulo, transformando o problema multirrótulo original em um conjunto de problemas simples-rótulo. Essa transformação pode ser baseada nos rótulos de classes dos exemplos ou nos próprios exemplos de treinamento (CARVALHO; FREITAS, 2009).

Na transformação baseada nos rótulos de classes, são utilizados L classificadores, sendo L o número de classes que estão envolvidas no problema. Cada classificador é associado a uma classe e treinado para resolver um problema de classificação binária, onde é considerada a classe a qual ele está associado contra todas as outras classes envolvidas.

Um exemplo da utilização desta abordagem é ilustrado pela Figura 2.12, no qual é considerado o problema multirrótulo com três classes. Como cada classificador é associado a uma classe, três classificadores são treinados. O problema é então dividido em três problemas de classificação binária, sendo um para cada classe. O i-ésimo classificador é treinado para considerar os exemplos pertencentes à i-ésima classe como positivos e os outros exemplos negativos. Desta forma, cada classificador torna-se especializado na classificação de uma classe particular.

Na transformação baseada em exemplos, um conjunto de classes associado a cada exemplo é definido de maneira a converter o problema multirrótulo em um ou mais problemas simples-rótulo. Diferente da abordagem baseada nos rótulos das classes, o qual produz apenas problemas de classificação binária, a abordagem baseada em exemplos pode produzir tanto problemas de classificação binária quanto multirrótulo.

Conforme (CARVALHO; FREITAS, 2009), três diferentes estratégias têm sido propostas para este tipo de transformação.

- Eliminação de exemplos multirrótulo - nesta estratégia, considerada a mais simples

e também a mais ineficaz, são eliminados do conjunto de dados original as instâncias multirrótulo. As instâncias removidas podem representar informações importantes sobre o domínio do problemas e essas informações são perdidas.

- Criação de novos rótulos para os exemplos multirrótulo existentes - nesta estratégia, para cada instância, todas as classes atribuídas àquele exemplo são combinadas em uma nova e única classe. Com essa combinação, o número de classes envolvidas no problema pode aumentar consideravelmente e algumas classes podem terminar com poucos exemplos.

- Conversão de exemplos multirrótulo em exemplos simples-rótulos - nesta estratégia existem duas variações. Na primeira, todos as instâncias multirrótulo são convertidas para exemplos simples-rótulos, em um processo chamado de simplificação ou eliminação de rótulos. Uma segunda variação, chamada de decomposição de rótulos, decompõe todas as instâncias multirrótulo em um conjunto de exemplos simples-rótulo.

Abordagem Dependente de Algoritmo

O método de adaptação de algoritmo cria algoritmos específicos para tratar o problema multirrótulo. De acordo com (CARVALHO; FREITAS, 2009), um algoritmo específico, feito para determinado problema de classificação, pode apresentar resultados melhores do que técnicas que seguem a abordagem independente de algoritmo. Algumas soluções baseadas nesta abordagem foram propostas para a resolução de problemas de classificação multirrótulo. Dentre elas podem-se citar técnicas baseadas em: Árvores de Decisão (FREUND; MASON, 1999), (COMITÉ; GILLERON; TOMMASI, 2003) e (CLARE; KING, 2001), Máquinas de Vetores de Suporte (AIOLLI; SPERDUTI, 2005) e (SU et al., 2005), entre outros.

2.1.3 Medidas de Avaliação dos Algoritmos de Classificação

A avaliação do desempenho de um modelo de classificação é baseado nas contagens de registros de testes, corretamente e incorretamente preditas pelo modelo (TAN; STEINBACH; KUMAR, 2005). 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. Esta seção analisa alguns dos métodos usados para avaliar o desempenho de um classificador.

Em problemas de classificação binária utiliza-se como base uma matriz de confusão, onde as seguintes situações referentes a predição podem ocorrer:

- Verdadeiro Positivo (VP) - O exemplo é predito corretamente como pertencendo a 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 à classe positiva.

A Tabela 2.1 ilustra a distribuição dessas quatro situações em uma matriz de confusão.

Tabela 2.1: 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 as equações 2.1 e 2.2, respectivamente.

$$TA = \frac{|VN| + |VP|}{|VN| + |VP| + |FN| + |FP|} \tag{2.1}$$

$$TE = \frac{|FN| + |FP|}{|VN| + |VP| + |FN| + |FP|} = 1 - TA \tag{2.2}$$

Além da taxa de acerto e da taxa de erro, existem outras medidas que são utilizadas para avaliar a qualidade de um classificador como Revocação, Especificidade, Precisão e Medida-F, conforme definidos em (OLSON; DELEN, 2008).

A medida de Revocação (R), definida na Equação 2.3, 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.

$$R = \frac{|VP|}{|VP| + |FN|} \tag{2.3}$$

A medida de Especificidade (E), definida na Equação 2.4, mede a capacidade de se predizer uma classe negativa cuja predição está correta, ou seja, quantos exemplos negativos foram preditos do total de exemplos.

$$E = \frac{|VN|}{|VN| + |FP|} \tag{2.4}$$

A medida de Precisão (P), definida na Equação 2.5 calcula a probabilidade da predição positiva estar correta em relação a todas as instâncias.

$$P = \frac{|VP|}{|VP| + |FP|} \tag{2.5}$$

A Medida-F é obtida através da combinação da medida de Revocação com a medida de Precisão. Nesta fórmula utiliza-se uma constante β que geralmente recebe o valor 1, como mostra a Equação 2.6

$$Medida\_F = \frac{(\beta^2 + 1) * P * R}{\beta^2 * P + R} \tag{2.6}$$

Outra forma de avaliar os modelos de classificação é através das curvas ROC (*Receiver Operating Characterisitics*) e PR (*Precision-Recall*). As curvas ROC e PR são normalmente utilizadas para avaliar o desempenho de um algoritmo em um determinado conjunto de dados tomando como base os resultados gerados na matriz de confusão.

A curva ROC é uma técnica de visualização, organização e seleção de classificadores baseados em seu desempenho (FAWCETT, 2006). Criada através da relação entre as medidas de Revocação e Especificidade, com a curva ROC é possível observar como o número de exemplos positivos classificados corretamente varia de acordo com o número de exemplos negativos classificados incorretamente (DAVIS; GOADRICH, 2006). Curvas ROC são gráficos bidimensionais em que a taxa de Revocação é plotada no eixo Y e a taxa de

Especificidade é plotada no eixo X.

Já a curva PR utiliza a relação entre as medidas de Precisão e Revocação. A taxa de Precisão é plotada no Y e taxa de Revocação no eixo X. De acordo com (DAVIS; GOADRICH, 2006), uma diferença importante entre a curva ROC e a curva PR é a representação visual das curvas. As curvas PR podem expor diferenças entre os algoritmos que não são aparentes nas curvas ROC.

Em (KIRITCHENKO; MATWIN; FAMILI, 2005), é proposto uma medida de avaliação hierárquica (HM) que vem sendo utilizada para avaliação de classificadores hierárquicos, mais especificamente, para problemas com estruturas DAG. Este método segue o padrão das medidas de Revocação e Precisão com as seguintes alterações: cada exemplo pertence não somente a sua classe, mas também a todos os ancestrais da classe em um gráfico hierárquico, exceto o nó raiz. As novas medidas são chamadas de Revocação Hierárquica hR, Precisão Hierárquica hP e Medida-hF.

Formalmente estas medidas são calculadas da seguinte forma: Um conjunto de classes C_i atribuído a uma instância d_i é chamado de consistente com uma dada hierarquia se C_i forma um subgrafo adequadamente conectado a um grafo hierárquico com um nó raiz no topo, exemplo: Se $c_k \in C_i$ e $c_j \in Ancestrais(c_k)$, então $c_j \in C_i$. Assim, para cada instância (d_i, C_i) classificadas no subconjunto $C_i^{'}$ são ampliados os conjuntos C_i e $C_i^{'}$ com os seguintes ancestrais correspondentes: $A_i = \bigcup_{c_k \in C_i} Ancestrais(c_k)$, $A_i^{'} = \bigcup_{c_k \in C_i^{'}} Ancestrais(c_k)$. Desta forma, as medidas são calculadas como mostram as equações 2.7, 2.8 e 2.9.

$$hR = \frac{|A_i \bigcap A_i^{'}|}{|A_i|} \tag{2.7}$$

$$hP = \frac{|A_i \bigcap A_i^{'}|}{|A_i^{'}|} \tag{2.8}$$

$$Medida\_hF = \frac{(\beta^2 + 1) * hP * hR}{\beta^2 * hP + hR}, \beta \in [0, +\infty] \tag{2.9}$$

2.2 Sistemas Classificadores

O Sistema Classificador (LCS) é um método baseado em regras para aprendizagem de máquina que oferece uma abordagem holística para sua concepção, análise e compreensão (DRUGOWITSCH, 2007). Este método representa a fusão de diferentes campos de pesquisa encapsulados dentro de um único algoritmo. O LCS desenvolve um modelo de tomada de decisões inteligente, empregando duas metáforas biológicas, evolução e aprendizagem, onde a aprendizagem guia o componente evolucionário para se mover em direção das melhores regras. A Figura 2.13 ilustra a hierarquia de campos em que se encontram os Sistemas Classificadores.

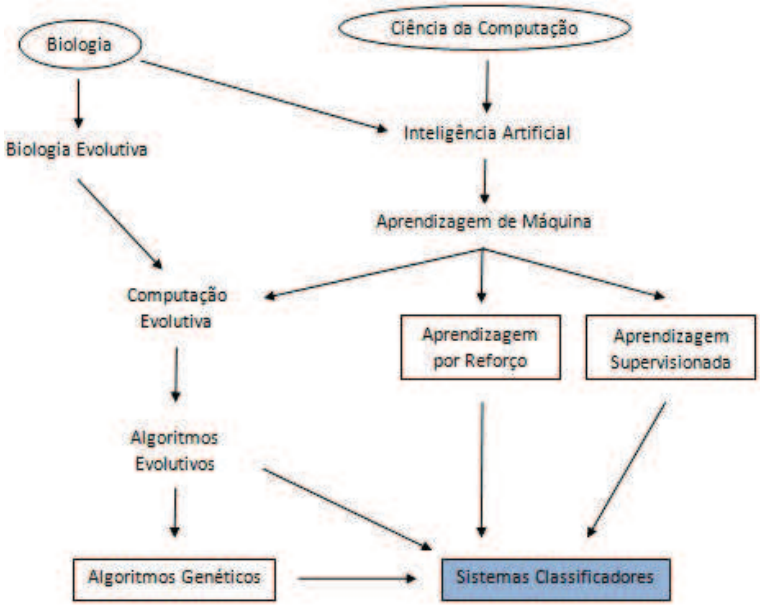

Figura 2.13: Fundamentos da comunidade LCS. Traduzido de (URBANOWICZ; MOORE, 2009)

Idealizado em 1975 por John Holland (HOLLAND, 1992), os sistemas classificadores consistem em um conjunto de regras representados por uma população de classificadores que, combinados com mecanismos adaptativos, são responsáveis pela sua evolução (SIGAUD; WILSON, 2007). Em (GOLDBERG, 1989), o autor acrescenta que o LCS "aprende" as regras para orientar o seu desempenho em um ambiente arbitrário, e (SIGAUD; WILSON, 2007) complementa que o LCS constroi seu conjunto de regras seguindo um formato do tipo "SE-ENTÃO", sendo a primeira parte da regra chamada de antecedente e que indica o conjunto de valores dos atributos e a segunda parte, chamada de consequente, que apresenta a classe prevista.

$$SE < atrib_1 = valor_1 > E < atrib_2 = valor_2 > ...E... < atrib_n = valor_n >$$
$$ENTAO < classe >$$

Os LCS são algoritmos de indução de regras de classificação, o que significa que o sistema, ao final da classificação, apresenta um conjunto de regras gerado a partir de um conjunto de dados, de forma que o usuário pode analisar e compreender as relações existentes entre os dados de uma mesma classe.

Um Sistema Classificador baseado nas ideias originais de Holland ou também chamado de abordagem Michigan, consiste de uma população de classificadores, que representa o conhecimento atual do sistema, um componente de desempenho que é responsável pelo comportamento de curto prazo do sistema, um componente de partilha de crédito, que distribui a recompensa recebida pelos classificadores e um mecanismo de evolução de regras que serve para melhorar o conhecimento atual (LANZI, 2008). Estes mecanismos e componentes interagem no que é conhecido como "ambiente" do sistema, sendo este ambiente a fonte de dados de entrada para o algoritmo LCS. Cada classificador possui uma energia chamada de *fitness*, que corresponde a capacidade de predição do classificador. (URBANOWICZ; MOORE, 2009) explica que, ao interagir com o ambiente, o LCS recebe um retorno sob a forma de recompensa numérica que impulsiona o processo de aprendizagem. Na abordagem Michigan, cada indivíduo codifica uma única regra de classificação ou uma parte da solução do problema. A Figura 2.14, mostra as etapas que formam o método LCS e estas serão detalhadas a seguir.

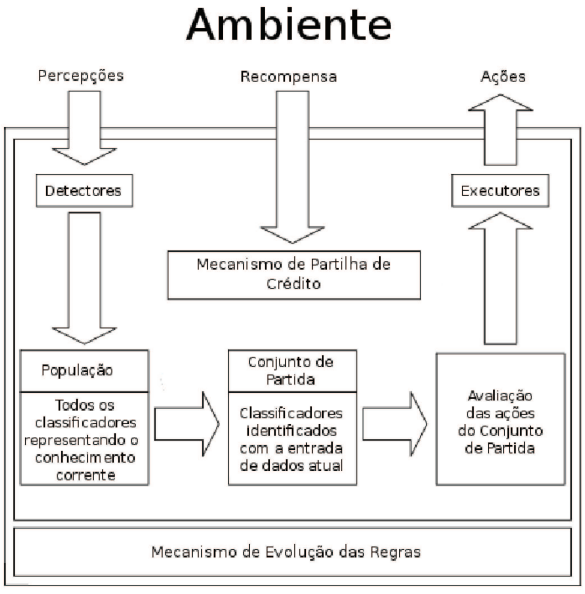

Figura 2.14: Sistemas Classificadores adaptado de (LANZI, 2008)

2.2.1 Componente de Desempenho

O componente de desempenho tem a função de analisar o sistema e avaliar o processo de aprendizagem. Um LCS deve aprender a resolver um problema, representado como um ambiente desconhecido, interagindo com ele (LANZI, 2008). A cada passo o LCS percebe o estado atual do problema através dos detectores, os quais são responsáveis pela recepção e codificação da mensagem (ou mensagens) recebidas pelo ambiente, transformando-as em uma linguagem inteligível ao Sistema Classificador; após esta etapa um conjunto de partida é constituído, contendo todos os classificadores da população, cuja condição corresponde à entrada no ambiente. Neste conjunto de partida, os classificadores são avaliados, aquele que estiver mais adaptado ao ambiente é escolhido e a sua ação é enviada para os executores. Os executores decodificam as ações propostas pelo sistema, para que as mesmas possam ser colocadas em prática. O mecanismo de partilha de crédito analisa as consequências encadeadas a partir de cada ação e determina a recompensa adequada ao classificador.

2.2.2 Mecanismo de Partilha de Crédito

O mecanismo de partilha de crédito é uma maneira de aprender o que fazer, como por meio de um mapa de ações, a fim de maximizar o valor de recompensa (SUTTON; BARTO, 1998). Os principais métodos de partilha de crédito consistem na tentativa de estimar interativamente os ensaios do agente em seu ambiente (SIGAUD; WILSON, 2007). O modelo Michigan, implementa como mecanismo de partilha de crédito o algoritmo Bucket Brigade (GEYER-SCHULZ, 1995). O algoritmo Bucket Brigade, modifica o valor da qualidade da regra através de recompensas do ambiente, modificando o *fitness* associado a cada classificador. De acordo com (GEYER-SCHULZ, 1995), o valor do *fitness* do classificador pode influenciar em dois aspectos importantes no LCS:

1. O *fitness* de um classificador afeta qual classificador pode se tornar ativo, influenciando o comportamento a curto prazo do sistema.

2. O *fitness* de um classificador pode servir como função no algoritmo genético, influenciando o comportamento a longo prazo no sistema.

Basicamente, no algoritmo de partilha de crédito, todos os classificadores que se identificaram com a mensagem do ambiente participarão de uma "competição", onde o ganhador será aquele que apresentar o maior lance efetivo, $eBid$, definido pela Equação 2.10. Ao ganhador será concedido o direito de atuar sobre o ambiente. A seguir o roteiro para o cálculo do $eBid$.

$$eBid_t = Bid_t + \alpha bid * N_t \tag{2.10}$$

$$Bid_t = k_0 * (k_1 + k_2 * Spec^{Spow}) * F_t \tag{2.11}$$

$$Spec = \frac{N - total}{N} \tag{2.12}$$

sendo:

- Bid_t: representa o Bid no instante $_t$.
- αbid representa o nível de perturbação do ruído N_t. A aplicação deste ruído permite que classificadores com menor energia também tenham chance de vencer a competição.
- N_t representa a modulação caracterizada por um ruído com distribuição normal de média 0 e variância 1.
- k_0, k_1 e k_2: são constantes positivas menores do que 1.
- $Spec$: representa o quanto específico é o classificador (definido na Equação 2.12), esta associado à proporção de variáveis inativas na regra.
- $SPow$: representa o parâmetro de controle da influência da especificidade no valor do Bid (normalmente igual a 1).
- F_t: representa o *fitness* do classificador no instante $_t$.
- N: representa o número de condições da parte antecedente do classificador.
- $total$: representa o total de condições inativas na regra.

Uma taxa de participação, representada na Equação (2.13), é cobrada de cada indivíduo que participou da "competição". O ganhador da "competição" paga também uma taxa, proporcional ao valor de seu próprio Bid_t, por ter o direito de atuar sobre o ambiente.

$$Taxa_{bid} = Bid_{tax} * Bid_t \tag{2.13}$$

sendo:

- $Taxa_{bid}$: representa a taxa de participação na competição.
- Bid_{tax}: constante aplicada sobre o Bid do classificador.
- Bid_t: valor do Bid do classificador no instante $_t$.

O classificador vencedor então atua no ambiente. De acordo com a resposta, uma recompensa ou punição é incorporada ao valor do *fitness* do classificador, conforme a Equação 2.14 que representa a fórmula geral da nova energia do classificador a cada iteração.

$$F_{t+1} = (1 - TaxaV) * F_t + R_t - Bid_t - Taxa_{bid} \tag{2.14}$$

$$TaxaV = 1 - \left(\frac{1}{2}\right)^{\frac{1}{n}} \tag{2.15}$$

sendo:

- $TaxaV$: é a taxa de vida do sistema representada pela Equação 2.15.
- F_t: é o *fitness* do classificador no instante $_t$.
- R_t: é o valor baseado na retroalimentação dada pelo ambiente, caso o classificador tenha sido vencedor da competição no instante t - 1. Observe que $Rt > 0$, em caso de recompensa (ação positiva) e $Rt < 0$ em caso de punição (ação negativa).
- Bid_t: é o Bid do classificador no instante $_t$, pago somente se o classificador foi vitorioso na competição no instante $t - 1$.
- $Taxa_{bid}$: é a taxa de participação na competição.
- n: representa o número de iterações do sistema.

2.2.3 Mecanismo de Evolução das Regras

Assegurar que cada classificador atinja o nível ideal de generalização é uma preocupação crucial no modelo LCS (SIGAUD; WILSON, 2007). Para isso, o sistema deve encontrar uma população que abranja o espaço de estado da forma mais compacta possível, sem ser prejudicial na otimização do comportamento. Os mecanismos responsáveis por essa propriedade variam de um sistema para o outro, mas todos eles dependem de exclusão e adição de novos classificadores na população.

Nos modelos atuais do LCS, esta etapa é geralmente implementada por meio de Algoritmos Genéticos (AG). Os Algoritmos Genéticos são técnicas probabilísticas de otimização global. Nos AG, a população de classificadores é submetida aos operadores genéticos de cruzamento e mutação (LINDEN, 2008). O processo evolutivo inicia a partir de uma população inicial aleatória de classificadores, que representa possíveis soluções codificadas para o problema que se deseja solucionar. Através dos operadores genéticos, os classificadores vão evoluindo e sua qualidade é calculada por uma função de aptidão que, em alguns casos, pode ser o próprio *fitness* do classificador. Assim, o AG irá guiar o processo evolutivo em busca de classificadores com índices de aptidão cada vez maiores.

Basicamente, os AG nos LCS podem ser resumidos da seguinte maneira.

1. Seleção dos pais para gerar novos indivíduos.
2. Aplicação dos operadores de cruzamento e mutação.
3. Avaliação dos novos indivíduos e substituição na população.
4. Se a condição de parada foi satisfeita finaliza, caso contrário, volta-se ao passo 1.

Os métodos de seleção nos AG utilizam o valor do *fitness* para escolher os melhores classificadores da população para reproduzirem e gerarem descendentes para as gerações seguintes. Entretanto, os métodos de seleção não podem desprezar classificadores com função de *fitness* baixa, pois, até mesmo classificadores de péssima avaliação, podem ter características genéticas que sejam favoráveis à criação de um classificador melhor adaptado para a solução de um problema em questão (LINDEN, 2008).

Os métodos mais utilizados para selecionar os melhores classificadores são a seleção por roleta e seleção por torneio.

No método de seleção por roleta, os pais são selecionados levando em conta o seu *fitness*. Quanto melhor é o classificador, mais chances ele tem de ser selecionado. Assim, dado um classificador (C_i) com *fitness* representado por f_i e a somatória dos *fitness* de

toda população representado por f, a probabilidade P_i de seleção de um classificador é calculada conforme a Equação 2.16.

$$P_i = \frac{f_i}{f} \tag{2.16}$$

Assim, cada classificador tem uma chance de ser escolhido proporcional ao seu *fitness*. Para isso, a roleta é dividida em fatias, cada uma representando um indivíduo e possuindo tamanho proporcional ao *fitness* relativo do classificador. O ato de rodar a roleta deve ser completamente aleatório, escolhendo um número entre 0 e a soma total dos valores do *fitness*. A cada rodada um classificador é selecionado. A Figura 2.15 representa um exemplo do método de proporção utilizado na seleção por roleta. De acordo com o valor do *fitness* o classificador apresenta uma probabilidade, reproduzida na figura como um gráfico circular.

Classificador (Ci)	Fitness (fi)	Probabilidade (Pi)
S1	20	0,08
S2	40	0,16
S3	50	0,20
S4	60	0,24
S5	80	0,32

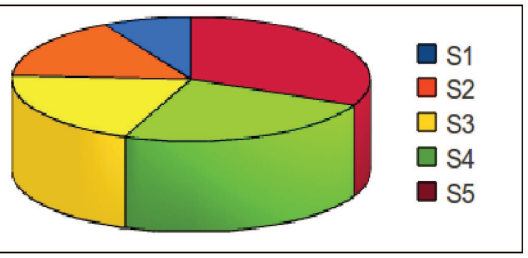

Figura 2.15: Exemplo do método de proporção utilizado na seleção por roleta

Um caso que deve ser observado quando utilizado o método da roleta é o problema do **super indivíduo**, que é definido como aquele que tem uma avaliação muito superior à média do resto da população. Neste caso, o algoritmo da roleta pode gerar problemas de convergência prematura devido à seleção frequente dos classificadores mais aptos. Neste caso, o método de Seleção por Torneio pode ser o mais indicado.

O método do torneio consiste em selecionar k classificadores da população aleatoriamente, e fazer com que eles entrem em competição direta pelo direito de seleção. Entre os k classificadores escolhidos, o que possuir o maior *fitness* é selecionado para a aplicação do operador genético.

O valor k, chamado de tamanho do torneio, pode ter influência no desempenho do algoritmo. O valor mínimo de k é igual a 2, mas não há nenhum limite teórico para o valor máximo deste parâmetro (LINDEN, 2008).

Após os classificadores serem selecionados, entram em ação os operadores genéticos. Os operadores genéticos são responsáveis em modificar os classificadores dentro da população para a produção de novos classificadores (DEJONG; SPEARS; GORDON, 1993). His-

toricamente, cruzamento e mutação tem sido os operadores genéticos mais importantes e utilizados.

O operador de cruzamento mais simples é chamado de um ponto. Este método consiste em selecionar dois pais pelo módulo de seleção, e selecionar um ponto de corte a partir do qual as informações genéticas dos pais serão trocadas. Toda informação anterior a este ponto no pai 1 é ligada à informação posterior a este ponto no pai 2 e vice-versa, formando dois novos classificadores, conforme o exemplo mostrado na Figura 2.16.

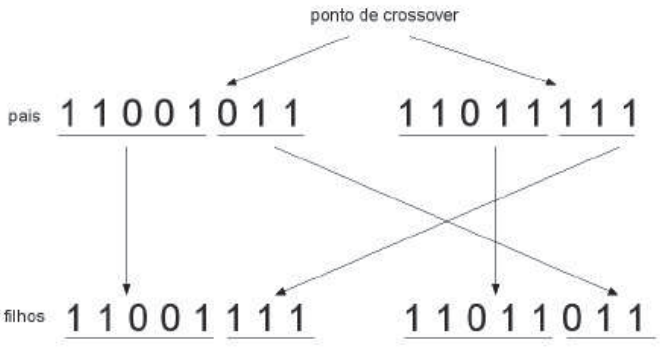

Figura 2.16: Cruzamento de um ponto

Outras variações do operador de cruzamento também são utilizadas, como por exemplo o multi pontos e uniforme. No multi pontos, ao contrário do de um ponto, vários pontos de cruzamento são escolhidos permitindo uma maior capacidade de processar esquemas. No uniforme, é possível combinar todo e qualquer esquema existente (LINDEN, 2008). Seu funcionamento pode ser descrito da seguinte forma: para cada gene é sorteado um número, zero ou um. Se o valor sorteado for igual a um, o filho número um recebe o gene da posição corrente do primeiro pai e o segundo filho o gene corrente do segundo pai. Caso o valor sorteado seja zero, as atribuições serão invertidas.

O operador de mutação é geralmente utilizado após a composição dos novos filhos. A este operador é associado uma probabilidade extremamente baixa, historicamente entre 0,5% e 1%. Um número entre 0 e 1 é sorteado e se ele for menor que a probabilidade predeterminada, o operador atua sobre o gene em questão alterando o seu valor aleatoriamente, conforme mostra a área em destaque na Figura 2.17. Em (LINDEN, 2008), o autor explica que o conceito fundamental quanto ao valor da probabilidade é que ele deve ser baixo, pois se for muito alto o AG se parecerá muito com uma técnica chamada *random walk*, na qual a solução é determinada de forma aleatória, e perderá suas características interessantes.

A cada iteração dos AG novos indivíduos são criados, e o tamanho da população é um dos principais parâmetros que afetam a robustez e a eficiência computacional do

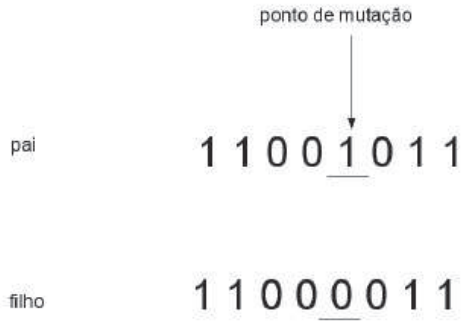

Figura 2.17: Exemplo de Mutação

algoritmo. Logo, é necessário estabelecer critérios que permitam decidir qual o tamanho da população e como os classificadores serão incluídos e excluídos da população (KOUMOUSIS; KATSARAS, 2006).

Algumas pesquisas trabalham com a substituição inteira da população a cada nova geração, outras trabalham com um método chamado de elitismo, que aproveita os melhores pais para comporem, juntamente com os novos filhos, a nova população.

O elitismo funciona da seguinte maneira: os n melhores classificadores de cada geração não devem "morrer" junto com a sua geração, mas sim passar para a próxima geração visando garantir que suas qualidades sejam preservadas (LINDEN, 2008). Esta pequena modificação no sistema garante que o desempenho do melhor classificador do AG nunca diminua no decorrer das gerações.

2.2.4 Modelo Pittsburgh

Uma outra linha dos Sistemas Classificadores é o modelo chamado de Pittsburgh. Neste modelo, cada indivíduo é um conjunto de classificadores que codifica uma solução candidata inteira e cada indivíduo geralmente consiste de um número variável de classificadores (LANZI, 2008). Conforme, (FREITAS, 2002), como cada indivíduo na abordagem Michigan codifica um único classificador, os indivíduos tendem a ser mais simples e sintaticamente mais curtos, o que tende a reduzir o tempo necessário para se calcular o *fitness* do indivíduo e simplificar os operadores genéticos. Em (ORRIOLS, 2008) o autor complementa as diferenças fundamentais entre o modelo de Michigan e o modelo de Pittsburgh.

- Representação do conhecimento: No modelo Pittsburgh, indivíduos são soluções completas para todo o problema, ou seja, cada indivíduo deve cobrir todo o espaço do recurso, ao invés de apenas uma parte da cobertura como na abordagem de

Michigan.

- Avaliação do sistema: Uma vez que cada indivíduo é um conjunto de classificadores que cobre todo o problema, não há necessidade de se avaliar a qualidade de cada um dos classificadores por conta própria, sendo que, uma única medida é suficiente para avaliar a qualidade de todo o indivíduo. Diferentes indicadores têm sido utilizados para avaliar a qualidade dos indivíduos, sendo a precisão da previsão e da generalidade dos indivíduos as mais comuns.

- Modo de aplicação do AG e da definição dos operadores genéticos: No modelo Michigan, a evolução da população ocorre por meio de ciclos dos algoritmos genéticos. Ou seja, a cada iteração, o sistema seleciona um conjunto de indivíduos, que pelas operações de cruzamento e mutação, são inseridos na população em substituição de outros indivíduos com baixa avaliação. Os operadores de cruzamento e mutação são adaptados para lidar com a representação dos indivíduos. Ao final do processo de aprendizagem, o melhor indivíduo da população é utilizado para prever a saída de novos exemplos de teste.

2.2.5 Evolução dos Sistemas Classificadores

Os Sistemas Classificadores, a partir das ideias originais desenvolvidas por John Holland (HOLLAND, 1992), vêm passando por algumas modificações em sua concepção, a fim de adaptá-los para a resolução de problemas específicos. Os primeiros sistemas classificadores trabalhavam somente com valores de atributos booleanos, mas recentemente, muitos sistemas foram adaptados para utilizarem valores nominais ou contínuos em suas soluções. Entre as principais propostas pode-se citar: o ZCS, o XCS e o ACS.

Em 1994, Stewart Wilson propôs o modelo ZCS (*Zeroth Level Classifier System*) (WILSON, 1994). O ZCS mantém o mesmo padrão do modelo original de Holland, mas foi simplificado para aumentar a inteligibilidade e o desempenho. Algumas diferenças principais do modelo ZCS em relação ao modelo original de Holland, é a criação de uma classificação heurística, uma medida estatística de distância utilizada no ciclo de performance e o uso de um operador de interferência genética.

Um ano após o desenvolvimento do ZCS em 1995, o mesmo autor, Stewart Wilson, propôs o XCS (*eXtended Classifier System*) (WILSON, 1995). O XCS tem sido atualmente o mais estudado e aplicado modelo de Sistema Classificador (KOVACS, 2002). A diferença entre o ZCS e o XCS está na definição do *fitness* do classificador, no mecanismo do Algoritmo Genético e em um mecanismo de seleção de ação mais sofisticado, que é baseado

em uma medida de precisão da predição. A Figura 2.18 mostra uma ilustração do modelo XCS.

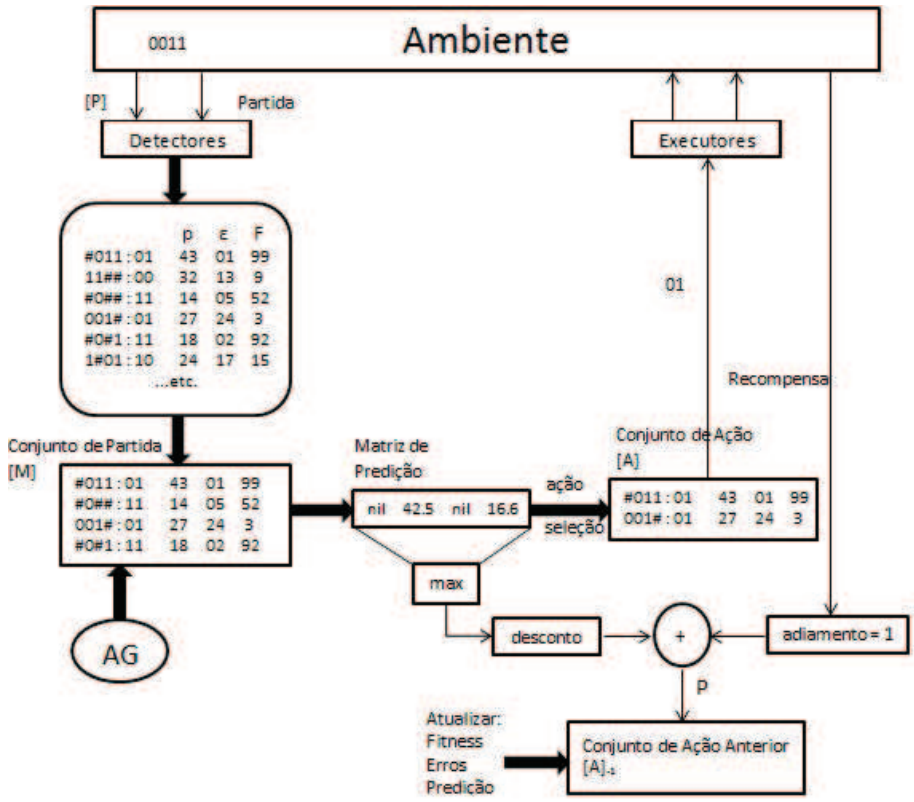

Figura 2.18: Ilustração do Modelo XCS. Adaptado de (WILSON, 1995)

No modelo original de Holland, o valor do *fitness* utilizado no mecanismo de partilha de crédito e o valor utilizado no Algoritmo Genético são os mesmos, já no XCS o *fitness* do AG é um valor distinto baseado na função inversa do erro médio da previsão do classificador, ou seja, é baseado na medida de precisão da predição.

Outra diferença com relação a versão original é que o XCS aplica o algoritmo genético somente em um subconjunto de classificadores [M] e não na população total de classificadores, conforme Figura 2.18. De acordo com (SIGAUD; WILSON, 2007), isso induz a uma competição entre classificadores que estão na mesma situação, ao invés de utilizar uma competição global. Ele seleciona dois classificadores de [M] utilizando o método de seleção por roleta e aplica os operadores de crossover e mutação com alguma probabilidade. No XCS o tamanho da população é limitado, se a população [P] contém menos do que N membros, as cópias são inseridas na população, caso contrário dois classificadores em [P] são excluídos.

Outro modelo introduzido em 1997 inicialmente por Wolfgang Stolzmann (STOLZMANN, 1998) e depois juntamente com Martin Butz e David Goldberg foi o ACS (*Anticipatory Classifier Systems*), um Sistema Classificador baseado na antecipação. De acordo

com (BUTZ; GOLDBERG; STOLZMANN, 2002), o ACS combina a estrutura do Sistema Classificador com um mecanismo cognitivo de controle comportamental antecipado.

Este mecanismo de aprendizagem como mostra a Figura 2.19, foi formulado por Joachim Hoffmann em 1992 e define, de acordo com (BUTZ; GOLDBERG; STOLZMANN, 2002), que as consequências de um determinado comportamento geralmente dependem da situação em que o comportamento é executado. Assim, a teoria da aprendizagem antecipada é definida da seguinte forma: (1) uma condição inicial S_{start}, que provoca a consequência de um comportamento ou resposta R, é acompanhada de uma antecipação desta consequência C_{ant}; (2) a antecipação da consequência C_{ant} é comparada com a consequência real C_{real}; (3) esta relação é reforçada se as antecipações forem corretas; (4) caso contrário, antecipações imprecisas ou incorretas levam a uma diferenciação das condições iniciais.

Figura 2.19: Controle de Comportamento Antecipado adaptado de (BUTZ; GOLDBERG; STOLZMANN, 2002)

Desta forma, embora o ACS aplique o mecanismo de partilha de crédito, a sua diferença em relação aos outros LCSs, é que o principal mecanismo de aprendizagem não é baseado na recompensa, mas sim, no mecanismo de aprendizagem antecipada (BUTZ; GOLDBERG; STOLZMANN, 2002). Este método é aplicado através de um processo de aprendizagem antecipada (ALP) que constroi um modelo completo, preciso e compacto do ambiente representado pela população de classificadores. A proposta do ALP é especializar cada vez mais o conjunto de classificadores da população, acelerando o processo de aprendizagem.

A partir destes modelos, uma série de outras versões já foram apresentadas, introduzindo adaptações para diversos tipos de problemas, entretanto, nenhuma proposta aplica conceitos para a solução de problemas onde os rótulos das classes são estruturados de forma hierárquica, não cobrindo portanto, soluções para alguns tipos de problemas reais como, por exemplo, a predição da função de proteínas que será explicada a seguir.

2.3 Proteínas

Proteínas são grandes moléculas que consistem em longas sequências ou cadeias de aminoácidos, também chamadas de cadeias polipeptídicas; elas se dobram em um número de diferentes estruturas e realizam quase todas as funções de uma célula em um organismo vivo (FREITAS; CARVALHO, 2007).

As proteínas podem ser representadas por quatro tipos de estruturas. Conforme a Figura 2.20, 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 uma 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 à 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, e elas podem ser idênticas ou diferentes.

As proteínas são conhecidas por terem 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.

Figura 2.20: Quatro tipos de representação das Estruturas das Proteínas

- 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 a presença de uma proteína fibrosa, o colágeno.

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

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

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, onde uma proteína X está envolvida no processo de replicação de células, até o químico, onde uma proteína X catalisa a hidrólise de certo substrato (RIGDEN; MELLO, 2002). De acordo com (CAMPBELL, 2000), 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 Predição da Função de Proteínas

Através do projeto de sequenciamento do genoma, milhares de novos genes estão sendo descobertos, incluindo aqueles que codificam novas famílias de proteínas. Muitas vezes, estas proteínas são evolutivamente conservadas, mas sua função é desconhecida (MARCOTTE, 2000).

Idealmente esta inferência deveria vir naturalmente: dada uma sequência, acha-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 proteínas diferentes, que são processos químicos pelos quais a estrutura de uma proteína assume sua configuração funcional, podem ter a mesma função. Assim, conforme (ALVES, 2010), para auxiliar neste trabalho diversos estudos em bioinformática têm sido realizados no intuito de desenvolver métodos computacionais capazes de realizar a predição destas funções, conforme mostra a Figura 2.21.

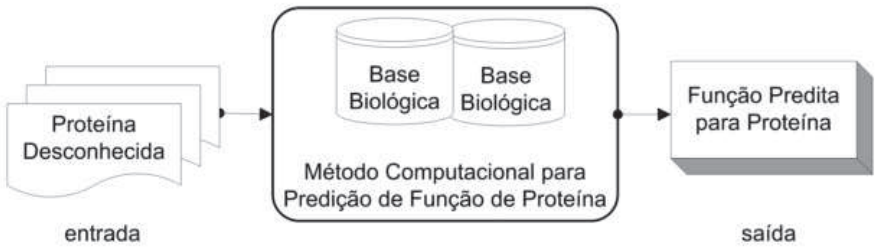

Figura 2.21: Predição de função de uma proteína desconhecida (ALVES, 2010)

De acordo com (KING; WISE; CLARE, 2004), as principais ferramentas de predição da função de proteínas como FASTA (PEARSON; LIPMAN, 1988) ou PSI-BLAST (ALTSCHUL et al., 1997), trabalham utilizando uma estatística baseada no método da homologia. Neste método, uma nova sequência de aminoácidos é comparada com outras sequências em uma base de dados para se obter uma maior similaridade. Quando encontrada, a função da sequência mais similar é inferida para a nova. Conforme (FRIEDBERG, 2006), o método de homologia é essencialmente a forma mais utilizada em ambientes computacionais de previsão. A justificativa biológica para a transferência baseada em homologia, é que se duas sequências têm um elevado grau de semelhança, então eles evoluíram de um ancestral comum e, desta forma, tem similar, se não idênticas, funções.

Entretanto, como já mencionado, nem sempre esta comparação é possível no caso das proteínas. Em (FRIEDBERG, 2006), o autor 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 que, assim como os bancos de dados de sequência estão crescendo rapidamente, a diversidade das sequências também está crescendo, e 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 vem sendo analisadas.

Baseado nesta situação, outras formas de predição da função de proteínas tem sido desenvolvidas. Em (KING; WISE; CLARE, 2004), é mostrado um método alternativo à transferência baseada em homologia chamado de Predição por Mineração de Dados. Desta forma, vários métodos de aprendizado de máquina tem sido utilizados e esses métodos se baseiam na combinação de informações de diversos tipos de elementos da proteína, como por exemplo:

- **Atributos de Aminoácidos -** Tamanho, pI, proporção de singletons e pares de resíduos, códons usados, entre outros, podem ser pistas sobre a função da proteína.
- **Estrutura -** A predição da estrutura também pode ser uma forte pista e informações sobre dobras e conexões podem auxiliar na predição da função da proteína.
- **Fusão de Genes -** Se dois domínios da proteína são encontrados fundidos em outro organismo, então é provável que estes dois domínios possuem funções relacionadas.
- **Proximidade de Cromossomos -** Em muitos casos, os genes que estão juntos em um cromossomo possuem funções relacionadas.
- **Padrões Filogenéticos -** Neste caso, o padrão das espécies em que são encontradas sequências homólogas pode ser um informativo sobre a função.

O uso da predição da função de proteínas por mineração de dados vem crescendo nos últimos anos, e os resultados iniciais tem sido bastante expressivos. Outro fato que vem auxiliando na melhoria das predições é o uso de ontologias, que fornecem um vocabulário comum e que facilita a comunicação entre os pesquisadores da área de biologia e os diversos usos ou propósitos destas informações para o processamento das mesmas (ALVES, 2010). As principais ontologias são destacadas a seguir.

2.3.2 Ontologia Gênica

A Ontologia Gênica (GO) (ASHBURNER et al., 2000), serve atualmente como abordagem dominante em ferramentas de anotação funcional. De acordo com (ASHBURNER et al., 2000), a GO prevê uma ontologia de termos definidos que representam as propriedades do gene, ela forma um vocabulário de termos consistente e estruturado para descrever domínios chave da biologia molecular, incluindo atributos e produtos gênicos, assim como

sequências biológicas. Estes termos podem ser anotados para sequências, genes ou produtos gênicos armazenados em bases de dados biológicos. A GO descreve atributos de produtos gênicos em três domínios disjuntos da biologia molecular:

- 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.

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

A Figura 2.22 mostra uma visão das funções do componente celular definido na Ontologia Gênica para um tipo específico de peixe de água doce. O tamanho do círculo representa o número relativo de genes anotados para cada nó pai.

A Versão 1.3451 do GO, possui 38048 termos definidos. Destes, 23878 são relacionados a processos biológicos, 9453 a funções moleculares e 3045 a componentes celulares.

2.3.3 FunCat

Outro modelo de descrição funcional que tem sido utilizado nos trabalhos de classificação de proteínas é o FunCat (MIPS Functional Catalogue) (RUEPP et al., 2004). O FunCat utiliza uma estrutura hierárquica em formato de árvore, independente, flexível e escalável, possibilitando a descrição funcional de proteínas a partir de qualquer organismo. O sistema de anotações FunCat consiste de 28 categorias principais que abrangem características gerais como transporte celular, metabolismo e regulamentação da atividade da proteína. O FunCat tem sido aplicado para a anotação manual e automática de procariontes, fungos, plantas e animais.

No total, a versão 2.1 do FunCat inclui 1362 categorias funcionais. Os níveis de categorias do FunCat são separadas por pontos; por exemplo, 01 metabolismo, é um

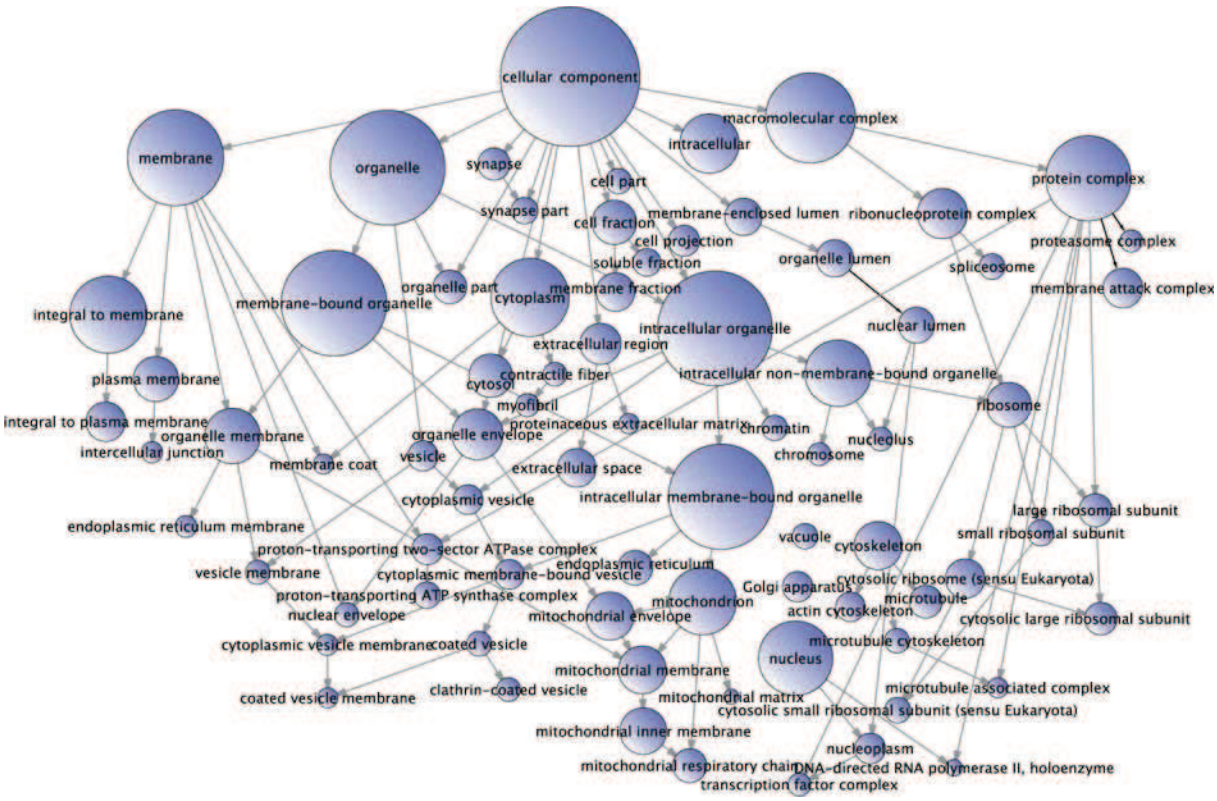

Figura 2.22: Componente celular específico do GO em DAG (SALZBURGER et al., 2008)

representante do mais alto nível, e 01.01.03.02.01 biossíntese do glutamato, pertence ao nível mais específico de FunCat. Para a maioria das proteínas, a função celular não pode ser completamente descrita com uma única categoria funcional, mas tem que ser considerado como a soma de propriedades diferentes.

2.4 Considerações Finais do Capítulo

Neste capítulo foram descritos os conceitos e as tecnologias que serão utilizadas neste trabalho. O modelo proposto nesta tese visa predizer a função de proteínas através de um sistema classificador hierárquico global multirrótulo, utilizando como informações, base de dados estruturadas hierarquicamente.

Capítulo 3

Classificação Hierárquica Multirrótulo da Função de Proteínas

3.1 Considerações Iniciais

Este capítulo tem o intuito de trazer o estado da arte dos trabalhos relacionados à classificação hierárquica multirrótulo da função de proteínas. Como já citado no capítulo anterior, a classificação da função de proteínas é um problema de classificação hierárquica. Isso quer dizer que, tipicamente, as classes são organizadas de duas formas: como uma árvore, onde cada classe tem ao menos uma classe pai, ou em forma de um grafo acíclico direcionado (DAG), onde cada classe pode ter mais do que uma classe pai. A maneira com que os dados são organizados depende do tipo de descrição funcional adotado, sendo que, para os modelos estruturados em árvore pode-se citar o FunCat e para os modelos estruturados em DAG a Ontologia Gênica (GO). Outro fator que diferencia os algoritmos é o tipo de abordagem que pode ser plana, local ou global e também a forma com que o algoritmo trabalha a questão de multirrótulo, como já explicado anteriormente. Com isso, a Seção 3.2, tem a proposta de mostrar uma visão de trabalhos que utilizam algoritmos de classificação para base de dados do tipo árvore e na seção 3.3 uma visão de trabalhos que utilizam algoritmos de classificação para bases de dados do tipo do DAG.

3.2 Classificação Hierárquica Multirrótulo em Árvores

Os principais trabalhos que utilizam métodos de classificação hierárquica multirrótulo para a função de proteínas estruturadas do tipo árvore são: (CLARE; KING, 2001), (CLARE; KING, 2003), (VALENTINI, 2009), (CERRI; CARVALHO, 2010a) e (CERRI; CARVALHO, 2010b).

Em (CLARE; KING, 2001), os autores utilizam o conhecido 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 previsão das classes é feita em cada nível da hierarquia, independente dos outros níveis, numa sequência *top-down*. Os autores otimizaram 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 utilizado uma fórmula de probabilidade para a escolha das possíveis classes de cada nó.

Este trabalho foi aprimorado em (CLARE; KING, 2003), onde 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 portanto a fórmula foi ponderada para valorizar estas classes.

Em (VALENTINI, 2009) o autor se baseia numa proposta local onde cada classificador base é associado a uma classe da hierarquia. Este trabalho foi aplicado em bases de dados do modelo FunCat. Os classificadores base são treinados para se tornarem especialistas na classificação de apenas uma classe na hierarquia. Para isso, cada classificador estima a probabilidade local $\hat{p}_i(x)$ que uma instância x possa pertencer a classe c_i. Esses classificadores trocam informações para que seja obtida uma classificação final baseada na combinação de suas predições. As informações são trocadas em um fluxo bidirecional, no qual predições positivas para uma classe influenciam suas classes ancestrais e predições negativas influenciam suas classes descendentes.

O trabalho de (CERRI; CARVALHO, 2010a) apresenta um método de classificação não hierárquica multirrótulo adaptado para problemas hierárquicos multirrótulo. O método utiliza a abordagem de classificação local e foi implementado utilizando Redes Neurais Artificiais e base de dados organizados conforme o modelo FunCat. O método chamado de *HMC-Label-Powerset (HMC-LP)* utiliza uma combinação de classes para transformar o problema hierárquico multirrótulo em um problema hierárquico simples-rótulo. No método HMC-LP o processo de combinação de classes é aplicado em todos os níveis da hierarquia. Assim, para cada instância, o método combina todas as classes atribuídas a ela num específico nível, em uma nova e única classe.

Por exemplo, dado uma instância pertencente a classe $A.D$ e $A.E$, e outra instância pertencente as classes $B.F$, $B.G$, $C.H$ e $C.I$, onde $A.D$, $A.E$, $B.F$, $B.G$, $C.H$ e $C.I$ são estruturadas hierarquicamente de tal modo que $A \leq_h D$, $A \leq_h E$, $B \leq_h F$, $B \leq_h G$, $C \leq_h$

H e $C \leq_h I$ com A, B e C pertencente ao primeiro nível e D, E, F, G, H, e I pertencente ao segundo nível. O resultado da combinação de classes para as duas instâncias seria uma nova estrutura hierárquica $C_A.C_{DE}$ e $C_{BC}.C_{FGHI}$, respectivamente, como mostra a Figura 3.1. Durante a fase de treinamento um ou mais classificadores multirrótulo são utilizados em cada nível da hierarquia. De acordo com os autores, a desvantagem deste método é que a combinação de classes pode aumentar consideravelmente o número de classes envolvidas no problema.

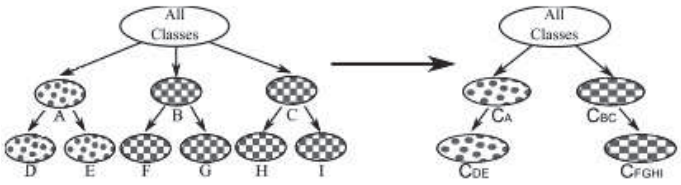

Figura 3.1: Método HMC-Label-Powerset proposto em (CERRI; CARVALHO, 2010a)

Os mesmos autores em (CERRI; CARVALHO, 2010b) apresentam uma outra forma de adaptação para problemas hierárquicos multirrótulo. O método chamado de *HMC-Cross-Training (HMC-CT)* utiliza um processo de decomposição onde todas as instâncias multirrótulo são decompostas em instâncias simples-rótulo. Neste processo, para cada instância, cada classe possível é considerada como uma classe positiva e cada instância multirrótulo é considerada mais do que uma vez na fase de treinamento. Por exemplo, considerando uma instância y_i que pertence a duas classes A e B, o processo de treinamento ocorre duas vezes, uma considerando os exemplos pertencentes a classe A e outra considerando os exemplos pertencentes a classe B. Este processo de decomposição é aplicado em todos os níveis da hierarquia e uma estratégia local utilizando Máquinas de Vetores de Suporte é aplicada durante as fases de treinamento e teste.

Um resumo com relação ao tipo de abordagem e os modelos de algoritmos utilizados nestes trabalhos está demonstrado na Tabela 3.1.

Além destes trabalhos, outros métodos, apesar de não utilizarem bases multirrótulo, vem contribuindo com soluções para a predição da função de proteínas. Em (HOLDEN; FREITAS, 2005) e (HOLDEN; FREITAS, 2006) os autores propõem um algoritmo híbrido utilizando os métodos de Otimização por Enxame de Partículas e Colônia de Formigas chamado de PSO/ACO, para a predição de enzimas organizadas em EC Code e bases derivadas do GPCR. Neste algoritmo, os autores utilizam um classificador local para a criação do conjunto de treinamento e para a classificação dos exemplos no conjunto de teste.

Tabela 3.1: Trabalhos de classificação hierárquica multirrótulo em estruturas do tipo árvore.

Trabalhos	**Abordagem**	**Algoritmo**
(CLARE; KING, 2001)	Classificação Plana	Árvore de Decisão
(CLARE; KING, 2003)	Classificação Global	Árvore de Decisão
(VALENTINI, 2009)	Classificação Local	Combinação de Classificadores
(CERRI; CARVALHO, 2010a)	Classificação Local	Redes Neurais Artificiais
(CERRI; CARVALHO, 2010b)	Classificação Local	Máquinas de Vetores de Suporte

Durante o desenvolvimento do modelo, o PSO/ACO é executado uma vez para cada nó (não-folha) interno da hierarquia da árvore. De acordo com (FREITAS; CARVALHO, 2007), no PSO/ACO, para cada nó interno, um conjunto de regras é descoberto para todas as classes associadas com os nós filhos. A hierarquia da classe é utilizada para selecionar um conjunto específico de exemplos positivos e negativos em cada execução do algoritmo, selecionando somente os exemplos diretamente relevantes.

Esta abordagem produz um conjunto de regras hierárquicas, onde cada nó interno da hierarquia está associado ao seu conjunto correspondente de regras. Ao classificar um novo exemplo no conjunto de teste, o exemplo é primeiro classificado pelo conjunto de regras associado com o nó raiz. Em seguida, ele é classificado pelo conjunto de regras associadas ao nó de primeiro nível, cuja classe era prevista pela regra estabelecida na raiz e assim sucessivamente, até o exemplo chegar a um nó folha onde lhe é atribuída a classe. Este modelo tem a desvantagem de que, se um exemplo é erroneamente classificado no primeiro nível, então claramente o exemplo também será classificado erroneamente em todos os níveis abaixo.

Em (SILLA; FREITAS, 2009), os autores propõem um modelo global de classificação hierárquica baseado no algoritmo Naïve Bayes e utiliza bases organizadas nos modelos EC Code e GPCR. No sistema, um modelo único de classificação é construído a partir do conjunto de treinamento, levando em conta a hierarquia de classes como um todo durante uma única execução do algoritmo de classificação. Quando utilizado durante a fase de teste, cada exemplo de teste é classificado pelo modelo induzido, um processo que pode atribuir classes potencialmente menores em todos os níveis da hierarquia para o exemplo de teste. Para implementar a fase de testes, foi utilizada a estratégia de predição *top-down*, utilizando no contexto do problema uma busca baseada no método de predição do nó folha não obrigatório.

3.3 Classificação Hierárquica Multirrótulo em DAG

Os principais trabalhos que utilizam métodos de classificação hierárquica para a função de proteínas estruturadas do tipo DAG são: (JENSEN L. J. GUPTA; HENRIK; BRUNAK, 2003), (LAGREID et al., 2003), (TU et al., 2004), (BARUTCUOGLU; SCHAPIRE; TROYANSKAYA, 2006), (VENS et al., 2008), (ALVES; DELGADO; FREITAS, 2008), (OTERO; FREITAS; JOHNSON, 2009) e (OTERO; FREITAS; JOHNSON, 2010), sendo que todas as propostas utilizam bases funcionais da Ontologia Gênica (GO).

Em (JENSEN L. J. GUPTA; HENRIK; BRUNAK, 2003), os autores propõe um modelo de classificação baseado em Redes Neurais Artificiais. O algoritmo utiliza uma abordagem plana e cada classe da GO é treinada individualmente pelo algoritmo. Toda relação pai-filho é ignorada no algoritmo, pois na fase de treinamento apenas uma classe é tratada como positiva e as demais como negativas. De maneira geral, para treinar cada classe da GO é utilizada uma Rede Neural Artificial multicamada com uma camada escondida de neurônios. Várias combinações de atributos são utilizadas para treinar a rede, cuja saída, consiste em determinar se as instâncias pertencem (valor 1) ou não (valor 0) a classe predita. Da mesma forma, foram feitas combinações de cinco redes neurais, com um único atributo de entrada, para o treinamento dos classificadores.

Neste trabalho, é visível a sua limitação com relação ao número de classes possíveis a serem preditas. Inicialmente, os autores tentaram prever 347 classes GO, mas esse número foi reduzido significativamente em uma etapa de "pré-processamento" do algoritmo. Primeiro, a maioria das classes foram descartadas porque elas não podiam ser previstas com uma precisão razoável, ou porque representavam classes triviais cuja previsão não era interessante. Com isso, o número de classes a serem previstas se reduziu a 26. Em seguida, mais um conjunto de classes foi descartada para evitar redundância com relação a outras classes, o que deixou somente 14 classes para serem previstas pelo modelo.

Para utilizar o método para predizer a função de novas sequências, todas as características da nova sequência são calculadas e codificadas da mesma forma que os exemplos de treinamento. Em seguida, são repassadas a cada uma das cinco redes neurais correspondentes a cada classe GO para serem analisadas, o modelo que apresentar a melhor medida de confiança em seu resultado é que será utilizado para predizer a classe a qual pertence a sequência. Para a avaliação de desempenho do modelo, os autores usaram as medidas de revocação e a taxa de falsos positivos.

No trabalho de (LAGREID et al., 2003), os autores utilizam uma combinação do algoritmo de indução de regras baseado na teoria *Rough Set* e do Algoritmo Genético

para criar um conjunto de regras do tipo *SE-ENTÃO* e predizer as classes da GO. Este modelo também segue a abordagem de classificação plana e a hierarquia das classes foi utilizada somente na etapa de pré-processamento do conjunto de treinamento. Nesta etapa, o conjunto de treinamento inicialmente constituído de 497 genes, possuia 284 genes conhecidos, ou seja, genes que tinham informações sobre o seu processo biológico definidos, o restante, 213, eram desconhecidos e foram descartados. Para estes 284 genes, os autores definiram 23 grandes grupos, sendo que 11 não pertenciam a nenhum dos grupos e foram descartados. Os 273 genes restantes, foram agrupados nos 23 grupos e deram origem a 549 instâncias de treinamento devido a alguns genes terem mais de um pai e em 167 genes foram encontrados mais de um processo biológico definido.

Com isso, na etapa de treinamento, um conjunto de regras foi descoberto para cada uma das 23 classes, gerando um modelo de classificação para cada uma delas.

Outro trabalho que utiliza classificação plana no seu modelo é o apresentado em (TU et al., 2004). Este modelo de predição também é baseado no algoritmo de Redes Neurais Artificiais. Entretanto, diferentemente dos trabalhos apresentados em (JENSEN L. J. GUPTA; HENRIK; BRUNAK, 2003) e (LAGREID et al., 2003), nesta proposta os autores levaram em consideração a relação pai-filho das classes na hierarquia na etapa de treinamento do modelo. De acordo com (FREITAS; CARVALHO, 2007), neste trabalho, um conjunto com as classes ancestrais da GO e suas classes descendentes é denominado pelos autores como um espaço de classificação. A cada espaço de classificação, uma Rede Neural Artificial plana foi treinada para predizer a classe filha para um exemplo que já havia sido predito como pertencente à classe pai. O objetivo desta técnica foi fazer uma predição adicional que fosse capaz de fazer anotações para as classes mais específicas a partir das anotações conhecidas, para suas classes ancestrais aparecerem antes na hierarquia.

No trabalho de (BARUTCUOGLU; SCHAPIRE; TROYANSKAYA, 2006), uma hierarquia de classificadores SVMs locais é utilizada para predição de processos biológicos da GO. Os classificadores são treinados para cada classe separadamente e tem suas predições combinadas utilizando aprendizado Bayesiano para obter o mais provável conjunto consistente de predições. O método Bayesiano permite que todos os níveis de classificadores sejam influenciados um com os outros. Assim, as predições feitas ficam consistentes com os relacionamentos hierárquicos.

Em (VENS et al., 2008), os autores desenvolveram um modelo de classificação hierárquica baseado no método de árvore de decisão para estruturas do tipo DAG. Neste 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) é um sistema que utiliza o método de classificação global, levando em conta todas as classes na predição através de uma única árvore de decisão.

O modelo é definido no *framework* Predictive Clustering Trees (PCT), que define uma árvore de decisão como uma hierarquia de grupos, onde 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 TDIDT (*Top-down induction of decision trees*). A ideia geral, consiste em particionar recursivamente o conjunto de dados de forma a reduzir a variância *intra-cluster*, 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 multirrótulo, as classes dos exemplos são representadas através de vetores com valores booleanos (o *i-ésimo* valor do vetor é 1 se o exemplo pertencer a classe c_i, caso contrário será 0). A Figura 3.2 mostra um exemplo desta operação, onde o subconjunto de classes $(1, 2, 2.2)$ indicadas em negrito na hierarquia, é representado como um vetor $v_i = [1, 1, 0, 1, 0]$.

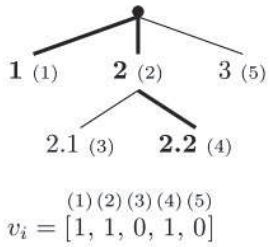

Figura 3.2: Exemplo da representação de classes (VENS et al., 2008)

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 $\bar{v}$. Como mostra a Equação 3.1 retirada de (VENS et al., 2008).

$$Var(S) = \frac{\sum_i d(v_i, \bar{v})^2}{|S|} \tag{3.1}$$

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 3.2.

$$d(v_1, v_2) = \sqrt{\sum_i w(c_i) * (v_{1,i} - v_{2,i})^2} \tag{3.2}$$

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 as 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 conforme 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 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 onde uma classe pode ter múltiplos pais, pela agregação de $w(pai(c))$ no cálculo do peso.

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. Assim, ao invés calcular a probabilidade condicional $P(c|pai(c))$ é calculado a probabilidade condicional $P(c|pai_j(c))$. Para fazer a predição, o produto da regra $P(c) = P(c|pai_j(c)) * P(pai(c))$, pode ser aplicado para cada classe pai $pai(c)$.

A Figura 3.3, 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 $w(c) = w_0 \sum_j w(pai_j(c))$ e $w_0 = 0.75$. Em (b), através da árvore construída pelo CLUS-HSC a probabilidade de predição de uma classe (c) é dada pela fórmula $P(c) = min_j P(c|pai_j(c)) * P(pai_j(c))$.

Os testes neste trabalho, foram feitos com vinte e quatro base 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 no forma de árvores e na descrição funcional baseada no GO, que organiza os dados na forma de DAGs. Os resultados foram avaliados através das medidas de precisão e revocação e pela curva AUC. 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

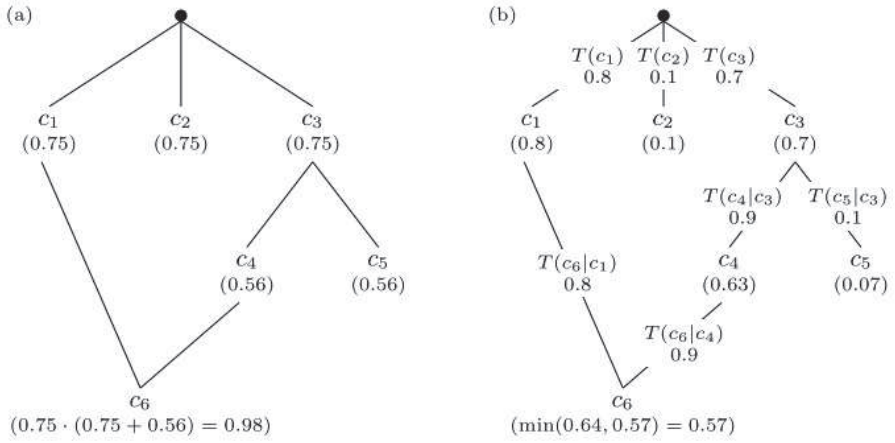

Figura 3.3: Exemplo da aplicação da hierarquia de classes estruturada em DAG desenvolvido por (VENS et al., 2008)

usado o HMC é muito menor.

Em (ALVES; DELGADO; FREITAS, 2008), autores construíram um modelo de classificação hierárquica multirrótulo 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 (instância) 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; DELGADO; FREITAS, 2008), a fase de treinamento do MHC-AIS é realizada por dois procedimento 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 descobertas. 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 revocação 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 3.3.

$$fit(c^i_{k*}) = max[fit(c^i_{k*}), fit(c^i_k)], c^i_{k*} \in Ancestral(c^i_k) \tag{3.3}$$

Isso quer dizer que, 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 autores avaliaram os dois modelos através das medidas de revocação, 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.

Em (OTERO; FREITAS; JOHNSON, 2009), os autores propõem uma solução global para a classificação da função de proteínas utilizando Colônia de Formigas. O sistema chamado de hAnt-Miner, utiliza o modelo GO e, para a avaliação do sistema, os autores criaram cinco conjunto de dados envolvendo a função de proteínas *Ion Channel*. O problema alvo do método proposto hAnt-Miner é a descoberta de regras de classificação hierárquica na forma *SE-ENTÃO*. O hAnt-Miner divide o processo de construção de regras em duas diferentes colônias de formigas, uma colônia para criar antecedentes da regra (SE) e uma colônia para criar os consequentes da regra (ENTÃO), que trabalham de forma cooperativa. A medida utilizada para avaliar as regras construídas é uma combinação de medidas Revocação Hieráquica e Precisão Hierárquica, e leva em conta o fato de que um exemplo não pertence somente a sua classe mais específica, mas também a todas as classes ancestrais de acordo com a hierarquia de classe (exceto a classe raiz, pois todos os exemplos trivialmente pertencem ao rótulo da classe raiz por padrão).

Os mesmos autores em um trabalho mais recente (OTERO; FREITAS; JOHNSON, 2010), identificaram algumas limitações no método hAnt-Miner, e desenvolveram um outro método chamado hmAnt-Miner (*Hierarchical Multi-Label Classification Ant-Miner*). A principal diferença deste método é a capacidade do consequente da regra poder prever mais do que uma classe ao mesmo tempo. Desta forma, o consequente da regra é calculado utilizando o seguinte procedimento determinístico: dado um conjunto de exemplos S_r cobertos pela regra r, o consequente da regra será um vetor de tamanho m, sendo m o número de classes da hierarquia. O valor de cada *i-ésimo* componente do vetor para a regra r é dado então pela Equação 3.4.

$$consequente_{r,i} = \frac{|S_r \& classe_i|}{|S_r|} \tag{3.4}$$

Sendo, $|S_r \& classe_i|$ o número de exemplos cobertos pela regra r que pertencem a *i-ésima* classe da hierarquia de classes $classe_i$. Isso quer dizer que, o consequente da regra é um vetor onde cada *i-ésimo* componente é a proporção de exemplos cobertos que pertencem a *i-ésima* classe.

Um resumo com relação ao tipo de abordagem e os modelos de algoritmos utilizados nestes trabalhos está demonstrado na Tabela 3.2.

Tabela 3.2: Trabalhos de classificação hierárquica multirrótulo em estruturas do tipo DAG.

Trabalhos	**Abordagem**	**Algoritmo**
(JENSEN L. J. GUPTA; HENRIK; BRUNAK, 2003)	Classificação Plana	Redes Neurais Artificiais
(LAGREID et al., 2003)	Classificação Plana	Indução de regras baseado na teoria *Rough Set* e Algoritmo Genético
(TU et al., 2004)	Classificação Plana	Redes Neurais Artificiais
(BARUTCUOGLU; SCHAPIRE; TROYANSKAYA, 2006)	Classificação Local	Máquinas de Vetores de Suporte
(VENS et al., 2008)	Classificação Plana, Local e Global	Árvore de Decisão
(ALVES; DELGADO; FREITAS, 2008)	Classificação Global	Sistemas Imunológicos Artificiais
(OTERO; FREITAS; JOHNSON, 2009)	Classificação Global	Colônia de Formigas
(OTERO; FREITAS; JOHNSON, 2010)	Classificação Global	Colônia de Formigas

3.4 Considerações Finais do Capítulo

Em um estudo realizado por (FREITAS; CARVALHO, 2007), os autores apresentam uma revisão de uma série de trabalhos relacionados com a previsão da função da proteína e concluem alguns aspectos interessantes com relação a este problema. Primeiro, os autores consideram o fato que, o custo de um erro de classificação, tende a variar significativamente entre os diferentes níveis da hierarquia. Isso quer dizer que, um erro que ocorre nos níveis mais baixos, é menos prejudicial dos que acorrem nos níveis superiores, sendo um tema pouco explorado na literatura. Outro fato interessante, discutido pelos autores, é que alguns trabalhos que utilizam como base os dados da GO, consideram uma parte extremamente pequena das classes disponíveis desta estrutura, sendo essas classes, muitas vezes escolhidas de acordo com a intuição dos autores.

Outros autores mostram em (SECKER et al., 2007), uma série de comparações entre algoritmos de classificação plana, local e global para predição hierárquica da função de proteínas e, de acordo com os resultados, concluem que o uso de classificadores globais deva ser uma proposta mais eficiente do que os de classificadores locais ou planos.

Aliado a estes trabalhos e considerando o estudo apresentado, é possível observar particularidades específicas em cada um dos modelos. Os trabalhos de (CLARE; KING, 2001), (JENSEN L. J. GUPTA; HENRIK; BRUNAK, 2003), (LAGREID et al., 2003), (TU et al., 2004), (BARUTCUOGLU; SCHAPIRE; TROYANSKAYA, 2006), (VALENTINI, 2009), (CERRI; CARVALHO, 2010a) e (CERRI; CARVALHO, 2010b) apresentam soluções de classificação plana, ou seja, não levam em consideração a hierarquia das classes no desenvolvimento dos modelos de predição. Como já visto, este tipo de abordagem tem uma série de limitações que levam a questionamentos os resultados apresentados pelos autores.

O trabalho de (CLARE; KING, 2003) apesar de apresentar uma proposta de classificação global, suportam somente bases estruturadas em árvores, não sendo capaz, portanto, de predizer as proteínas classificadas pelo método da Ontologia Gênica.

Os trabalhos de (VENS et al., 2008), (ALVES; DELGADO; FREITAS, 2008), (OTERO; FREITAS; JOHNSON, 2009) e (OTERO; FREITAS; JOHNSON, 2010) que utilizam classificação global e suportam estruturas baseadas em DAG, são propostas que trazem funcionalidade interessantes, que até então não haviam sido mencionadas. Por exemplo, as propostas apresentadas em (VENS et al., 2008), (ALVES; DELGADO; FREITAS, 2008) e (OTERO; FREITAS; JOHNSON, 2010), levam em consideração a possibilidade de predição multirrótulo. Nos trabalhos de (OTERO; FREITAS; JOHNSON, 2009) e (OTERO; FREITAS; JOHNSON, 2010) os autores utilizam medidas de avaliação que levam em consideração a hierarquia das classes. Além disso, em (ALVES; DELGADO; FREITAS, 2008) os autores utilizam a hierarquia das classes para o cálculo da energia da regra.

Com isso, é possível concluir que este últimos trabalhos apesar de já trazerem resultado melhores que os anteriores, podem ainda, por serem propostas mais recentes, serem aperfeiçoados de outras formas.

Capítulo 4

Sistema HLCS

Neste capítulo apresentam-se todas as etapas de desenvolvimento do algoritmo de classificação global hierárquica multirrótulo para a predição da função de proteínas proposto nesta tese. Na Seção 4.1 são feitas algumas considerações iniciais sobre o algoritmo proposto. A Seção 4.2 faz uma descrição geral do algoritmo e detalha cada um dos componentes que fazem parte do algoritmo. A Seção 4.3 define a estrutura da versão HLCS-Local do modelo. A Seção 4.4 define a estrutura da versão HLCS-Tree do modelo. A Seção 4.5 define a estrutura da versão HLCS-DAG do modelo. A Seção 4.6 define a versão do final do modelo chamada de HLCS-Multi. Por fim, a Seção 4.7 mostra uma simulação passo a passo da execução do modelo HLCS-Multi .

4.1 Considerações Iniciais

Como comentado anteriormente, a predição da função de proteínas possui certas particularidades que devem ser consideradas no desenvolvimento de uma solução de classificação. Primeiramente, a predição da função de proteínas é conhecido como um problema de classificação hierárquica, sua estrutura por ser definida como uma árvore ou um grafo acíclico dirigido (DAG) sendo importante que esta seja preservada na execução do algoritmo. A maioria dos algoritmos de classificação desenvolvidos para suportar esse tipo de estrutura, normalmente, não avalia o modelo hierárquico como um todo (abordagem global), o que pode alterar o resultado preditivo das instâncias. Segundo, o uso de ontologias em predição de funções de proteínas tem sido cada vez mais utilizado, sendo a FunCat e a Ontologia Gênica (GO) as mais utilizadas pelos pesquisadores da área. Os termos da GO são estruturados na forma de um DAG multirrótulo, no qual um termo "filho" pode estar conectado a um ou mais termos "pais", tornando a solução ainda mais complexa. E outra questão bastante importante, é com relação aos resultados obtidos

pelo algoritmo, que devem ser compreensíveis o suficiente para que se possa expressar facilmente os conhecimentos alcançados.

Com isso, o algoritmo Hierarchical Learning Classifier System - Multilabel (HLCS-Multi) proposto neste trabalho, apresenta uma solução global hierárquica multirrótulo para a classificação da função de proteínas respeitando a hierarquia das classes em todo o desenvolvimento do modelo. É desenvolvido especificamente para trabalhar com bases hierárquicas de proteínas e utiliza como modelo de desenvolvimento o Sistema Classificador (LCS), que é um método que gera seus resultados em um conjunto de regras no formato *SE-ENTÃO*, que são representações, de acordo com (FREITAS; WIESER; APWEILER, 2010) mais compreensíveis do que outros modelos como redes neurais, máquinas de vetor de suporte entre outros.

O LCS tem sido utilizado com sucesso em diversas áreas como na robótica (HURST; BULL, 2004), ambientes para simulação de navegação (WILSON, 1995),(LANZI; LOIACONO, 2007), aproximação de funções (HAMZEH; RAHMANI, 2007), mineração de dados (ORRIOLS-PUIG; CASILLAS; BERNADó-MANSILLA, 2007), entre outros. No entanto, para problemas de classificação hierárquica, até então, não se tem conhecimento da utilização da abordagem LCS.

Para se chegar no modelo final desta Tese, o HLCS-Multi passou por uma série de evoluções que contribuiu ainda mais para o aprendizado e amadurecimento da proposta. A primeira versão desenvolvida foi o HLCS-Local, esta abordagem apresenta uma solução local por nó para trabalhar com bases hierárquicas. A segunda versão foi o HLCS-Tree, que utiliza uma abordagem global para bases hierárquicas estruturadas em formato de árvores. A terceira e última versão antes do HLCS-Multi foi o HLCS-DAG, que também utiliza uma abordagem global mas com a capacidade de trabalhar com bases hierárquicas estruturadas no formato DAG. Todos estes modelos foram derivados do modelo principal Hierarchical Learning Classifier System (HLCS).

A seguir será dada uma descrição geral do HLCS e uma visão de cada abordagem, destacando os pontos principais da evolução do algoritmo até se chegar no modelo HLCS-Multi.

4.2 Descrição Geral do Algoritmo Proposto

O HLCS é um algoritmo baseado em regras chamadas de classificadores. A primeira função do sistema é criar uma população inicial de classificadores, através de uma análise feita nas instâncias da base de treinamento de tamanho N, que representa a fonte de dados do sistema. A definição do tamanho da população de classificadores ($TamPop$)

varia de acordo com o modelo HLCS, sendo que na população inicial só são adicionados classificadores diferentes. O conjunto de todos os classificadores constitui o modelo de predição. Cada classificador C_i $(0 < i \leq TamPop)$ do HLCS é composto por: um conjunto n de condições (onde n=número de atributos da instância de treinamento), o valor da classe e o valor da medida de qualidade do classificador, como mostra a Equação 4.1.

$$C_i = (([Cond_0]E...E[Cond_n])(VClasse)(VQualidade)) \tag{4.1}$$

Cada condição possui três parâmetros: OP, VL, A/I, sendo:

- OP: operador de relação (= ou !=)
- VL: valor da condição.
- A/I: opção de ativa (A) ou inativa (I) da condição, que significa se a condição será utilizada na classificação ou não.

Para formar cada classificador da população inicial de classificadores, o HLCS escolhe aleatoriamente uma instância $Instancia_i$ $(0 < i \leq N)$ da base de treinamento como modelo. Para cada atributo da instância de treinamento é criada uma condição no classificador. No início todas as condições começam com o operador de relação (OP) "=". O valor da condição (VL) recebe o valor do atributo da instância de treinamento e aleatoriamente, de acordo com a probabilidade definida nas configurações iniciais do HLCS, é decidido se a condição ficará ativa (A) ou inativa (I).

O HLCS executa somente em bases com atributos nominais, caso a base possua atributos contínuos, o HLCS utiliza do método de discretização, que tem a função de transformar os atributos contínuos das bases de dados em intervalos discretos representados por atributos nominais.

Cada classificador criado representa em sua estrutura, a definição de uma regra que será utilizada para a predição. Após a definição das condições do classificador é o momento de determinar a classe de predição que será atribuída ao classificador. Para isso, o classificador é analisado com todas as instâncias da base de treinamento, cada instância coberta pelo classificador tem sua classe pontuada. Ao final da análise, a classe mais pontuada é então escolhida como a classe do classificador.

A última etapa da criação da população inicial é o cálculo da qualidade do classificador. Esta etapa difere para cada modelo HLCS é será explicada nas próximas seções.

Este processo é então repetido de acordo com a quantidade de classificadores que foi definida.

Com a população inicial de classificadores criada, inicia-se o processo de aprendizagem e desenvolvimento dos classificadores a fim de otimizar a sua capacidade de predição. Na primeira fase, o HLCS escolhe aleatoriamente uma instância de treinamento e a compara com os classificadores da população. A comparação é feita entre os atributos da instância de treinamento com as condições do classificador. Aqueles classificadores cujas condições ativas são iguais aos atributos da instância de treinamento, formam um conjunto de ação e são encaminhados para o componente de desempenho onde participarão de uma competição.

O componente de desempenho serve para analisar os classificadores e avaliar o processo de aprendizagem. Nesta competição o classificador que oferecer o maior lance efetivo ($eBid$), como mostra o cálculo da Equação 4.2, terá a chance de predizer a classe da instância de treinamento.

$$eBid = 1 + (Spec * VQualidade) * (1 + Mod) \tag{4.2}$$

$$Spec = \frac{total - ativas}{total} \tag{4.3}$$

sendo:

- $Spec$: valor da especificidade do classificador (definido na Equação 4.3), associada à proporção de condições ativas do classificador.
- $VQualidade$: valor da medida de qualidade do classificador. Este valor difere para cada modelo HLCS e será definido nas próximas seções;
- Mod: valor aleatório que representa a modulação caracterizada por um ruído com distribuição normal de média 0 e variância 1.
- $total$: total de condições do classificador;
- $ativas$: total de condições ativas do classificador.

Com o classificador vencedor definido, o componente de cessão de crédito analisa a predição da classe da instância de treinamento e em seguida define a recompensa do

classificador conforme o resultado obtido. Se o classificador vencedor prever corretamente a classe da instância, lhe é dada uma recompensa. Caso contrário o classificador recebe uma punição pelo erro de predição. Os cálculos de recompensa e punição diferem de cada modelo HLCS e serão explicados nas próximas seções.

Após as etapas de competição e de aprendizagem, os classificadores são encaminhados para o componente AG, que tem a função de criar novos classificadores que irão substituir os classificadores menos aptos. O componente AG é o responsável em criar e modificar os classificadores para que estes se tornem cada vez mais eficientes. Este componente utiliza como base os Algoritmos Genéticos que são técnicas probabilísticas de otimização global. Através dos operadores genéticos de cruzamento e mutação, os classificadores evoluem e sua qualidade tende a melhorar.

As configurações do HLCS permitem definir alguns parâmetros importantes na evolução da população, como por exemplo: a probabilidade de cruzamento e a probabilidade de mutação.

Para a execução deste componente, a cada ciclo, o HLCS seleciona aleatoriamente dois classificadores do conjunto de ação através do método do torneio. Um número aleatório é gerado, caso este número esteja dentro da probabilidade de cruzamento definida, executa-se o operador de cruzamento de um ponto nos dois classificadores selecionados e dois novos classificadores são gerados.

Os novos classificadores gerados são submetidos aos processos de identificação da classe e cálculo da medida de qualidade. Os quatro classificadores, os dois antigos e os dois novos gerados, são então comparados em relação ao valor da medida de qualidade ($VQualidade$) de cada um. Os dois classificadores que tiverem o maior valor de qualidade são encaminhados para a população e os outros dois são descartados.

Conforme a probabilidade de mutação, os classificadores também são submetidos ao operador de mutação. No operador de mutação para cada atributo do classificador um número aleatório é gerado, caso este número seja menor que a probabilidade de mutação definida, executa-se o operador de mutação que altera a opção de condição ativa ou inativa. O classificador resultante tem o valor da medida qualidade calculada e caso este valor seja maior que o atual, mantêm-se a mutação, caso contrário retorna os valores anteriores.

Tanto na execução do operador de cruzamento quanto na execução do operador de mutação a estratégia utilizada é sempre manter na população os classificadores mais aptos, ou seja, aqueles que possuírem o maior valor de qualidade.

Terminada esta etapa inicia-se um novo processo de aprendizagem com a cobrança de uma taxa de vida. O valor da taxa de vida, calculada de acordo com o ciclo de evolução, é cobrada de todos os classificadores da população e subtraída do valor da

medida de qualidade, como mostra a Equação 4.4. O classificador que tiver uma medida de qualidade menor ou igual a zero é excluído da população e um novo classificador é criado e adicionado na população.

$$VQualidade = VQualidade * (1 - \left(\frac{1}{2}\right)^{\frac{1}{n}}) \tag{4.4}$$

sendo:

- n: o número do ciclo de evolução do sistema.

Concluída esta etapa, um novo ciclo de evolução se inicia conforme as configurações de cada modelo. Nas próximas seções será dado um maior detalhamento de cada modelo do HLCS.

4.3 HLCS-Local

O algoritmo Hierarchical Learning Classifier System - Local (HLCS-Local) é um modelo local por nó baseado em LCS, proposto para trabalhar com problemas hierárquicos para a classificação da função de proteínas. Apresentado pelo autor em (ROMAO; NIEVOLA, 2011), o HLCS-Local ignora completamente a hierarquia de classes da base de dados, se comportando como um algoritmo de classificação tradicional durante as fases de treinamento e teste e fornecendo uma solução indireta para o problema da classificação hierárquica. Neste caso, classificadores binários independentes são treinados para cada nó da hierarquia exceto o nó raiz. Assim, para determinar o conjunto de instâncias positivas/negativas de cada classificador, para cada classe l_i (sendo i=número de classes da base de treinamento) é associado um classificador e uma análise é realizada em todos os exemplos da base de treinamento. O conjunto de exemplos positivos consiste em todas as instâncias que pertencem a classe l_i e o conjunto de negativos os exemplos restantes.

Para criação da população inicial de classificadores neste modelo, o usuário deve inicialmente definir a quantidade de classificadores que irão compor a população. No HLCS-Local cada classificador C_i tem a sua medida de qualidade calculada conforme mostra a Equação 4.5

$$VQualidade = \frac{\sum VP_i}{\sum VP_i + \sum FN_i} * \frac{\sum VN_i}{\sum VN_i + \sum FP_i} \tag{4.5}$$

sendo:

- VP_i: (Verdadeiro Positivo) é o número de instâncias que possuem todos os atributos iguais as condições ativas do classificador C_i e onde a classe predita é igual a classe real.

- VN_i: (Verdadeiro Negativo) é o número de instâncias que possuem todos os atributos iguais as condições ativas do classificador C_i e onde a classe predita é diferente da classe real.

- FP_i: (Falso Positivo) é o número de instâncias que não possuem todos os atributos iguais as condições ativas do classificador C_i e onde a classe predita é diferente da classe real.

- FN_i: (Falso Negativo) é o número de instâncias que não possuem todos os atributos iguais as condições ativas do classificador C_i e onde a classe predita é igual a classe real.

A partir da definição da população inicial os classificadores passam pelas etapas de competição e evolução. Neste modelo é preciso definir nas configurações iniciais a quantidade de ciclos que serão executados em cada uma destas etapas. Essa característica pode prejudicar a evolução do aprendizado dos classificadores por não existir um padrão definido de parada. Feito isso, é formado o conjunto de ação que irá participar da competição e encaminhado para o componente de desempenho seguindo o roteiro geral do modelo.

O classificador vencedor da competição, aquele que possui o maior $eBid$, tem a sua predição analisada pelo componente de cessão de crédito da seguinte forma: se o classificador predizer corretamente a classe da instância de treinamento, um percentual de recompensa $Perc\_Rec$ é acrescentado á sua medida de qualidade, conforme mostra a Equação 4.6. Por outro lado, caso a predição for incorreta, o classificador tem o valor da sua medida de qualidade reduzido pelo percentual de recompensa, conforme mostra a Equação 4.7. Os valores formam então a nova medida de qualidade do classificador.

$$VQualidade = VQualidade * (1 + Perc\_Rec) \tag{4.6}$$

$$VQualidade = VQualidade * (1 - Perc\_Rec) \tag{4.7}$$

sendo: $Perc\_Rec$ - percentual de recompensa definido nas configurações iniciais do HLCS-Local.

O processo segue então para o componente AG, responsável pela geração de novos classificadores. Dois classificadores são selecionados do conjunto de ação através do método de torneio e de acordo com as probabilidades definidas nas configurações iniciais são executas as operações de cruzamento e mutação conforme o modelo geral do HLCS.

4.4 HLCS-Tree

O algoritmo Hierarchical Learning Classifier System - Tree (HLCS-Tree) é um modelo global baseado em LCS, proposto para trabalhar com problemas hierárquicos para a classificação da função de proteínas estruturadas em árvores. Apresentado pelo autor em (ROMAO; NIEVOLA, 2012b), o HLCS-Tree cria um único modelo de classificação relativamente complexo que 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.

Para trabalhar com a hierarquia das classes, esta nova versão apresenta um componente específico para a tarefa que é o componente de avaliação dos classificadores. Este componente tem a missão de analisar a predição dos classificadores, considerando a hierarquia das classes.

O HLCS-Tree inicia a fase da criação da população inicial de classificadores seguindo a mesma sequência do modelo geral do HLCS. Entretanto, no HLCS-Tree para calcular a medida de qualidade dos classificadores da população inicial, dois fatores são considerados: o porcentagem de classes preditivas positivas (*revocacao*) e a avaliação do controle hierárquico do classificador (*avaliacao_h*). Esta avaliação representa a habilidade preditiva do classificador, considerando não somente a classe em questão, mas todos os antecedentes da classe na hierarquia.

A avaliação é uma forma de considerar as predições feitas pelos classificadores na hierarquia do problema. Principalmente, no caso da predição da função da proteína, é muito importante em termos biológicos, o conhecimento de todas as classes que são parte de uma função, desde a raiz até o mais específico. Com base neste princípio, o componente de avaliação é responsável em promover os classificadores que estão próximos do seu objetivo principal, levando em consideração a qualidade do classificador em prever ao menos alguma classe antecedente da classe real.

Através do componente de avaliação, o HLCS-Tree é capaz de verificar se a predição foi correta, incorreta ou parcialmente correta. Com este modelo é possível fazer com que a recompensa dada para o classificador seja mais dinâmica, de acordo com a sua predição. Assim, para cada predição, o HLCS-Tree calcula a avaliação ($avaliacao\_h(i)$

sendo: $0 < i \leq$ número de instâncias da base de treinamento) do classificador como mostra a Equação 4.8:

$$avaliacao\_h(i) = \begin{cases} 1, & \text{Se Correta} \\ 0, & \text{Se Incorreta} \\ \frac{NCAC}{NCR}, & \text{Se Parcialmente Correta} \end{cases} \tag{4.8}$$

sendo:

- $NCAC$: o número de classes antecedentes em comum
- NCR: nível da classe real na hierarquia

A predição correta ocorre quando a classe predita é igual a classe real, como mostra a Figura 4.1. Nos exemplos, o retângulo preto representa a classe predita e o círculo preto a classe real. No caso do exemplo da Figura, de acordo com a Equação 4.8 o valor da avaliação do classificador é 1.

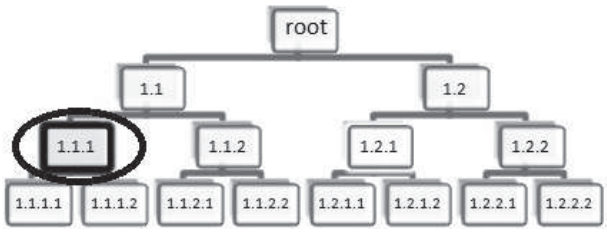

Figura 4.1: Exemplo de predição correta no HLCS-Tree

A predição incorreta ocorre quando a classe predita é diferente da classe real e de todos os seus antecedentes, como mostra a Figura 4.2. No caso do exemplo da Figura, de acordo com a Equação 4.8 o valor da avaliação do classificador é 0.

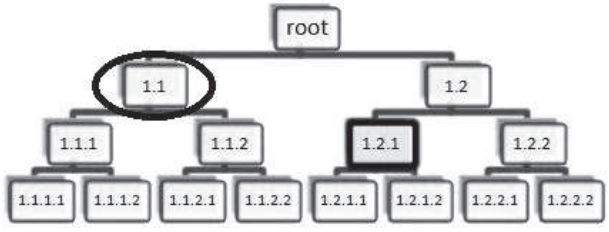

Figura 4.2: Exemplo de predição incorreta no HLCS-Tree

A predição parcialmente correta ocorre quando o algoritmo erra a classe real mas prediz corretamente ao menos um antecedente da classe real, desconsiderando o nó raiz. Neste caso, o valor da sua medida de qualidade varia de acordo com o erro, que é levado

em conta conforme o número de classes antecedentes em comum e o nível da classe real, como mostra a Equação 4.8. Na Figura 4.3, a classe predita (1.2.1) tem um antecedente em comum com a classe real (1.2.2) e, como a classe real está no segundo nível, o valor da avaliação do classificador é 0.5

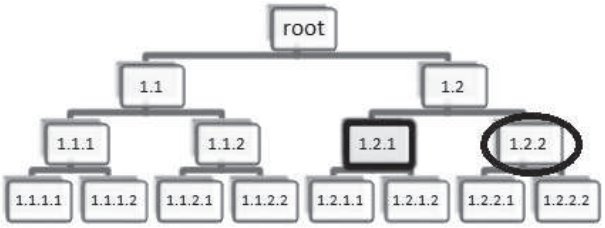

Figura 4.3: Exemplo de predição parcialmente correta no HLCS-Tree

A avaliação final do classificador ($avaliacao\_h$) é então calculada como o somatório das avaliações, como mostra a Equação 4.9. A medida de revocação ($revocacao$) utiliza a fórmula padrão como mostra a Equação 4.10, e estes valores formam a medida de qualidade ($VQualidade$) do classificador, como mostra a Equação 4.11

$$avaliacao\_h = \sum avaliacao\_h(i) \tag{4.9}$$

$$revocacao = \frac{VP}{(VP + FN)} \tag{4.10}$$

$$VQualidade = revocacao * avaliacao\_h \tag{4.11}$$

sendo:

- VP: (Verdadeiro Positivo) é o número de instâncias que possuem todos os atributos iguais as condições ativas do classificador C_i e onde a classe predita é igual a classe real.
- FN: (Falso Negativo) é o número de instâncias que não possuem todos os atributos iguais as condições ativas do classificador C_i e onde a classe predita é igual a classe real.

Este processo é então repetido para toda a população inicial de classificadores.

Seguindo com as etapas de competição e evolução, estas seguem o mesmo padrão do HLCS-Local. O classificador vencedor da competição, aquele que possuir o maior $eBid$, tem a sua predição analisada pelo componente de cessão de crédito. Entretanto, neste caso, o cálculo de recompensa e punição realizado pelo componente de cessão de crédito é modificado para também levar em consideração a avaliação do controle hierárquico do classificador. Assim, se o classificador predizer corretamente a classe da instância de treinamento, um percentual de recompensa $Perc\_Rec$ e o valor da sua avaliação é acrescentado á sua medida de qualidade, conforme mostra a Equação 4.12. Por outro lado, caso a predição for incorreta, o classificador tem o valor da sua medida de qualidade reduzido pelo percentual de recompensa, mas lhe é acrescentado o valor da sua avaliação, conforme mostra a Equação 4.7, caso a predição esteja parcialmente correto. Esta característica faz com que os classificadores que estiverem próximos de uma predição correta (parcialmente corretos), possam ser beneficiados também.

$$VQualidade = VQualidade * (1 + Perc\_Rec + avaliacao\_h) \tag{4.12}$$

$$VQualidade = VQualidade * (1 - Perc\_Rec + avaliacao\_h) \tag{4.13}$$

O processo segue então para o componente AG, responsável pela geração de novos classificadores conforme o modelo geral do HLCS.

4.5 HLCS-DAG

O algoritmo Hierarchical Learning Classifier System - DAG (HLCS-DAG) é um modelo global baseado em LCS proposto para trabalhar com problemas hierárquicos para a classificação da função de proteínas estruturadas na forma de DAGs. Apresentado pelo autor em (ROMAO; NIEVOLA, 2012a), o HLCS-DAG assim como o HLCS-Tree, também cria um único modelo de classificação relativamente complexo que 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. O diferencial entre os dois modelos é a capacidade do HLCS-DAG poder trabalhar com bases mais complexas estruturadas em grafos. Esta característica é bastante importante pelo fato de que as bases de dados que são utilizadas pela Ontologia Gênica, que servem atualmente como abordagem dominante

em ferramentas de anotação funcional, são estruturadas desta forma.

O HLCS-DAG inicia com a criação da população inicial de classificadores da mesma forma que o modelo HLCS-Tree. A diferença ocorre no cálculo da avaliação do controle hierárquico do classificador (*avaliacao_h*), que neste caso deve considerar a estrutura na forma de DAG.

Para isso, no HLCS-DAG, a fórmula que o componente de avaliação utiliza para verificar a predição do classificador foi modificada, principalmente com relação as predições parcialmente corretas. Esta modificação ocorreu pelo fato de que nas estruturas em DAG o nível que se encontram os antecedentes das classes, pode diferenciar conforme a estrutura da base. Com isso, utilizou-se uma fórmula baseada na menor distância entre a classe predita e a classe real para determinar o valor da avaliação nas predições parcialmente corretas. Assim, para cada predição, o HLCS-DAG calcula a avaliação ($avaliacao\_h(i)$ sendo: $0 < i \leq$ número de instâncias da base de treinamento) do classificador como mostra a Equação 4.14

$$avaliacao\_h(i) = \begin{cases} 1, & \text{Se Correta} \\ 0, & \text{Se Incorreta} \\ \frac{1}{1+(distancia_{p,r})}, & \text{Se Parcialmente Correta} \end{cases} \tag{4.14}$$

sendo:

- $distancia_{p,r}$: este é o número de arestas entre a classe real e a classe predita. Este valor é calculado utilizando o algoritmo de Dijkstra para encontrar o menor caminho dentro do grafo. Este distância é valida somente se existir pelo menos um nó que precede a classe real e a classe predita, descartando o nó raiz.

Desta forma, conforme a Figura 4.4, no item (a) a classe predita é igual a classe real. No exemplo, o círculo em linha dupla representa a classe predita e o círculo cinza a classe real.

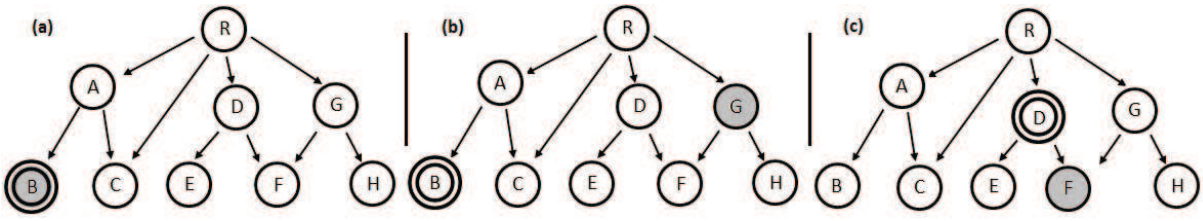

Figura 4.4: (a) Exemplo de predição correta. (b) Exemplo de predição incorreta. (c) Exemplo de predição parcialmente correta.

Neste caso, a classificação está correta e de acordo com a Equação 4.14 o valor da avaliação do classificador é 1. No item (b) a classe predita é diferente da classe real e de todos os seus antecedentes. Neste caso, a classificação está incorreta e de acordo com a Equação 4.14 o valor da avaliação do classificador é 0. No item (c) a classe predita é antecedente da classe real. A distância entre elas é de uma aresta e portanto, neste caso, a classificação é considerada parcialmente correta e de acordo com a Equação 4.14 o valor da avaliação do classificador é 0.5.

Em seguida, o HLCS-DAG inicia as etapas de competição e evolução. Estas etapas e o restante da execução do HLCS-DAG são realizadas da mesma forma do modelo HLCS-Tree definidas anteriormente.

4.6 HLCS-Multi

O algoritmo Hierarchical Learning Classifier System - Multilabel (HLCS-Multi) é o resultado final desta Tese e apresenta um modelo global hierárquico multirrótulo para a predição de problemas hierárquicos.

O HLCS-Multi inicia sua execução realizando um processo de decomposição das instâncias na base de treinamento. Este processo faz com que as instâncias multirrótulo se transformem em um conjunto de instâncias simples-rótulos conforme mostra a Tabela 4.1.

Tabela 4.1: Decomposição Multirrótulo

Instância	$Atributo_1$	**...**	$Atributo_n$	**Classe**
1	(-0.133-0.596]	...	(-0.366–0.039]	GO0003674 @ GO0005624 @ GO0045324
1.1	(-0.133-0.596]	...	(-0.366–0.039]	GO0003674
1.2	(-0.133-0.596]	...	(-0.366–0.039]	GO0005624
1.3	(-0.133-0.596]	...	(-0.366–0.039]	GO0045324

No exemplo, a instância 1 é composta por três classes, após a decomposição esta instância é substituída por três novas instâncias (1.1, 1.2, 1.3). Neste processo, os valores dos atributos das instâncias simples-rótulos geradas são mantidos iguais aos atributos das instâncias multirrótulo. Apesar de aumentar o quantidade de exemplos na base de treinamento, este simples processo faz com que se possa trabalhar com um novo conjunto de dados sem perder o conhecimento hierárquico entre eles.

Com o novo conjunto de treinamento definido inicia-se a etapa de criação da população inicial de classificadores. O HLCS-Multi define sua população inicial da mesma

forma com que é feita nos outros modelos e, dependendo do tipo de estrutura, utiliza os mesmos cálculos do HLCS-DAG e HLCS-Tree para a medida de qualidade dos classificadores. No HLCS-Multi o tamanho da população ($TamPop$) é determinado nas configurações iniciais por uma porcentagem ($Perc\_Pop$) em relação a quantidade de instâncias na base de treinamento ($Qtd\_Inst\_Inicial$), conforme mostra a Equação 4.15.

$$TamPop = Qtd\_Inst\_Inicial * Perc\_Pop/100 \tag{4.15}$$

Com a população inicial criada, inicia-se no HLCS-Multi o processo de aprendizagem e desenvolvimento dos classificadores. Este processo é executado conforme os outros modelos HLCS. Na primeira fase, o HLCS-Multi escolhe aleatoriamente uma instância de treinamento e a compara com os classificadores da população inicial. A comparação é feita entre os atributos da instância de treinamento com as condições do classificador. Aqueles classificadores cujas condições ativas são iguais aos atributos da instância de treinamento, formam um conjunto de ação e são encaminhados para o componente de desempenho onde participarão de uma competição.

O componente de desempenho serve para analisar os classificadores e avaliar o processo de aprendizagem. Nesta competição o classificador que oferecer o maior lance efetivo ($eBid$), como mostrado anteriormente na Equação 4.2, terá a chance de predizer a classe da instância de treinamento. O Algoritmo 1 mostra uma visão geral deste processo.

Algoritmo 1 Componente de Desempenho

```
1: Id_Instancia = Seleciona Instância() Aleatório;
2: for each i = 1 to TamPop do
3:     if Compara(instancia(Id_Instancia),classificador(i)) then
4:         Adiciona classificador(i) para Conjunto de Ação();
5:     end if
6: end for
7: for each i = 1 to Tamanho Conjunto de Ação do
8:     classificador(i).eBid ← 1 + (Spec * VQualidade) * (1 + Mod);
9:     if classificador(i).eBid > Maior_eBid then
10:        Maior_eBid ← classificador(i).eBid;
11:        Class_Vencedor ← i;
12:    end if
13: end for
14: Return Class_Vencedor
```

Com o classificador vencedor definido, o componente de cessão de créditos analisa a predição da classe da instância de treinamento e em seguida define a recompensa do classificador conforme o resultado obtido. O cálculo de recompensa e punição realizado pelo componente de cessão de crédito no HLCS-Multi, segue o mesmo processo realizado

pelos modelos HLCS-Tree e HLCS-DAG, conforme a estrutura da base de dados. O Algoritmo 2 mostra uma visão geral deste processo.

A diferença do HLCS-Multi com relação aos outros modelos, é que após a etapa de cessão de crédito uma nova população de classificadores é gerada. Esta população, chamada de população final, constituirá no modelo final de predição do HLCS-Multi e irá conter os vencedores de cada competição que predizer corretamente ou parcialmente correta a classe da instância escolhida no início da competição. Esta população final não possui uma quantidade de classificadores pré-definida, este valor dependerá do poder de aprendizado do modelo.

Algoritmo 2 Cessão de Crédito

```
1: if Analisa(classe(Id_Instancia),classe(Class_Vencedor)) then
2:    if (Classificação == CORRETA) then
3:       avaliacao_h ← 1;
4:       VQualidade ← VQualidade * (1 + Perc_Rec + avaliacao_h);
5:    else if (Classificação == PARCIALMENTE CORRETA) then
6:       avaliacao_h ← 1/1 + (distancia_{p,r});
7:       VQualidade ← VQualidade * (1 − Perc_Rec + avaliacao_h);
8:    else
9:       avaliacao_h ← 0;
10:      VQualidade ← VQualidade * (1 − Perc_Rec + avaliacao_h);
11:   end if
12: end if
```

Para definir a quantidade de execuções de aprendizado, cada classificador inserido na população final é comparado com todas as instâncias da base de treinamento. Todas as instâncias que forem cobertas pelo classificador e que a classe real da instância esteja correta ou parcialmente correta com a classe predita do classificador, são excluídas da base de treinamento. Após isso, o processo de aprendizagem recomeça até que uma porcentagem mínima de instâncias sejam cobertas pelos classificadores da população final ($Perc\_Cob$).

Este processo mantêm uma coerência com relação aos classificadores enviados para a população final e a característica hierárquica das instâncias da base de treinamento. Além disso, este critério define uma condição de parada no aprendizado do modelo.

Todo classificador inserido na população final é excluído da população inicial. Com isso, um novo classificador é gerado e incluído na população inicial do modelo para manter o tamanho do população. O processo segue então para o componente AG, responsável pela geração de novos classificadores conforme o modelo geral do HLCS. O Algoritmo 3 mostra uma visão geral da execução do modelo HLCS-Multi.

Algoritmo 3 Modelo Geral HLCS-Multi

```
1: Decomposição Multirrótulo ();
2: Qtd_Inst_Atual ← Qtd_Inst_Inicial;
3: TamPop ← Qtd_Inst_Inicial * Perc_Pop/100;
4: Gerar População Inicial(TamPop);
5: while Qtd_Inst_Atual > Qtd_Inst_Inicial * Perc_Cob do
6:    Componente Desempenho();
7:    Cessão de Crédito();
8:    Adiciona População Final(Class_Vencedor);
9:    for each i = 1 to Qtd_Inst_Atual do
10:      if Compara(instancia(i), Class_Vencedor) then
11:         if Analisa(classe(i),classe(Class_Vencedor)) then
12:            if (Classificação == CORRETA ou PARCIALMENTE CORRETA) then
13:               Remove instancia(i);
14:               Qtd_Inst_Atual − −;
15:            end if
16:         end if
17:      end if
18:   end for
19:   Crossover();
20:   Mutation();
21: end while
```

4.7 Simulação do Modelo HLCS-Multi

O processo geral do funcionamento do algoritmo HLCS-Multi envolve três fases principais: pré processamento dos dados, treinamento e teste.

A primeira fase, pré processamento dos dados, tem como objetivo fazer a preparação dos conjuntos de dados de treinamento e de teste. A segunda fase, treinamento, refere-se ao procedimento de criação do modelo proposto. E a terceira fase, teste, tem como finalidade testar o modelo proposto e avaliar os resultados obtidos.

Para facilitar o entendimento da execução do algoritmo HLCS-Multi fez-se uma simulação do mesmo passo-a-passo. Toda a simulação foi feita com dados factícios. Esta simulação segue as três fases de execução do algoritmo.

4.7.1 Pré Processamento dos Dados

A fase de pré processamento é aplicada para aumentar a qualidade e o poder de expressão dos dados a serem utilizados. Nesta fase é feita a limpeza, transformação e discretização dos dados para tornar possível a utilização do HLCS-Multi.

Na primeira etapa é importante fazer uma limpeza dos dados selecionados de forma a assegurar a qualidade dos fatos por eles representados. Nesta etapa são analisados os valores faltantes, e são feitas as identificações de “outliers” e ruídos e correções de informações errôneas ou inconsistentes.

Na transformação dos dados, a proposta é aumentar a qualidade e o poder de

expressão dos dados a serem minerados. No HLCS é feito o uso do método de discretização, que tem a função de transformar as variáveis contínuas das bases de dados em intervalos discretos representados por variáveis nominais. De acordo com (CIOS et al., 2007), o objetivo da discretização é reduzir o número de valores das variáveis contínuas, agrupando-as em um número n de intervalos. O uso de variáveis nominais torna o processo de aprendizagem mais simples e eficiente, diminuindo a necessidade de poder computacional de processamento. Além disso, o uso de variáveis nominais pode ainda resultar em regras mais adequadas ao domínio do problema.

Os métodos de discretização disponíveis são divididos em duas principais categorias: supervisionados e não supervisionados. Na discretização supervisionada existe um interdependência entre os valores das variáveis e a classe das instâncias do conjunto de treinamento, já a discretização não supervisionada não leva em consideração a informação de classes das instâncias do conjunto de treinamento.

Para esta simulação, a fase de Pré Processamento dos Dados foi simplificada com a criação de uma base de dados hierárquica multirrótulo estruturada no formato de um DAG. Foram criados dois conjuntos de dados, um de treinamento e outro para teste. Os valores dos conjuntos de dados de treinamento e teste, foram gerados aleatoriamente, assim como a atribuição das classes para cada instância.

Os valores das instâncias das bases foram discretizados com o método não supervisionado de particionamento de igual largura. Considerado um método simples de discretização (DOUGHERTY; KOHAVI; SAHAMI, 1995), este método divide o espaço dos valores observados em N intervalos de tamanho igual. Se A e B são os valores mínimo e máximo do atributo, a largura dos intervalos será: $W = (B - A)/N$. Este é um método paramétrico, pois o usuário tem de fornecer o número de intervalos em que deseja dividir o atributo.

Para a execução deste método foi definido o valor de dez intervalos e utilizada a ferramenta WEKA - (Waikato Environment for Knowledge Analysis) disponível no endereço http://www.cs.waikato.ac.nz/ml/weka/.

O padrão dos arquivos utilizados pelos modelos HLCS segue o formato ARFF. O formato ARFF foi desenvolvido pelo *Machine Learning Project* do Departamento de Ciência da Computação da Universidade de Waikato para ser utilizado com o software WEKA. Além disso, este padrão é utilizado também pelos principais algoritmos já descritos anteriormente. A Figura 4.5, mostra o arquivo de treinamento criado para a simulação.

Observa-se que o atributo *class*, em destaque, é do tipo *hierarchical* e representa a estrutura hierárquica dos dados. A Figura 4.6 mostra esta estrutura através do relacionamento entre as classes do conjunto de dados utilizado nesta simulação.

```
@relation base_de_treinamento-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last

@attribute 1
@attribute 2
@attribute 3
@attribute 4
@attribute 5
@attribute class        hierarchical root/A,A/D,A/E,root/B,root/C,B/E,B/F,C/G,D/H,D/I,C/F,F/I,F/J,G/J,G/L,root/E

@data
'\'(-inf-0.346]\'','\'(-inf-0.476]\'','\'(0.195-inf)\'','\'(-inf-0.594]\'','\'(-inf-0.389]\'',J@I@L
'\'(0.398-0.424]\'','\'(0.476-0.502]\'','\'(0.12-0.135]\'','\'(0.798-0.832]\'','\'(0.446-0.465]\'',A@E
'\'(0.346-0.372]\'','\'(0.658-0.684]\'','\'(0.18-0.195]\'','\'(0.866-inf)\'','\'(-inf-0.389]\'',C@D
'\'(0.554-inf)\'','\'(0.476-0.502]\'','\'(-inf-0.075]\'','\'(0.832-0.866]\'','\'(0.446-0.465]\'',D@G
'\'(0.476-0.502]\'','\'(0.684-inf)\'','\'(0.15-0.165]\'','\'(0.764-0.798]\'','\'(0.541-inf)\'',E
```

Figura 4.5: Exemplo do Arquivo de Treinamento

4.7.2 Treinamento

A fase de treinamento é onde ocorre a principal aplicação do algoritmo HLCS-Multi e está dividida em configuração dos parâmetros iniciais, leitura dos dados, decomposição e treinamento do modelo.

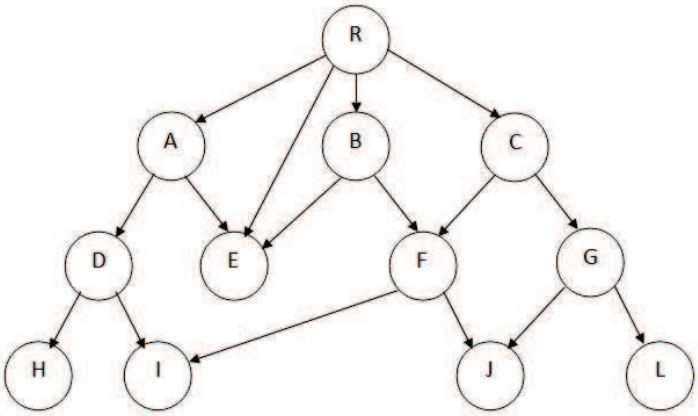

Figura 4.6: Visão do Relacionamento entre as Classes utilizadas nesta simulação

A configuração dos parâmetros iniciais consiste em atribuir valores para alguns parâmetros de inicialização do algoritmo HLCS-Multi, como segue:

- *Prop_Ativos*: (0% a 100%) - Define, na criação da população inicial, a proporção de condições ativas para cada classificador gerado.

- *Perc_Pop*: (0% a 100%) - Define o percentual de classificadores que serão gerados na população inicial, com base na quantidade de instâncias da base de treinamento.

- *Perc_Rec*: (0 a 1) - Representa o percentual de recompensa utilizado no cálculo da análise da predição feita pelo classificador.

- $Prob\_Crossover$: (0% a 100%) - Define a probabilidade de ocorrência da operação de cruzamento no sistema.

- $Prob\_Mutacao$: (0% a 100%) - Define a probabilidade de ocorrência da operação de mutação no sistema.

- $Perc\_Cob$: (0% a 100%) - Representa o critério de parada. É definido pelo percentual de instâncias na base de treinamento que devem ser cobertas pelos classificadores da população final.

Os parâmetros que necessitam ser inicializados e os valores utilizados nesta simulação são mostrados na Tabela 4.2.

Tabela 4.2: Parâmetros iniciais do HLCS-Multi nesta Simulação

Parâmetros	**Valores**
$Prop\_Ativos$	20%
$Perc\_Pop$	50%
$Perc\_Rec$	0.1
$Prob\_Crossover$	80%
$Prob\_Mutacao$	5%
$Perc\_Cob$	100%

A leitura dos dados é feita na base de treinamento de dimensão m. Esta base é formada por instâncias representadas por $B_{train} = [v_1, v_2, v_3, ..., v_y]$, onde y é o total de instâncias. Cada instância de B_{train} é formada por atributos, $v_i = [a_1, a_2, ..., a_j, a_{jk}]$, em que a_j é o *j-ésimo* atributo, $1 \leq j$ e a_{jk} representa o atributo classe. Como se trata de um problema multirrótulo o atributo a_{jk} pode ser formado por uma ou mais classes.

Ainda na leitura dos dados, é feito o aprendizado da hierarquia das classes H que será utilizada pelo algoritmo HLCS-Multi, representado na base de dados da seguinte forma: $H = (raiz/des_1, raiz/des_2, raiz/des_x, des_1/des_3, des_x des_y)$ onde, $raiz$ representa o nó raiz, des representa o nó descendente, A/B indica que A é pai de B, x e y a quantidade de relações A/B.

Com essas informações é gerada uma matriz que representa todos os caminhos possíveis entre a raiz e entre as classes descendentes da base e que serão utilizados pelo componente de avaliação durante a execução do sistema. A Figura 4.7 mostra a matriz gerada a partir da base de treinamento e que representa todos os caminhos possíveis entre a raiz e as classes descendentes da base.

	D	E	F	G	A	R	B	C	L	H	I	J
D	0	0	0	0	1	0	0	0	0	0	0	0
E	0	0	0	0	1	1	1	0	0	0	0	0
F	0	0	0	0	0	0	1	1	0	0	0	0
G	0	0	0	0	0	0	0	1	0	0	0	0
A	0	0	0	0	0	1	0	0	0	0	0	0
R	0	0	0	0	0	0	0	0	0	0	0	0
B	0	0	0	0	0	1	0	0	0	0	0	0
C	0	0	0	0	0	1	0	0	0	0	0	0
L	0	0	0	1	0	0	0	0	0	0	0	0
H	1	0	0	0	0	0	0	0	0	0	0	0
I	1	0	1	0	0	0	0	0	0	0	0	0
J	0	0	1	1	0	0	0	0	0	0	0	0

Figura 4.7: Representação da Hierarquia das Classes

Decomposição

O HLCS-Multi inicia sua execução realizando o processo de decomposição das instâncias na base de treinamento. Este processo faz com que as instâncias multirrótulo se transformem em um conjunto de instâncias simples-rótulos. Neste processo os valores dos atributos das instâncias simples-rótulo geradas são mantidos iguais aos atributos das instâncias multirrótulo. A Figura 4.8 mostra uma visão da base de treinamentos após o processo de decomposição.

```
@relation base_de_treinamento-decomposição-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last

@attribute 1
@attribute 2
@attribute 3
@attribute 4
@attribute 5
@attribute class        hierarchical root/A,A/D,A/E,root/B,root/C,B/E,B/F,C/G,D/H,D/I,C/F,F/I,F/J,G/J,G/L,root/E

@data
'\'(-inf-0.346]\'','\'(-inf-0.476]\'','\'(0.195-inf)\'','\'(-inf-0.594]\'','\'(-inf-0.389]\'',J
'\'(-inf-0.346]\'','\'(-inf-0.476]\'','\'(0.195-inf)\'','\'(-inf-0.594]\'','\'(-inf-0.389]\'',I
'\'(-inf-0.346]\'','\'(-inf-0.476]\'','\'(0.195-inf)\'','\'(-inf-0.594]\'','\'(-inf-0.389]\'',L
'\'(0.398-0.424]\'','\'(0.476-0.502]\'','\'(0.12-0.135]\'','\'(0.798-0.832]\'','\'(0.446-0.465]\'',A
'\'(0.398-0.424]\'','\'(0.476-0.502]\'','\'(0.12-0.135]\'','\'(0.798-0.832]\'','\'(0.446-0.465]\'',E
'\'(0.346-0.372]\'','\'(0.658-0.684]\'','\'(0.18-0.195]\'','\'(0.866-inf)\'','\'(-inf-0.389]\'',C
'\'(0.346-0.372]\'','\'(0.658-0.684]\'','\'(0.18-0.195]\'','\'(0.866-inf)\'','\'(-inf-0.389]\'',D
'\'(0.554-inf)\'','\'(0.476-0.502]\'','\'(-inf-0.075]\'','\'(0.832-0.866]\'','\'(0.465-0.484]\'',D
'\'(0.554-inf)\'','\'(0.476-0.502]\'','\'(-inf-0.075]\'','\'(0.832-0.866]\'','\'(0.465-0.484]\'',G
'\'(0.476-0.502]\'','\'(0.684-inf)\'','\'(0.15-0.165]\'','\'(0.764-0.798]\'','\'(0.541-inf)\'',E
```

Figura 4.8: Processo de Decomposição da Base de Treinamento

Treinamento do Modelo

A primeira etapa do treinamento do modelo é a criação da população inicial dos classificadores. A Tabela 4.3, mostra as instâncias do conjunto de treinamento e que representam o ambiente do sistema. Estas informações serão utilizadas para facilitar a compreensão de todos os passos.

Tabela 4.3: Conjunto de Treinamento

Instância	Atributo1	Atributo2	Atributo3	Atributo4	Atributo5	Classe
1	(-inf-0.346]	(-inf-0.476]	(0.195-inf)	(-inf-0.594]	(-inf-0.389]	J
2	(-inf-0.346]	(-inf-0.476]	(0.195-inf)	(-inf-0.594]	(-inf-0.389]	I
3	(-inf-0.346]	(-inf-0.476]	(0.195-inf)	(-inf-0.594]	(-inf-0.389]	L
4	(0.398-0.424]	(0.476-0.502]	(0.12-0.135]	(0.798-0.832]	(0.446-0.465]	A
5	(0.398-0.424]	(0.476-0.502]	(0.12-0.135]	(0.798-0.832]	(0.446-0.465]	E
6	(0.346-0.372]	(0.658-0.684]	(0.18-0.195]	(0.866-inf)	(-inf-0.389]	C
7	(0.346-0.372]	(0.658-0.684]	(0.18-0.195]	(0.866-inf)	(-inf-0.389]	D
8	(0.554-inf)	(0.476-0.502]	(-inf-0.075]	(0.832-0.866]	(0.446-0.465]	D
9	(0.554-inf)	(0.476-0.502]	(-inf-0.075]	(0.832-0.866]	(0.446-0.465]	G
10	(0.476-0.502]	(0.684-inf)	(0.15-0.165]	(0.764-0.798]	(0.541-inf)	E

Para criar o primeiro classificador (1), aleatoriamente, foi escolhida a instância 8 do conjunto de treinamento como modelo. Para cada atributo da instância de treinamento é criada uma condição no classificador. No início todas as condições começam com o operador de relação (OP) "=". O valor da condição (VL) recebe o valor do atributo da instância de treinamento e aleatoriamente é decidido se a condição ficará ativa (A) ou inativa (I). Nesta simulação a probabilidade do atributo ficar ativo foi de 20%, como definido anteriormente. A Figura 4.9 demonstra o resultado deste procedimento.

Instância	Atributo1	Atributo2	Atributo3	Atributo4	Atributo5	Classe
8	(0.554-inf)	(0.476-0.502]	(-inf-0.075]	(0.832-0.866]	(0.446-0.465]	D

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5		
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I
1	=	(0.554-inf)	I	=	(0.476-0.502]	A	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	A

Figura 4.9: Criação do Classificador

Após esta etapa, o classificador 1 é comparado com todas as instâncias da base de treinamento, a fim de atribuir sua classe. Cada instância coberta teve sua classe pontuada. Para o classificador 1, foram pontuadas quatro classes, como mostra a Tabela 4.4. Pelo fato das classes apresentarem a mesma pontuação, a primeira classe identificada, classe A, passou então a ser a classe correspondente do classificador 1.

Tabela 4.4: Classes Identificadas para o Classificador 1

Classe	**Pontuação**
A	1
E	1
D	1
G	1

O próximo passo foi calcular o valor da medida de qualidade do classificador 1. Novamente, o classificador 1 foi comparado com todas as instâncias da base de treinamento para gerar a sua medida de qualidade. As instâncias que foram cobertas pelo classificador 1 são mostradas na Tabela 4.5.

Tabela 4.5: Instâncias cobertas pelo Classificador 1

4	(0.398-0.424]	(0.476-0.502]	(0.12-0.135]	(0.798-0.832]	(0.446-0.465]	A
5	(0.398-0.424]	(0.476-0.502]	(0.12-0.135]	(0.798-0.832]	(0.446-0.465]	E
8	(0.554-inf)	(0.476-0.502]	(-inf-0.075]	(0.832-0.866]	(0.446-0.465]	D
9	(0.554-inf)	(0.476-0.502]	(-inf-0.075]	(0.832-0.866]	(0.446-0.465]	G

O cálculo da medida de qualidade foi feito de acordo com a Equação 4.11, demonstrada anteriormente. Para o cálculo da avaliação do classificador é levado em consideração a predição realizada com relação à hierarquia das classes como definido na Equação 4.14.

Como o HLCS-Multi utiliza o mecanismo de respostas corretas, incorretas e parcialmente corretas, nesta fase foi preciso analisar a predição dada para cada uma das instâncias cobertas pelo classificador 1, a fim de calcular sua avaliação.

Para a instância 4, como a classe predita foi A e a classe real também é A, a predição foi considerada correta e o valor da avaliação do classificador 1 foi 1, conforme mostra a Figura 4.10. Nas figuras relacionadas, o círculo preto representa a classe predita e o círculo preenchido de cinza a classe real.

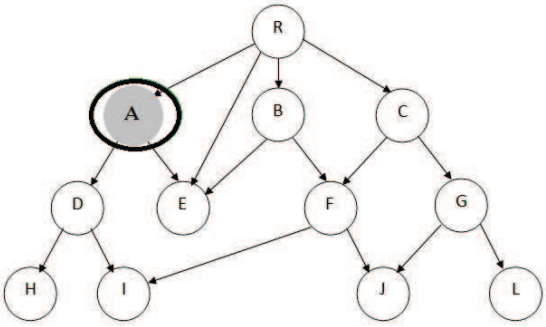

Figura 4.10: Instância Classificada Correta

Para a instância 5, como a classe predita foi A e a classe real é E, foi analisado se a classe predita era antecedente da classe real. Neste caso, como mostra a Figura 4.11, a classe predita é antecedente da classe real, sendo então a predição considerada parcialmente correta. O valor da avaliação do classificador, levou em consideração a

menor distância entre a classe real e a classe predita, conforme a Equação 4.14. Como a distância é de uma aresta, o valor da avaliação do classificador 1 para este caso é de 0.5.

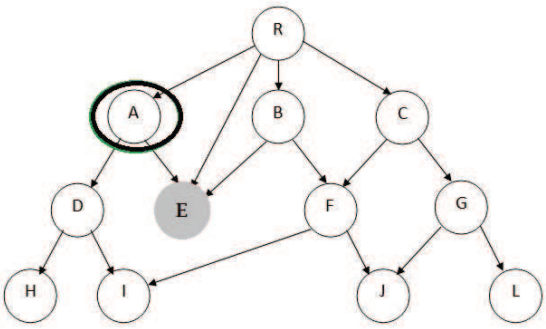

Figura 4.11: Instância Classificada Parcialmente Correta - Exemplo 1

Da mesma forma aconteceu para a instância 8, como a classe predita foi A e a classe real é D, foi analisado se a classe predita era antecedente da classe real. Neste caso, como mostra a Figura 4.12, a classe predita é antecedente da classe real, sendo então a predição considerada parcialmente correta. Como a distância também é de uma aresta, conforme a Equação 4.14, o valor da avaliação do classificador 1 para este caso é de 0.5.

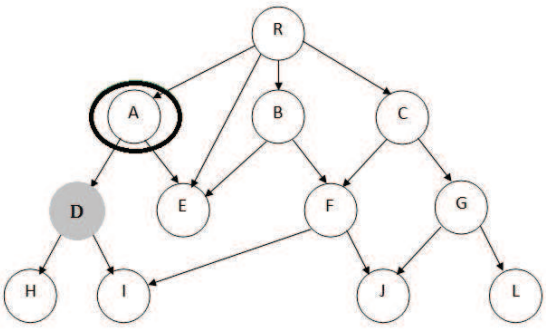

Figura 4.12: Instância Classificada Parcialmente Correta - Exemplo 2

Para a instância 9, como a classe predita foi A e a classe real era G, foi analisado se a classe predita era antecedente da classe real. Como o único antecedente em comum é o nó raiz, como mostra a Figura 4.13, a predição foi considerada incorreta e o valor da avaliação do classificador 1 para este caso é 0.

Assim, o valor da medida de qualidade do classificador 1 ficou definido como:

$$VQualidade = revocacao * avaliacao\_h$$

$$revocacao = \frac{VP}{(VP+FN)}$$

$$VQualidade = \frac{1}{(1+0)} * (1 + 0.5 + 0.5 + 0)$$

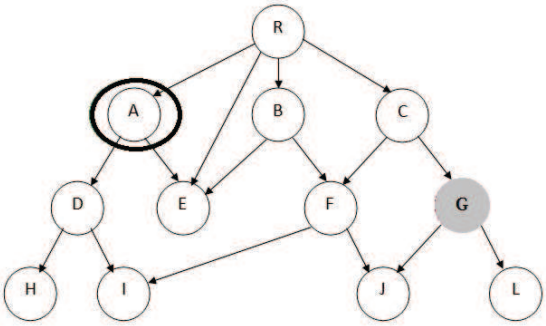

Figura 4.13: Instância Classificada Incorreta

$$VQualidade = 2.0$$

Este processo foi repetido até que toda a população inicial fosse criada. Como o tamanho da população foi definido como sendo 50% do tamanho da base de treinamento, a população inicial foi composta por cinco classificadores. A Tabela 4.6, mostra a população inicial dos classificadores após o procedimento.

Tabela 4.6: População Inicial de Classificadores

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5			Classe	VQualidade
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I		
1	=	(0.554-inf)	I	=	(0.476-0.502]	A	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	A	A	2.0
2	=	(0.554-inf)	I	=	(0.476-0.502]	I	=	(-inf-0.075]	A	=	(0.832-0.866]	I	=	(0.446-0.465]	A	D	0.5
3	=	(0.346-0.372]	I	=	(0.658-0.684]	I	=	(0.18-0.195]	A	=	(0.866-inf)	I	=	(-inf-0.389]	I	C	1.0
4	=	(0.554-inf)	A	=	(0.476-0.502]	I	=	(-inf-0.075]	I	=	(0.832-0.866]	A	=	(0.446-0.465]	I	D	0.5
5	=	(-inf-0.346]	I	=	(-inf-0.476]	I	=	(0.195-inf)	I	=	(-inf-0.594]	I	=	(-inf-0.389]	A	J	1.99

Com a população inicial de classificadores definida, iniciou-se a etapa de evolução dos classificadores. O primeiro passo é o ciclo de competição realizado pelo componente de desempenho. Para este procedimento, escolheu-se aleatoriamente uma instância da base de treinamento. Esta instância foi comparada com todos os classificadores da população. Os classificadores que cobrem a instância, formam um conjunto de ação para participar da competição. Para este exemplo, aleatoriamente, foi escolhida da base de treinamento a instância 8, ela e o conjunto de ação gerado são mostrados na Figura 4.14.

Para este conjunto de ação, foi então feito o cálculo do valor $eBid$ que corresponde ao lance efetivo do classificador na competição, conforme definido na Equação 4.2. O classificador 1 que obteve o maior $eBid$ foi então o escolhido para atuar sobre a instância.

Instância	Atributo1	Atributo2	Atributo3	Atributo4	Atributo5	Classe
8	(0.554-inf)	(0.476-0.502]	(-inf-0.075]	(0.832-0.866]	(0.446-0.465]	D

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5			Classe
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	
1	=	(0.554-inf)	I	=	(0.476-0.502]	A	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	A	A
2	=	(0.554-inf)	I	=	(0.476-0.502]	I	=	(-inf-0.075]	A	=	(0.832-0.866]	I	=	(0.446-0.465]	A	D
4	=	(0.554-inf)	A	=	(0.476-0.502]	I	=	(-inf-0.075]	I	=	(0.832-0.866]	A	=	(0.446-0.465]	I	D

Figura 4.14: Conjunto de Ação para a Competição

A Tabela 4.7 mostra o resultado deste cálculo e em destaque o vencedor.

Tabela 4.7: Resultado do Valor da Aposta *eBid*

Classificador	Spec	VQualidade	Mod	eBid	Classe
1	0.6	2.0	0.4472	2.7366	A
2	0.6	0.5	0.4877	1.4463	D
4	0.6	0.5	0.4682	1.4404	D

Definido o classificador vencedor, o componente de cessão de crédito, após ter analisado o resultado da predição, tem a função de compensar ou punir o classificador, conforme as equações definidas anteriormente em 4.12 e 4.13. Neste caso, o componente de avaliação também foi utilizado para a analisar a predição. Como resultado, uma nova medida de qualidade foi definida para o classificador, como mostra a Tabela 4.8.

Tabela 4.8: Resultado da Predição do Classificador Vencedor

Classificador	Predita	Real	Resultado	VQualidade	Perc_Rec	Avaliação	Novo-VQualidade
1	A	D	PARC. CORRETA	2.0	0.1	0.5	2.8

Observe que, neste exemplo, a classificação foi parcialmente correta. O classificador foi punido mas teve o valor da sua medida de qualidade aumentado pelo fato da predição estar próximo do resultado real. Com isso o classificador é enviado para a população final e excluído da população inicial de classificadores. Além disso, todas as instâncias cobertas pelo classificador são excluídas da base de treinamento.

A próxima etapa da evolução dos classificadores foi realizada pelo componente AG. Este componente utiliza como base os Algoritmos Genéticos e através dos operadores genéticos de cruzamento e mutação, os classificadores vão evoluindo e sua qualidade

melhorada. Para a etapa de cruzamento dois classificadores foram escolhidos do conjunto de ação através do método do torneio para participar da operação de cruzamento e dois novos classificadores são gerados. A Figura 4.15, mostra em destaque, a operação de cruzamento realizada entre os classificadores 1 e 4 escolhidos aleatoriamente. O ponto de corte foi realizado na condição 2 e os dois novos classificadores, A e B, foram criados.

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5		
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I
1	=	(0.554-inf)	I	=	(0.476-0.502]	A	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	A
4	=	(0.554-inf)	A	=	(0.476-0.502]	I	=	(-inf-0.075]	I	=	(0.832-0.866]	A	=	(0.446-0.465]	I

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5		
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I
A	=	(0.554-inf)	A	=	(0.476-0.502]	I	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	A
B	=	(0.554-inf)	I	=	(0.476-0.502]	A	=	(-inf-0.075]	I	=	(0.832-0.866]	A	=	(0.446-0.465]	I

Figura 4.15: Operação de Cruzamento

Os novos classificadores gerados são então submetidos aos processos de identificação da classe e cálculo da medida de qualidade. Com isso, os quatro classificadores, os dois antigos e os dois novos gerados, são comparados entre si através do valor da medida de qualidade. Os dois classificadores que tiverem o maior valor são encaminhados para a população e os outros dois são descartados.

Depois disso, é feita a cobrança da taxa de vida de todos os classificadores da população, conforme a Equação 4.4. Assim, termina-se a primeira iteração do ciclo de evolução. Este ciclo é repetido até que, como foi definido anteriormente, 100 % das instâncias da base de treinamento sejam cobertas. A população final, representada na Figura 4.16, é então o modelo de predição do sistema.

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5			Classe	VQualidade
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I		
1	=	(0.554-inf)	I	=	(0.476-0.502]	A	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	A	A	3.200
2	=	(0.554-inf)	A	=	(0.476-0.502]	A	=	(-inf-0.075]	A	=	(0.832-0.866]	I	=	(0.446-0.465]	I	A	2.786
3	=	(-inf-0.346]	I	=	(-inf-0.476]	A	=	(0.195-inf)	A	=	(-inf-0.594]	I	=	(-inf-0.389]	A	J	3.753
4	=	(0.554-inf)	A	=	(0.476-0.502]	I	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	I	G	2.058
5	=	(0.346-0.372]	I	=	(0.658-0.684]	I	=	(0.18-0.195]	I	=	(-inf-0.594]	I	=	(-inf-0.389]	A	D	2.058
6	=	(0.398-0.424]	A	=	(0.476-0.502]	A	=	(0.12-0.135]	I	=	(0.798-0.832]	A	=	(0.446-0.465]	I	E	1.029

Figura 4.16: População Final de Classificadores - Modelo de Predição

4.7.3 Teste

A última etapa refere-se a fase de teste do modelo de predição do sistema gerado e consiste em ler os dados de teste, realizar a predição e avaliar o resultado.

Leitura dos Dados

A etapa de leitura de dados foi realizada utilizando como entrada o conjunto de dados de teste, como mostrado na Figura 4.17.

```
@relation base_de_teste-weka.filters.unsupervised.attribute.Discretize-B10-M-1.0-Rfirst-last

@attribute 1
@attribute 2
@attribute 3
@attribute 4
@attribute 5
@attribute class        hierarchical root/A,A/D,A/E,root/B,root/C,B/E,B/F,C/G,D/H,D/I,C/F,F/I,F/J,G/J,G/L,root/E

@data
'\'(0.554-inf)\'','\'(-inf-0.476]\'','\'(-inf-0.075]\'','\'(-inf-0.594]\'','\'(-inf-0.389]\'',J@D
'\'(0.398-0.424]\'','\'(0.476-0.502]\'','\'(0.105-0.12]\'','\'(0.798-0.832]\'','\'(0.465-0.484]\'',B@A
'\'(0.554-inf)\'','\'(0.476-0.502]\'','\'(0.15-0.165]\'','\'(0.832-0.866]\'','\'(0.541-inf)\'',G@E
```

Figura 4.17: Exemplo do Arquivo de Teste

A Tabela 4.9, mostra as instância do conjunto de teste. Estas informações serão utilizadas para facilitar a compreensão dos passos do teste.

Tabela 4.9: Conjunto de Teste

Instância	Atributo1	Atributo2	Atributo3	Atributo4	Atributo5	Classe
1	(0.554-inf)	(-inf-0.476]	(-inf-0.075]	(-inf-0.594]	(-inf-0.389]	J@D
2	(0.398-0.424]	(0.476-0.502]	(0.105-0.12]	(0.798-0.832]	(0.465-0.484]	B@A
3	(0.554-inf)	(0.476-0.502]	(0.15-0.165]	(0.832-0.866]	(0.541-inf)	G@E

Realizar Predição

Para realizar a predição do modelo dos classificadores gerados na fase de treinamento, cada instância do conjunto de teste é comparada com todos os classificadores do modelo. A quantidade de classificadores selecionados irá variar de acordo com o número de classes da instância. Os classificadores que tiverem a maior medida de qualidade e que cobrirem a instância são escolhidos. Caso a quantidade de classificadores seja inferior a quantidade de classes da instância é utilizado um classificador padrão. Nesta simulação a classe do classificador padrão é E, por ser a classe que está presente mais vezes nas instâncias da base de treinamento.

No exemplo da Figura 4.18, foi feita a análise da instância 1 do conjunto de teste. A Figura mostra a instância e os classificadores identificados para realizar a predição.

Instância	Atributo1	Atributo2	Atributo3	Atributo4	Atributo5	Classe
1	(0.554-inf)	(-inf-0.476]	(-inf-0.075]	(-inf-0.594]	(-inf-0.389]	J@D

Classificadores Identificados

Classificador	Condição1			Condição2			Condição3			Condição4			Condição5			Classe
	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	OP	VL	A/I	
4	=	(0.554-inf)	A	=	(0.476-0.502]	I	=	(-inf-0.075]	I	=	(0.832-0.866]	I	=	(0.446-0.465]	I	G
Padrão	=	0	I	=	0	I	=	0	I	=	0	I	=	0	I	E

Figura 4.18: Exemplo da Análise do Modelo

Avaliar o Resultado

Para avaliar o resultado da predição, todas as possíveis combinações entre as classes preditas e as classes reais são realizadas. Aquela combinação que trouxer o melhor resultado de predição, levando em consideração as predições corretas e parcialmente corretas, é a que será utilizada para análise da predição. Desta forma, como resultado do teste do exemplo anterior, a classe J foi prevista como sendo G e a classe D como sendo a classe padrão E. Como ambas as classes preditas fazem parte dos antecedentes das classes reais, as predições foram consideradas como parcialmente corretas.

Para avaliar o resultado final da predição foi criada uma matriz de confusão, distribuindo todas as possíveis classes da hierarquia, como pode ser observada na Figura 4.19.

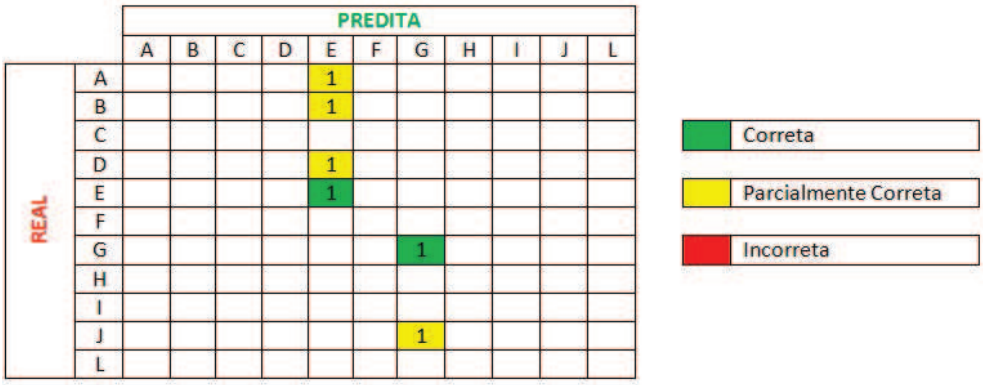

Figura 4.19: Resultado da Predição

Para concluir a avaliação do teste, foi utilizado os modelos de avaliação Revocação Hierárquica hR, Precisão Hierárquica hP e Medida-hF descritos anteriormente nas equações 2.7, 2.8 e 2.9 respectivamente. Além disso, foi utilizada para avaliação do modelo a taxa de acerto proporcional a hierarquia (aH).

Tomando como base o componente de avaliação do HLCS-Multi, caso a predição

esteja correta, considera-se 100% de acerto, caso a predição esteja incorreta, considera-se 0% de acerto. Caso a condição esteja parcialmente correta é calculada a distância entre a classe real e a classe predita, conforme a Equação 4.14, definida anteriormente. Com isso, a taxa de acerto pode variar, conforme a distância entre a classe real e a classe predita. A medida aH é então definida como o somatório do resultado da avaliação de todas as predições realizadas, como mostra a Equação 4.16

$$aH = \sum \frac{avaliacao\_h}{numero\_classes} \tag{4.16}$$

sendo:

- $avaliacao\_h$: o resultado obtido através da Equação 4.9
- $numero\_classes$: quantidade de classes na instância

Portanto, como mostra a Tabela 4.10, neste teste de simulação, considerando os resultados dos modelos de avaliação hierárquica, obteve-se os seguintes valores:

Tabela 4.10: Resultado da Avaliação Hierárquica da Simulação

hR	**hP**	**hF**	**aH**
0.666	0.816	0.734	63.88%

4.8 Considerações Finais do Capítulo

Neste capítulo foi apresentado o sistema HLCS com todas as suas versões que auxiliaram no aprendizado e amadurecimento da proposta. A primeira versão desenvolvida foi o HLCS-Local, esta abordagem apresenta uma solução local por nó para trabalhar com bases hierárquicas. A segunda versão foi o HLCS-Tree, que utiliza uma abordagem global para bases hierárquicas estruturadas em formato de árvores. A terceira versão foi o HLCS-DAG, que também utiliza uma abordagem global mas com a capacidade de trabalhar com bases hierárquicas estruturadas no formato DAG. E a quarta versão foi o HLCS-Multi que apresenta uma abordagem global multirrótulo para a classificação da função de proteínas, podendo trabalhar com bases hierárquicas estruturadas em formato de árvores ou DAG.

O HLCS-Local apresenta uma solução que ignora completamente a hierarquia de classes da base de dados, se comportando como um algoritmo de classificação tradicional

durante as fases de treinamento e teste, fornecendo uma solução indireta para o problema da classificação hierárquica. Neste caso, classificadores binários independentes são treinados para cada nó da hierarquia exceto o nó raiz.

O HLCS-Tree é um modelo global proposto para trabalhar com problemas hierárquicos para a classificação da função de proteínas estruturadas em árvores. O HLCS-Tree cria um único modelo de classificação relativamente complexo que 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.

Para trabalhar com a hierarquia das classes o HLCS-Tree apresenta um componente específico para a tarefa que é o componente de avaliação dos classificadores. Este componente tem a missão de analisar a predição dos classificadores, considerando a hierarquia das classes. Através do componente de avaliação, o HLCS-Tree é capaz de verificar se a predição foi correta, incorreta ou parcialmente correta.

Para calcular a medida de avaliação do controle hierárquico do classificador quando a predição parcialmente correta, o HLCS-Tree leva em consideração o número de classes antecedentes em comum e o nível da classe real.

O HLCS-DAG é um modelo global proposto para trabalhar com problemas hierárquicos para a classificação da função de proteínas estruturadas na forma de DAGs. O HLCS-DAG assim como o HLCS-Tree, também cria um único modelo de classificação relativamente complexo que 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. O diferencial entre os dois modelos é a capacidade do HLCS-DAG poder trabalhar com bases mais complexas estruturadas em grafos.

Para calcular a medida de avaliação do controle hierárquico do classificador quando a predição parcialmente correta, o HLCS-DAG leva em consideração o número de arestas entre a classe real e a classe predita. Este valor é calculado utilizando o algoritmo de Dijkstra para encontrar o menor caminho dentro do grafo. Este distância é valida somente se existir pelo menos um nó que precede a classe real e a classe predita, descartando o nó raiz.

O HLCS-Multi é o resultado final desta Tese e apresenta um modelo global hierárquico multirrótulo para a predição de problemas hierárquicos. O HLCS-Multi inicia sua execução realizando um processo de decomposição das instâncias na base de treinamento. Este processo faz com que as instâncias multirrótulo se transformem em um conjunto de instâncias simples-rótulos. Este simples processo faz com que se possa trabalhar com um novo conjunto de dados sem perder o conhecimento hierárquico entre eles. O cálculo da medida de avaliação do controle hierárquico do classificador utiliza os mesmos cálculos do HLCS-

DAG e HLCS-Tree para a medida de qualidade dos classificadores dependendo do tipo de estrutura.

A diferença do HLCS-Multi com relação aos outros modelos, é que após a etapa de cessão de crédito uma nova população de classificadores é gerada. Esta população, chamada de população final, constituirá no modelo final de predição do HLCS-Multi e irá conter os vencedores de cada competição que predizer corretamente ou parcialmente correta a classe da instância escolhida no início da competição.

A tabela 4.11 mostra um resumo das características dos algoritmos HLCS conforme critérios definidos em (SILLA; FREITAS, 2011) e apresentados no capítulo 2.

Tabela 4.11: Características dos algoritmos HLCS.

	Estrutura	**Profundidade**	**Cardinalidade**	**Abordagem**
HLCS-Local	DAG	Opcional em nós-folha	Simples	Local por Nó
HLCS-Tree	Árvore	Opcional em nós-folha	Simples	Global
HLCS-DAG	DAG	Opcional em nós-folha	Simples	Global
HLCS-Multi	Árvore e DAG	Opcional em nós-folha	Múltiplos	Global

Capítulo 5

Resultados Obtidos

Neste capítulo apresentam-se os resultados obtidos com os modelos HLCS propostos. A seção 5.1 mostra os resultados da avaliação do modelo HLCS-Local comparado com o conhecido método de classificação RIPPER. A seção 5.2 mostra os resultados obtidos entre o HLCS-Tree e o algoritmo GMNB. A seção 5.3 mostra o resultados da comparação entre o HLCS-DAG e o algoritmo hAnt-Miner. E finalmente na seção 5.4 os resultados obtidos com o algoritmo HLCS-Multi em comparação com os algoritmos HLCS-Tree, HLCS-DAG e Clus-HMC.

5.1 HLCS-Local X RIPPER

Em (ROMAO; NIEVOLA, 2011) o método HLCS-Local foi comparado com o conhecido método de classificação baseado em regras RIPPER (*Repeated Incremental Pruning to Produce Error Reduction*) proposto em (COHEN, 1995). O RIPPER foi executado no software WEKA (*Waikato Environment for Knowledge Analysis*). O algoritmo RIPPER induz a classificação de regras ordenadas do tipo SE-ENTÃO para um conjunto de exemplos pré-rotulados produzindo regras competitivas. Utiliza o critério de poda para reduzir taxas de erro e o uso de uma heurística baseada em MDL (*Minimum Description Length*), para determinar quantas regras devem formar o modelo.

Os experimentos foram realizados em três conjuntos de dados de funções de proteínas, Cellcycle, Church e Derisi que foram utilizados no trabalho de (VENS et al., 2008) e estão disponíveis em (http://dtai.cs.kuleuven.be/clus/hmcdatasets). Estas bases contêm informações de classes baseadas nos termos da Ontologia Gênica (GO) e são organizadas em uma estrutura em forma de um DAG. Destas bases, para este experimento, foram considerados os três primeiros níveis dos exemplos relacionados ao domínio da função molecular da GO e que contivessem, ao menos, quatro exemplos por classe. Os valores

dos exemplos das bases foram discretizados utilizando o método não supervisionado de particionamento de igual largura (DOUGHERTY; KOHAVI; SAHAMI, 1995). Para isso foi utilizado a implementação *Discretize* disponível no WEKA definido com dez intervalos e reorganizados para conter somente uma classe por exemplo. A Tabela 5.1 mostra detalhes dos dados utilizados no experimento.

Tabela 5.1: Detalhes dos Conjuntos de dados de Funções de Proteínas.

Base de Dados	**Atributos**	**Classes**	**Exemplos**
Cellcycle	78	29	1349
Church	28	29	1352
Derisi	64	29	1320

O desempenho do HLCS-Local e do RIPPER foi comparado utilizando as medidas de precisão hierárquica (hP), revocação hierárquica (hR) e Medida-hF propostas em (KIRITCHENKO; MATWIN; FAMILI, 2005) já definidas anteriormente. Os experimentos foram realizados utilizando como procedimento o método de validação cruzada fator 10 e os resultados são descritos como valores-médios e desvio padrão computados sobre 10 execuções. Os resultados da comparação entre o método proposto HLCS-Local contra o RIPPER são mostrados na Tabela 5.2, onde um valor é mostrado em negrito se a medida hierárquica é significativamente maior de acordo com a avaliação de duas caudas da distribuição t de Student com 95% de confidência.

Tabela 5.2: Valores das medidas hierárquica de Revocação (hR), Precisão (hP) e Medida-hF (hF) (média ± desvio padrão) nos três conjuntos de dados.

	RIPPER		
	hR	hP	hF
Cellcyle	0.6925±0.147	**0.5853±0.040**	0.6262±0.063
Church	0.9458±0.040	0.5358±0.029	0.6851±0.017
Derisi	0.9438±0.033	**0.5657±0.026**	**0.7068±0.021**

	HLCS-Local		
	hR	hP	hF
Cellcyle	**0.9842±0.017**	0.4964±0.023	0.6596±0.021
Church	0.9028±0.045	0.5310±0.035	0.6679±0.033
Derisi	**0.9917±0.012**	0.4939±0.024	0.6591±0.023

Trabalhando com algoritmos que naturalmente não consideram a hierarquia dos

dados, pode-se dizer que o fato de existir muito mais exemplos negativos do que positivos para cada nó da GO trouxe uma dificuldade maior para a predição. Entretanto, durante os testes tornou-se evidente que o algoritmo HLCS-Local foi capaz de tratar melhor este problema. Os resultados mostram que o HLCS-Local foi significativamente superior em algumas medidas.

5.2 HLCS-Tree X GMNB

Em (ROMAO; NIEVOLA, 2012b) foi realizado a comparação do método HLCS-Tree com a abordagem proposta em (SILLA; FREITAS, 2009) chamada de GMNB (*Global-Model Naive Bayes*), que aborda um classificador hierárquico global baseado no algoritmo Naive Bayes.

Os experimentos foram realizados com bases de dados de duas diferentes famílias de proteínas: Enzimas e GPCR. Estas bases foram utilizadas no trabalho de (SILLA; FREITAS, 2009) e estão disponíveis em https:// sites.google.com/site/carlossillajr/resources. Cada base de dados possui quatro versões baseadas em diferentes tipos de atributos preditores. Cada tipo de atributo preditor binário indica se ocorre uma "assinatura da proteína" ou não. As assinaturas utilizadas neste experimento foram: *Interpro*, *Pfam*, *Prints* e *Prosite*. A Tabela 5.3 mostra as características finais de cada base de dados depois da etapa de pré-processamento realizada e detalhada pelos autores em (SILLA; FREITAS, 2009). Em todas as bases para cada instância é atribuído ao menos uma classe de cada nível da hierarquia.

Tabela 5.3: Resumo dos dados. A coluna # Atributos define a quantidade de atributos da base, a coluna # Exemplos define o número de exemplos da base e a coluna # Classe/Nível representa o número de classes em cada nível da hierarquia (1o/2o/3o/4o nível)

Tipo Proteína	Assinatura	# Atributos	# Exemplos	# Classe/Nível
Enzima	Interpro	1216	14027	6/41/96/187
	Pfarm	708	13897	6/41/96/190
	Prints	382	14025	6/45/92/208
	Prosite	585	14041	6/42/89/187
GPCR	Interpro	450	7444	12/54/82/50
	Pfarm	75	7053	12/52/79/49
	Prints	283	5004	8/46/76/49
	Prosite	129	6246	9/50/79/49

O desempenho do HLCS-Tree e do GMNB foi comparado utilizando as medidas

de precisão hierárquica (hP), revocação hierárquica (hR) e Medida-hF já definidas anteriormente. Os experimentos foram realizados utilizando como procedimento o método de validação cruzada fator 10 e os resultados são descritos como valores-médios. Os resultados referentes as duas abordagens são descritos na Tabela 5.4.

Tabela 5.4: Valores das medidas hierárquicas de Revocação (hR), Precisão (hP) e Medida-hF (hF) sobre o conjunto de dados hierárquicos da Função da Proteína.

Tipo Proteína	Assinatura	HLCS-Tree			GMNB		
		hP	hR	HF	hP	hR	HF
Enzima	Interpro	87.80	85.36	86.56	94.96	89.58	90.53
	Pfarm	86.34	81.47	83.83	95.15	86.94	88.72
	Prints	89.69	82.33	85.85	92.21	87.26	87.98
	Prosite	90.35	86.27	88.26	95.14	89.53	90.70
GPCR	Interpro	90.26	74.30	81.51	87.60	71.33	77.01
	Pfarm	82.53	60.30	69.69	77.23	57.52	64.40
	Prints	86.50	68.18	76.25	87.06	69.42	75.38
	Prosite	79.42	60.45	68.65	75.64	53.73	61.14

Os resultados mostram que o HLCS-Tree teve uma performance melhor nas bases de dados do grupo das proteínas tipo GPCR, provavelmente devido ao fato que estas bases tem uma melhor distribuição das classes nos diferentes níveis da hierarquia. Nos outros resultados existe certo equilíbrio entre as medições. A principal vantagem do modelo HLCS-Tree com relação a abordagem GMNB é a forma com o qual o modelo apresenta os resultados gerados. Enquanto o GMNB apresenta um modelo baseado em probabilidades, o HLCS-Tree gera um conjunto de regras, tornando o conhecimento mais fácil de ser compreendido pelas comunidades médica e científica.

5.3 HLCS-DAG X hAnt-Miner

Em (ROMAO; NIEVOLA, 2012a) o HLCS-DAG foi testado em conjuntos de dados que envolvem a função de proteínas *Ion Channel* e comparado com o algoritmo proposto por (OTERO; FREITAS; JOHNSON, 2009) chamado de hAnt-Miner, que aborda uma solução global para a classificação da função de proteínas utilizando Colônia de Formigas.

O desempenho do HLCS-DAG e do hAnt-Miner foi comparado utilizando as medidas de precisão hierárquica (hP), revocação hierárquica (hR) e Medida-hF já definidas anteriormente. Os experimentos foram realizados utilizando como procedimento o método de validação cruzada fator 10 e os resultados são descritos como valores-médios e desvio padrão.

Os resultados da comparação entre o modelo HLCS-DAG e o método hAnt-Miner são mostrados na Tabela 5.5. De modo a medir se existe alguma diferença estatisticamente significante entre os métodos de classificação, foi utilizado o teste de Wilcoxon (*Wilcoxon Signed Rank Test*).

Tabela 5.5: Valores das medidas hierárquicas de Revocação (hR), Precisão (hP) e Medida-hF (hF) (média ± desvio padrão) nos três conjuntos de dados. Na coluna 'hF', os melhores resultados são mostrados em negrito.

	hAnt-Miner		
	hP	hR	hF
DS1 AA	0.56 ± 0.06	0.55 ± 0.06	0.56 ± 0.06
DS1 InterPro	0.82 ± 0.04	0.81 ± 0.04	**0.81 ± 0.04**
DS2 AA	0.63 ± 0.02	0.59 ± 0.02	0.61 ± 0.01
DS2 InterPro	0.83 ± 0.01	0.75 ± 0.01	**0.79 ± 0.01**

	HLCS-DAG		
	hP	hR	hF
DS1 AA	0.84 ± 0.02	0.64 ± 0.03	**0.73 ± 0.02**
DS1 InterPro	0.54 ± 0.03	0.65 ± 0.02	0.59 ± 0.02
DS2 AA	0.86 ± 0.04	0.58 ± 0.03	**0.69 ± 0.01**
DS2 InterPro	0.64 ± 0.04	0.61 ± 0.03	0.63 ± 0.02

O teste de Wilcoxon mostrou que não existem diferenças significativas entre os dois classificadores com uma confidência de 95%. Entretanto os valores mostram que o HLCS-DAG tem melhores resultados em algumas medidas quando comparado com o método hAnt-Miner. O HLCS-DAG superou o hAnt-Miner em duas das quatro bases de dados, chamadas de 'DS1 AA' e 'DS2 AA'. Estas bases consistem de atributos reais, enquanto as outras bases de dados são compostas por atributos booleanos. Como a maioria dos problemas reais consistem de dados com atributos reais, mostra que o HLCS-DAG é mais robusto que o hAnt-Miner ao lidar com este tipo de problema.

5.4 Resultados HLCS-Multi

O modelo HLCS-Multi foi comparado contra os algoritmos HLCS-Tree e HLCS-DAG definidos anteriormente e contra o algoritmo Clus-HMC proposto em (VENS et al., 2008). Para estes testes foram selecionadas vinte bases de dados ligadas a bioinformática e que utilizam duas diferentes estruturas hierárquicas de classes: árvore (bases FunCat) e DAG (bases Ontologia Gênica).

As bases que estão disponíveis no endereço (http://dtai.cs.kuleuven.be/clus/ hmc-datasets) são definidas em três conjuntos de dados: treinamento, validação e teste. Para as avaliações, os algoritmos foram executados utilizando somente os conjuntos de treinamento e teste, conservando os mesmos exemplos disponíveis. A Tabela 5.6 apresenta os detalhes das bases utilizadas neste experimento.

Tabela 5.6: Resumo dos conjuntos de dados utilizados no experimento. A primeira coluna ('Bases') define o nome da base de dados, a segunda coluna ('Treinamento') define o número de exemplos de treinamento, a terceira coluna ('Teste') define o número de exemplos de teste, a quarta coluna ('Atributos') define o número de atributos de cada base e a quinta coluna ('Classes') define o número de classes na classe hierárquica.

FunCat				
Bases	Treinamento	Teste	Atributos	Classes
Cellcycle	2476	1281	77	500
Church	1630	1281	27	500
Derisi	2450	1275	63	500
Eisen	1587	837	79	462
Expr	2488	1291	551	500
Gasch1	2480	1284	173	500
Gasch2	1639	1291	52	500
Pheno	1009	582	69	456
Seq	2580	1339	478	500
Spo	2437	1266	80	500
Ontologia Gênica - GO				
Bases	Treinamento	Teste	Atributos	Classes
Cellcycle	2473	1278	77	4126
Church	1627	1278	27	4126
Derisi	2447	1272	63	4120
Eisen	1583	835	79	3574
Expr	2485	1288	551	4132
Gasch1	2477	1281	173	4126
Gasch2	1635	1288	52	4126
Pheno	1005	581	69	3128
Seq	2568	1332	478	4134
Spo	2434	1263	80	4120

Os teste foram executados em um servidor Intel Xeon Quad Core E5410, com 8GB de memória RAM rodando Ubuntu Server 64 bits versão 10.10. As variáveis de inicialização utilizadas pelo HLCS-Multi são mostradas na Tabela 5.7.

Nos testes entre o HLCS-Multi e os modelos HLCS-Tree e HLCS-DAG, foram utilizadas as medidas de precisão hierárquica (hP), revocação hierárquica (hR) e Medida-

Tabela 5.7: Parâmetros iniciais do HLCS-Multi utilizados nos testes

Parâmetros	**Valores**
Prop_Ativos	20%
Perc_Pop	10%
Perc_Rec	0.1
Prob_Crossover	80%
Prob_Mutacao	5%
Perc_Cob	95%

hF (hF). Como forma de avaliar o acerto dos métodos foi utilizado também a taxa de acerto proporcional a hierarquia (aH) definido na Equação 4.16. Os resultados são descritos como valores-médios e desvio padrão computados sobre 10 execuções. De modo a medir se existe alguma diferença estatisticamente significante entre os algoritmos, foi utilizado o teste de Wilcoxon (*Wilcoxon Signed Rank Test*) com um nível de confiança de 95%, fortemente recomendado em (DEMŝAR, 2006) para este tipo de caso.

No primeiro teste, o HLCS-Multi foi comparado com o algoritmo HLCS-Tree utilizando as bases de dados da ontologia FunCat. De acordo com os resultados, o HLCS-Multi apresentou resultados significativamente melhores do que o HLCS-Tree em quase todas as bases analisadas. Esta resposta se deve principalmente pela complexidade das bases com relação a característica multirrótulo. Os resultados da comparação entre o modelo HLCS-Multi e o HLCS-Tree são mostrados na Tabela 5.8.

No segundo teste, o HLCS-Multi foi comparado com o algoritmo HLCS-DAG utilizando as bases de dados da Ontologia Gênica (GO). De acordo com os resultados, o HLCS-Multi apresentou resultados significativamente melhores do que o HLCS-DAG em todas as bases analisadas. Conforme já definido anteriormente, as bases GO são de grande complexidade, alguns exemplos possuem até 22 classes e o total de classes ultrapassa em mais de 4100 na maioria das bases. No entanto, o HLCS-Multi através dos componentes hierárquico e multirrótulo, mostrou ser mais adequado para este tipo de problema. Os resultados da comparação entre o modelo HLCS-Multi e o HLCS-DAG são mostrados na Tabela 5.9.

O último teste foi realizado contra o algoritmo Clus-HMC. O Clus-HMC consiste em um modelo de classificação hierárquica baseado no método de árvore de decisão. No teste realizado, a avaliação dos dados foi prejudicada pelo fato dos autores do mesmo, publicarem e disponibilizarem através do algoritmo somente resultados baseados em medidas binárias. Como visto em todo trabalho, o HLCS-Multi é um algoritmo global hierárquico multirrótulo, que promove seus classificadores que possuem a capacidade de prever ao

Tabela 5.8: Valores das medidas hierárquicas de Revocação (hR), Precisão (hP) e Medida-hF (hF) (média ± desvio padrão) nos três conjuntos de dados e medida da taxa de acerto proporcional a hierarquia (aH). A Medida-hF (hF) em negrito, mostra que o valor é significativamente superior de acordo com o teste de Wilcoxon.

		HLCS-Tree		
	hP	**hR**	**hF**	**aH**
Cellcyle	0.2184 ± 0.03	0.1982 ± 0.02	0.2078 ± 0.03	11.72%
Church	0.2091 ± 0.06	0.1813 ± 0.04	0.1942 ± 0.03	10.50%
Derisi	0.2036 ± 0.05	0.0900 ± 0.03	0.1248 ± 0.02	9.32%
Eisen	0.2047 ± 0.03	0.1676 ± 0.04	0.1843 ± 0.03	10.23%
Expr	0.2091 ± 0.04	0.1821 ± 0.04	0.1947 ± 0.04	10.40%
Gasch1	0.2089 ± 0.04	0.1820 ± 0.03	0.1945 ± 0.03	10.29%
Gasch2	0.1270 ± 0.02	0.0708 ± 0.04	0.0909 ± 0.04	6.14%
Pheno	0.2907 ± 0.04	0.5638 ± 0.05	0.3836 ± 0.05	22.56%
Seq	0.2126 ± 0.01	0.1863 ± 0.03	0.1986 ± 0.03	10.80%
Spo	0.1814 ± 0.03	0.1351 ± 0.04	0.1549 ± 0.02	8.97%
		HLCS-Multi		
	hP	**hR**	**hF**	**aH**
Cellcyle	0.2496 ± 0.05	0.2777 ± 0.04	**0.2629 ± 0.04**	15.48%
Church	0.2059 ± 0.03	0.2601 ± 0.03	**0.2298 ± 0.03**	14.00%
Derisi	0.3440 ± 0.02	0.3200 ± 0.03	**0.3316 ± 0.03**	18.63%
Eisen	0.2096 ± 0.05	0.2463 ± 0.04	**0.2265 ± 0.05**	11.68%
Expr	0.1886 ± 0.03	0.2243 ± 0.04	0.2049 ± 0.03	10.15%
Gasch1	0.2048 ± 0.01	0.2384 ± 0.02	**0.2203 ± 0.02**	12.56%
Gasch2	0.1611 ± 0.05	0.2427 ± 0.05	**0.1936 ± 0.04**	14.65%
Pheno	0.5686 ± 0.04	0.3082 ± 0.02	0.3997 ± 0.03	22.00%
Seq	0.1992 ± 0.03	0.2320 ± 0.02	0.2147 ± 0.03	10.60%
Spo	0.1869 ± 0.03	0.2466 ± 0.03	**0.2126 ± 0.02**	13.76%

menos alguma classe antecedente da classe real.

Este fato faz com que o HLCS-Multi tenha um percentual de acerto baixo quando comparado com os tradicionais métodos de avaliação binária, precisão e revocação. Entretanto, como visto nos testes anteriores, levando em consideração a hierarquia, o HLCS-Multi apresenta alguns resultados satisfatórios.

No entanto, para que se pudesse avaliar o HLCS-Multi contra o Clus-HMC, testes foram realizados utilizando as bases de dados FunCat e as bases de dados GO. Com os resultados de precisão e revocação obtidos pelo Clus-HMC, utilizando as configurações de limiares padrão e executadas conforme material disponibilizado pelos autores no endereço (http://dtai.cs.kuleuven.be/clus/hmcdatasets), foram desenhados os gráficos da Curva PR para todas as bases de dados.

Como os valores de precisão e revocação do HLCS-Multi não mudam muito com

Tabela 5.9: Valores das medidas hierárquicas de Revocação (hR), Precisão (hP) e Medida-hF (hF) (média ± desvio padrão) nos três conjuntos de dados e medida da taxa de acerto proporcional a hierarquia (aH). A Medida-hF (hF) em negrito, mostra que o valor é significativamente superior de acordo com o teste de Wilcoxon.

		HLCS-DAG		
	hP	**hR**	**hF**	**aH**
Cellcyle	0.2207 ± 0.03	0.1463 ± 0.04	0.1759 ± 0.03	9.62%
Church	0.2848 ± 0.05	0.1469 ± 0.06	0.1939 ± 0.04	9.47%
Derisi	0.2208 ± 0.02	0.1029 ± 0.03	0.1404 ± 0.02	8.06%
Eisen	0.1748 ± 0.03	0.1530 ± 0.03	0.1647 ± 0.02	8.70%
Expr	0.1564 ± 0.04	0.0889 ± 0.03	0.1134 ± 0.03	7.89%
Gasch1	0.2384 ± 0.02	0.1416 ± 0.04	0.1776 ± 0.03	10.10%
Gasch2	0.2716 ± 0.03	0.1049 ± 0.03	0.1514 ± 0.04	8.14%
Pheno	0.2303 ± 0.05	0.0730 ± 0.04	0.1109 ± 0.05	8.76%
Seq	0.1987 ± 0.04	0.1358 ± 0.03	0.1613 ± 0.02	9.58%
Spo	0.1789 ± 0.03	0.1645 ± 0.03	0.1714 ± 0.04	9.40%
		HLCS-Multi		
	hP	**hR**	**hF**	**aH**
Cellcyle	0.2611 ± 0.04	0.4209 ± 0.02	**0.3223 ± 0.03**	16.63%
Church	0.2259 ± 0.06	0.6183 ± 0.03	**0.3309 ± 0.04**	17.52%
Derisi	0.2948 ± 0.03	0.5655 ± 0.03	**0.3873 ± 0.03**	14.55%
Eisen	0.2206 ± 0.03	0.2407 ± 0.04	**0.2302 ± 0.04**	11.27%
Expr	0.3254 ± 0.05	0.3568 ± 0.03	**0.3109 ± 0.04**	15.57%
Gasch1	0.2577 ± 0.03	0.4203 ± 0.05	**0.3195 ± 0.03**	16.37%
Gasch2	0.2831 ± 0.02	0.4703 ± 0.03	**0.3534 ± 0.04**	18.15%
Pheno	0.2962 ± 0.04	0.7444 ± 0.04	**0.4281 ± 0.05**	20.01%
Seq	0.2347 ± 0.05	0.3547 ± 0.06	**0.2825 ± 0.04**	18.65%
Spo	0.2975 ± 0.03	0.3992 ± 0.04	**0.3410 ± 0.04**	17.74%

configurações diferentes de parâmetros, os resultados obtidos pelo HLCS-Multi foram marcados como um ponto no gráfico. Assim, como a comparação está sendo feita entre curva e ponto, é possível definir uma melhor performance do HLCS-Multi quando a curva estiver abaixo do ponto e uma melhor performance do Clus-HMC quando a curva estiver acima do ponto. Além disso, outro fator que pode ser levado em consideração é o fato de que, nestas bases, o número de exemplos negativos para cada classe supera em muito o número de exemplos positivos. O que sugere que, neste caso, o valor obtido pela medida de revocação seja mais significativo do que o da precisão.

Com isso, para as bases FunCat, como mostra a Figura 5.1, o algoritmo HLCS-Multi possui em quase todos os casos um alto valor de revocação e superando a curva Clus-HMC quando analisado em determinados limiares. Com relação as bases GO, como mostra a Figura 5.2, o algoritmo possui em determinadas bases um valor razoável de

revocação acima dos 0.4 pontos e também em algumas bases uma posição favorável do ponto em relação a curva Clus-HMC. A Curva PR utiliza a relação entre as medidas de Precisão plotada no eixo Y e as medidas de Revocação plotadas no eixo X.

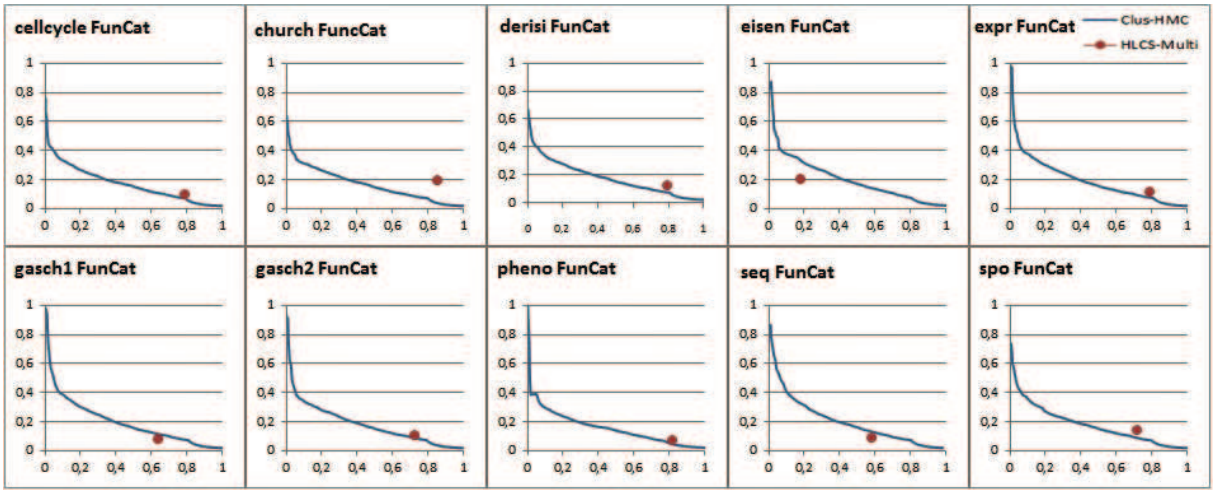

Figura 5.1: Gráfico Precisão Revocação - Bases FunCat

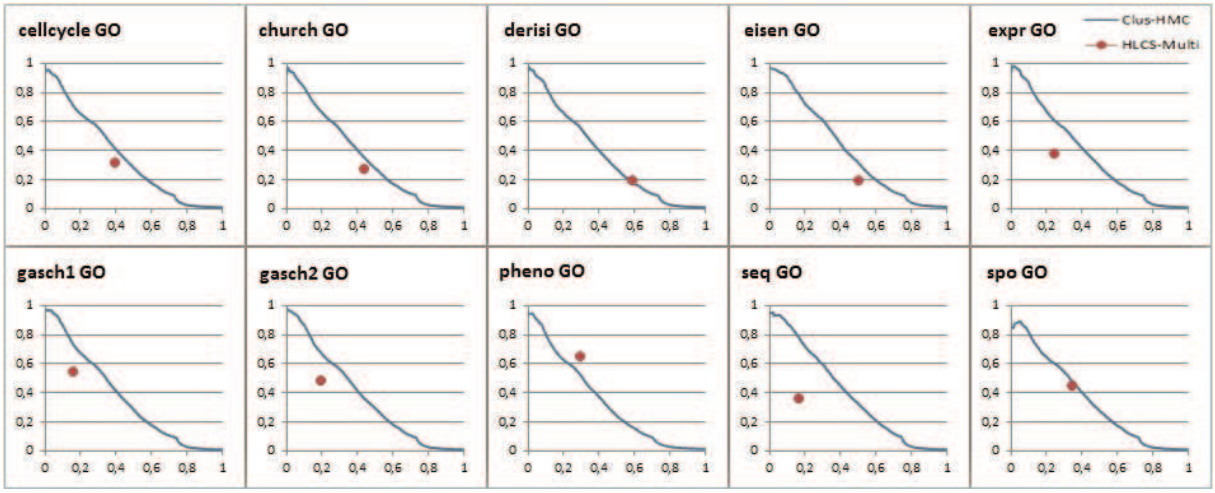

Figura 5.2: Gráfico Precisão Revocação - Bases GO

5.5 Considerações Finais do Capítulo

Neste capítulo foram apresentados os resultados obtidos com os modelos HLCS propostos. A comparação do algoritmo HLCS-Local com o método de classificação RIPPER, mostrou durante os que o HLCS-Local, apesar de não considerar a hierarquia dos dados na fase treinamento, foi capaz de tratar melhor este problema. Os resultados mostram que o HLCS-Local foi significativamente superior ao RIPPER em alguma medidas.

A comparação do algoritmo HLCS-Tree com o método GMNB, mostrou que o HLCS-Tree teve uma performance melhor nas bases de dados do grupo das proteínas tipo GPCR, provavelmente devido ao fato que estas bases tem uma melhor distribuição das classes nos diferentes níveis da hierarquia. A principal vantagem do modelo HLCS-Tree com relação a abordagem GMNB é a forma com o qual o modelo apresenta os resultados gerados. Enquanto o GMNB apresenta um modelo baseado em probabilidades, o HLCS-Tree gera um conjunto de regras, tornando o conhecimento mais fácil de ser compreendido pelas comunidades médica e científica.

Na comparação entre o algoritmo HLCS-DAG e a abordagem hAnt-Miner, mostrou que não existem diferenças significativas entre os dois classificadores. Entretanto os valores mostram uma pequena superioridade do HLCS-DAG em duas das quatro bases de dados, chamadas de 'DS1 AA' e 'DS2 AA'.

Na comparação utilizando o HLCS-Multi, os resultados mostram um superioridade significativamente superior quando comparado com os modelos HLCS-Tree e o HLCS-DAG. Na comparação com o algoritmo Clus-HMC, o HLCS-Multi teve sua avaliação prejudicada pelo fato dos autores do mesmo, publicarem e disponibilizarem através do algoritmo somente resultados baseados em medidas binárias. Mesmo assim, foi possível analisar utilizando os gráficos de Precisão-Revocação das bases de dados FunCat e GO, que o HLCS-Multi pode ser considerado superior quando comparado com determinados limiares ds Clus-HMC.

Capítulo 6

Conclusão

Há quase 10 anos o projeto genoma foi concluído com extremo sucesso, sequenciando 99% do genoma humano, com uma precisão de 99,99%. Contudo, estimativas mostram que apenas 2% do material genético humano é composto de genes que codificam instruções para a síntese de proteínas, o restante parece não conter instruções para a formação de proteínas, e existe provavelmente por razões estruturais. Além disso, de todos os genes que tiveram sua sequência determinada, apenas 50% aproximadamente codificam para proteínas de função conhecida.

O projeto genoma produziu uma enorme quantidade de informações, mas também mostrou que muito ainda precisa ser entendido. Mais de 100.000 pessoas morrem anualmente por apresentarem respostas adversas a medicamentos que podem ser benéficos para outras. Cerca de 2,2 milhões de pessoas apresentam fortes reações a um medicamento, enquanto outras não respondem. Entretanto, o progresso da genômica e o potencial de suas aplicações permite predizer que a biologia será uma das ciências mais importantes no século 21, auxiliando no desenvolvimento de fármacos muito mais potentes, assim como a compreensão de diversas doenças genéticas humanas. Além disso, a lista de aplicações potenciais passa por áreas como a medicina molecular, agricultura, desenvolvimento de novas fontes de energia, avaliação dos riscos de saúde de indivíduos expostos a radiações, busca por doadores que possuam órgãos compatíveis com os de receptores, entre outras aplicações extremamente importantes para a melhoria de vida da população.

Para contribuir com este processo e auxiliar com que as informações obtidas através do projeto genoma possam se transformar em conhecimento, esta tese apresentou uma nova proposta para classificar e predizer funções de proteínas utilizando a mineração de dados.

A predição da função de proteínas através da mineração de dados trabalha com informações de diversos tipos de elementos da proteína, como por exemplo: atributos

de aminoácidos, estrutura, fusão de genes, proximidade de cromossomos, padrões filogenéticos, entre outros. A complexidade deste tipo de aplicação vem da própria estrutura de organização da proteína. Atualmente, é cada vez mais frequente o uso de ontologias para organização de funções de proteínas, que descrevem estas funções utilizando uma hierarquia estruturada em árvores, como é o caso do modelo FunCat, ou em DAG, como na Ontologia Gênica.

Deste modo, esta tese propôs um novo método baseado no modelo dos Sistemas Classificadores (LCS) para a classificação global hierárquica multirrótulo da função de proteínas, chamado de Hierarchical Learning Classifier System - Multilabel (HLCS-Multi). O uso do LCS se deu principalmente pelo fato deste método gerar seus resultados em um formato de regras *SE-ENTÃO*, que são representações mais fáceis de serem interpretadas.

Em geral, o sistema HLCS-Multi cria sua população de classificadores através da análise dos exemplos do ambiente, que representa os dados da base de treinamento. A medida que o HLCS-Multi interage com o ambiente, por meio dos componente de desempenho e cessão de créditos, os classificadores são avaliados e recebem um retorno na forma de uma recompensa que impulsiona o processo de aprendizagem. Este processo de aprendizagem tem a intervenção do componente de avaliação, que ajuda no cálculo da recompensa avaliando as predições do sistema conforme a sua hierarquia. No fim desta etapa, os classificadores passam por um mecanismo de evolução através do componente AG, que utiliza algoritmos genéticos como base para a melhoria do conhecimento atual sistema.

Como forma de analisar os resultados obtidos pelo HLCS-Multi, foram feitas comparações com os métodos HLCS-Tree e HLCS-DAG, que são métodos desenvolvidos pelo autor, e com o algoritmo Clus-HMC proposto em (VENS et al., 2008). Os testes entre o HLCS-Multi contra os algoritmos HLCS-Tree e HLCS-DAG, mostra uma superioridade do HLCS-Multi em relação aos dois outros métodos. Esta superioridade se mostra principalmente pelas bases de dados utilizadas nos experimentos. Tanto as bases FunCat organizadas em uma estrutura em árvore, quanto as bases da GO organizadas em uma estrutura em DAG, são de grande complexidade por apresentarem dados multirrótulo e de grande profundidade na hierarquia. Com isso, pelo fato do HLCS-Multi ser desenvolvido especificamente para estes casos, os resultados comprovam a eficiência do algoritmo proposto.

O teste contra o Clus-HMC teve sua avaliação prejudicada pelo fato dos autores do mesmo, publicarem e disponibilizarem através do algoritmo somente resultados baseados em medidas binárias. Mesmo assim, foi possível analisar utilizando os gráficos de Precisão-Revocação das bases de dados FunCat e GO, que o HLCS-Multi pode ser considerado

superior quando comparado com determinados limiares ds Clus-HMC.

Sendo o Clus-HMC um método desenvolvido em 2008 e que vem sendo aperfeiçoado nos últimos anos, acredita-se que o HLCS-Multi possa, quando avaliado com medidas hierárquicas, também se tornar referência em pesquisas relacionadas com a predição de bases hierárquicas multirrótulo.

6.1 Contribuições e Trabalhos Futuros

O algoritmo HLCS-Multi atendeu aos requisitos necessários para realizar o tarefa de predição da função de proteínas. Baseado no modelo LCS, o HLCS-Multi é um algoritmo global hierárquico multirrótulo, que realiza constantes iterações com os exemplos de treinamento, fazendo com que o modelo de classificação se torne mais flexível e com uma maior capacidade de adaptação a novos conjuntos de dados. A sua forma de demostrar os resultados obtidos, contribui para uma melhor compreensão das informações por parte das comunidades médica e científica.

Um dos problemas observados inicialmente com relação ao LCS, foi a grande quantidade de variáveis que deveriam ser inicializadas antes da execução do algoritmo, sendo algumas delas, como o tamanho da população e a quantidade de execuções do modelo, fundamentais para o resultado final da predição. Este problema foi minimizado a cada modelo do HLCS, e no HLCS-Multi somente seis variáveis necessitam ser inicializadas pelo usuário.

A função do componente hierárquico foi fundamental para se trabalhar com bases hierárquicas. Através da análise hierárquica das predições é possível tornar a recompensa dos classificadores mais dinâmica. Além disso, como a proteína é classificada através de famílias, sugere-se que a ferramenta que conseguir prever o ramo ao qual pertence determinada sequência estará de certa forma contribuindo para uma maior agilidade nas pesquisas para o descobrimento de novas proteínas

O trabalho contribuiu também apresentando três versões desenvolvidas até se chegar no modelo final do HLCS-Multi. A primeira versão foi o HLCS-Local, que apresenta uma solução local por nó para trabalhar com bases hierárquicas. A segunda versão foi o HLCS-Tree, que utiliza uma abordagem global para bases hierárquicas estruturadas em formato de árvores e a terceira versão foi o HLCS-DAG, que também utiliza uma abordagem global mas com a capacidade de trabalhar com bases hierárquicas estruturadas no formato DAG. Cada modelo HLCS cooperou para a elaboração de um artigo, sendo todos aprovados em congressos internacionais.

Como proposta de trabalhos futuros, acredita-se que as medidas de avaliação dos

algoritmos de classificação, atualmente utilizadas pelos autores das principais ferramentas já propostas, deva ser revisto. O uso de medidas de avaliação binária para algoritmos hierárquicos faz com que não se tenha uma real análise das predições realizadas. Alternativas como as já apresentadas como a precisão e revocação hierárquicas devem ser utilizadas e aprimoradas.

Além disso, o HLCS-Multi precisa passar por uma minuciada revisão para diminuir determinadas repetições ao longo do processo de treinamento afim de melhorar seu desempenho. Testes com outras bases de dados também serão de grande importância para um melhor refinamento do processo. Mudanças nos valores das variáveis de inicialização deverão ser feitas para analisar o comportamento do algoritmo.

A determinação da função de um proteína é uma tarefa árdua, e deve ser realizada por especialistas. Até a completa definição da função de uma nova sequência, anos de pesquisas e testes são consumidos. Qualquer auxílio que uma ferramenta confiável possa trazer para os cientistas, irá contribuir na diminuição do tempo das pesquisas e consequentemente no desenvolvimento de novas alternativas para a melhoria da saúde e qualidade de vida da população. Sendo esta a intenção deste trabalho.

Referências Bibliográficas

AIOLLI, F.; SPERDUTI, A. Multiclass classification with multi-prototype support vector machines. *J. Mach. Learn. Res.*, JMLR.org, v. 6, p. 817–850, dez. 2005. ISSN 1532-4435.

ALTSCHUL, S. F. et al. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Research*, v. 25, n. 17, p. 3389–3402, 1997.

ALVES, R. T. *Um Sistema Imunológico Artificial para Classificação Hierárquica e Multi-Label de Funções de Proteínas*. 219 p. Tese (Doutorado) — Universidade Tecnológica Federal do Paraná, 2010.

ALVES, R. T.; DELGADO, M. R.; FREITAS, A. A. Multi-label hierarchical classification of protein functions with artificial immune systems. Springer-Verlag, Berlin, Heidelberg, p. 1–12, 2008.

ASHBURNER, M. et al. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*, v. 25, n. 1, p. 25–9, maio 2000.

BARD, J. B. L.; RHEE, S. Y. Ontologies in biology: design, applications and future challenges. *Nature Reviews Genetics*, Nature Publishing Group, v. 5, n. 3, p. 213–222, mar. 2004. ISSN 1471-0056.

BARUTCUOGLU, Z.; SCHAPIRE, R. E.; TROYANSKAYA, O. G. Hierarchical multi-label prediction of gene function. *Bioinformatics*, Oxford University Press, Oxford, UK, v. 22, n. 7, p. 830–836, abr. 2006. ISSN 1367-4803.

BLOCKEEL, H. et al. Decision trees for hierarchical multilabel classification: A case study in functional genomics. In: *Knowledge Discovery in Databases:PKDD 2006, 10th European Conference on Principles and Practice of Knowledge Discovery in Databases, Proceedings*. [S.l.]: Springer, 2006. (Lecture Notes in Artificial Intelligence, v. 4213), p. 18–29.

BUTZ, M. V.; GOLDBERG, D. E.; STOLZMANN, W. The anticipatory classifier system and genetic generalization. Kluwer Academic Publishers, Hingham, MA, USA, v. 1, p. 427–467, November 2002. ISSN 1567-7818.

BUTZ, M. V. et al. Toward a theory of generalization and learning in xcs. *IEEE Trans. Evolutionary Computation*, p. 28–46, 2004.

BUTZ, M. V.; STOLZMANN, W. An algorithmic description of acs2. In: LANZI, P.; STOLZMANN, W.; WILSON, S. (Ed.). *Advances in Learning Classifier Systems*. [S.l.]: Springer Berlin / Heidelberg, 2002. (Lecture Notes in Computer Science, v. 2321), p. 361–390. ISBN 978-3-540-43793-2.

CAMPBELL, M. K. *Bioquímica*. [S.l.]: ArtMed, 2000. ISBN 85-7307-673-3.

CARVALHO, A. C. P. L. F.; FREITAS, A. A. A tutorial on multi-label classification techniques. In: ABRAHAM, A.; HASSANIEN, A.-E.; EL, V. S. (Ed.). *Foundations of Computational Intelligence Volume 5*. [S.l.]: Springer Berlin / Heidelberg, 2009, (Studies in Computational Intelligence, v. 205). p. 177–195. ISBN 978-3-642-01535-9.

CERRI, R.; CARVALHO, A. C. P. L. F. Hierarchical multilabel classification using top-down label combination and artificial neural networks. *Neural Networks, Brazilian Symposium on*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 253–258, 2010.

CERRI, R.; CARVALHO, A. C. P. L. F. New top-down methods using svms for hierarchical multilabel classification problems. In: *International Joint Conference on Neural Networks*. [S.l.]: IEEE, 2010. p. 1–8.

CERRI, R. et al. Classificação Hierárquica de Proteínas Utilizando Abordagens Top-Down Big-Bang. *IV Workshop em Algoritmos e Aplicações de Mineração de Dados*, IV Workshop em Algoritmos e Aplicações de Mineração de Dados, Campinas, p. 46–54, 2008.

CERRI, R.; CARVALHO, A. C. P. L. F.; FREITAS, A. A. Adapting non-hierarchical multilabel classification methods for hierarchical multilabel classification. *Intell. Data Anal.*, v. 15, n. 6, p. 861–887, 2011.

CIOS, K. J. et al. *Data Mining: A Knowledge Discovery Approach*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007. ISBN 0387333339.

CLARE, A.; KING, R. D. Knowledge discovery in multi-label phenotype data. In: *Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge*

Discovery. London, UK, UK: Springer-Verlag, 2001. (PKDD '01), p. 42–53. ISBN 3-540-42534-9.

CLARE, A.; KING, R. D. Predicting gene function in saccharomyces cerevisiae. *Bioinformatics*, v. 19, p. 42–49, 2003.

CLIFF, D.; ROSS, S. Adding temporary memory to zcs. *Adapt. Behav.*, MIT Press, Cambridge, MA, USA, v. 3, n. 2, p. 101–150, set. 1994. ISSN 1059-7123.

COHEN, W. W. Fast Effective Rule Induction. In: PRIEDITIS, A.; RUSSELL, S. (Ed.). *Proc. of the 12th International Conference on Machine Learning*. Tahoe City, CA: Morgan Kaufmann, 1995. p. 115–123.

COMITÉ, F. D.; GILLERON, R.; TOMMASI, M. Learning multi-label alternating decision trees from texts and data. In: *Proceedings of the 3rd international conference on Machine learning and data mining in pattern recognition*. Berlin, Heidelberg: Springer-Verlag, 2003. (MLDM'03), p. 35–49. ISBN 3-540-40504-6.

DAM, H. H. et al. Neural-based learning classifier systems. *IEEE Trans. on Knowl. and Data Eng.*, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 20, n. 1, p. 26–39, jan. 2008. ISSN 1041-4347.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In: *Proceedings of the 23rd international conference on Machine learning*. New York, NY, USA: ACM, 2006. (ICML '06), p. 233–240. ISBN 1-59593-383-2.

DEJONG, K. A.; SPEARS, W. M.; GORDON, D. F. Using genetic algorithms for concept learning. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 13, n. 2-3, p. 161–188, nov. 1993. ISSN 0885-6125.

DEMŝAR, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, JMLR.org, v. 7, p. 1–30, dez. 2006. ISSN 1532-4435.

DOUGHERTY, J.; KOHAVI, R.; SAHAMI, M. Supervised and unsupervised discretization of continuous features. In: *MACHINE LEARNING: PROCEEDINGS OF THE TWELFTH INTERNATIONAL CONFERENCE*. [S.l.]: Morgan Kaufmann, 1995. p. 194–202.

DRUGOWITSCH, J. *Learning classifier systems from first principles*. 378 p. Tese (Doutorado) — University of Bath, 2007.

FAWCETT, T. An introduction to roc analysis. *Pattern Recogn. Lett.*, Elsevier Science Inc., New York, NY, USA, v. 27, n. 8, p. 861–874, jun. 2006. ISSN 0167-8655.

FAYYAD, U. M. et al. (Ed.). *Advances in knowledge discovery and data mining*. Menlo Park, CA, USA: American Association for Artificial Intelligence, 1996. ISBN 0-262-56097-6.

FREITAS, A. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002. ISBN 3540433317.

FREITAS, A. A.; CARVALHO, A. C. P. L. F. A tutorial on hierarchical classification with applications in bioinformatics. In: _____. [S.l.]: Idea Group, 2007. Research and Trends in Data Mining Technologies and Applications, cap. VII, p. 175–208. ISBN 159904272X.

FREITAS, A. A.; WIESER, D. C.; APWEILER, R. On the importance of comprehensible classification models for protein function prediction. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 7, n. 1, p. 172–182, jan. 2010. ISSN 1545-5963.

FREUND, Y.; MASON, L. The alternating decision tree learning algorithm. In: *Proceedings of the Sixteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. (ICML '99), p. 124–133. ISBN 1-55860-612-2.

FRIEDBERG, I. Automated protein function prediction the genomic challenge. *Briefings in Bioinformatics*, v. 7, n. 3, p. 225–242, 2006.

GAO, Y. et al. Learning classifier system ensemble for data mining. In: *Proceedings of the 2005 workshops on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2005. (GECCO '05), p. 63–66.

GéRARD, P.; SIGAUD, O. Designing efficient exploration with macs: modules and function approximation. In: *Proceedings of the 2003 international conference on Genetic and evolutionary computation: PartII*. Berlin, Heidelberg: Springer-Verlag, 2003. (GECCO'03), p. 1882–1893. ISBN 3-540-40603-4.

GéRARD, P.; STOLZMANN, W.; SIGAUD, O. Yacs: a new learning classifier system using anticipation. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, Springer Berlin / Heidelberg, v. 6, p. 216–228, 2002. ISSN 1432-7643. 10.1007/s005000100117.

GEYER-SCHULZ, A. Holland classifier systems. *SIGAPL APL Quote Quad*, ACM, New York, NY, USA, v. 25, n. 4, p. 43–55, jun. 1995. ISSN 0163-6006.

GOLDBERG, D. E. *Genetic Algorithms in Search, Optimization and Machine Learning*. 1st. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1989. ISBN 0201157675.

HAMZEH, A.; RAHMANI, A. A new architecture of xcs to approximate real-valued functions based on high order polynomials using variable-length ga. In: *Proceedings of the Third International Conference on Natural Computation - Volume 03*. Washington, DC, USA: IEEE Computer Society, 2007. (ICNC '07), p. 515–519.

HOLDEN, N.; FREITAS, A. A. A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In: *Swarm Intelligence Symposium. SIS 2005. Proceedings 2005 IEEE*. [S.l.]: IEEE Press, 2005. p. 100–107.

HOLDEN, N.; FREITAS, A. A. Hierarchical classification of g-protein-coupled receptors with a pso/aco algorithm. IEEE Press, p. 77–84, 2006.

HOLLAND, J. H. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1992. ISBN 0262082136.

HURST, J.; BULL, L. A self-adaptive neural learning classifier system with constructivism for mobile robot control. In: YAO, X. et al. (Ed.). *Parallel Problem Solving from Nature - PPSN VIII*. [S.l.]: Springer Berlin / Heidelberg, 2004, (Lecture Notes in Computer Science, v. 3242). p. 942–951. ISBN 978-3-540-23092-2.

JENSEN L. J. GUPTA, R.; HENRIK, H. S.; BRUNAK, S. Prediction of human protein function according to Gene Ontology categories. *Bioinformatics*, v. 19, n. 5, p. 635–642, mar. 2003.

JUNGCK, J. R. et al. Bioinformatics education dissemination with an evolutionary problem solving perspective. *Briefings in bioinformatics*, v. 11, n. 6, p. 570–81, nov. 2010. ISSN 1477-4054.

KING, R. D.; WISE, P. H.; CLARE, A. Confirmation of data mining based predictions of protein function. *Bioinformatics*, Oxford University Press, Oxford, UK, v. 20, n. 7, p. 1110–1118, maio 2004. ISSN 1367-4803.

KIRITCHENKO, S.; MATWIN, S.; FAMILI, A. F. Functional annotation of genes using hierarchical text categorization. In: *Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology (held at ISMB-05*. [S.l.: s.n.], 2005.

KOUMOUSIS, V. K.; KATSARAS, C. P. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. *IEEE Transactions on Evolutionary Computation*, v. 10, n. 1, p. 19–28, fev. 2006.

KOVACS, T. Learning classifier systems resources. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, Springer, v. 6, n. 3, p. 240–243, 2002. ISSN 1432-7643.

LAGREID, A. et al. Predicting gene ontology biological process from temporal gene expression patterns. *Genome research*, v. 13, n. 5, p. 965–79, maio 2003. ISSN 1088-9051.

LANZI, P. L. Adding memory to xcs. In: *Proceedings of the IEEE Conference on Evolutionary Computation (ICEC98*. [S.l.]: IEEE Press, 1998.

LANZI, P. L. Learning classifier systems: then and now. *Evolutionary Intelligence*, Springer Berlin / Heidelberg, v. 1, p. 63–82, 2008. ISSN 1864-5909.

LANZI, P. L.; LOIACONO, D. Classifier systems that compute action mappings. In: *Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007. (GECCO '07), p. 1822–1829. ISBN 978-1-59593-697-4.

LESK, A. M. *Introdução a Bioinformática*. [S.l.]: Artmed, 2008.

LINDEN, R. *Algoritmos Genéticos*. [S.l.]: Brasport, 2008. 428 p. ISBN 9788574523736.

MARCOTTE, E. M. Computational genetics: finding protein function by nonhomology methods. *Current opinion in structural biology*, v. 10, n. 3, p. 359–65, jun. 2000.

OLSON, D. L.; DELEN, D. *Advanced Data Mining Techniques*. 1st. ed. [S.l.]: Springer Publishing Company, Incorporated, 2008. ISBN 3540769161, 9783540769163.

ORRIOLS, A. P. *New Challenges in Learning Classifier Systems: Mining Rarities and Evolving Fuzzy Models*. Tese (Doutorado) — Universitat Ramon Llull, Barcelona, Catalonia, Spain, Barcelona, 2008.

ORRIOLS-PUIG, A.; CASILLAS, J.; BERNADó-MANSILLA, E. Fuzzy-ucs: preliminary results. In: *Proceedings of the 2007 GECCO conference companion on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007. (GECCO '07), p. 2871–2874. ISBN 978-1-59593-698-1.

OTERO, F. E.; FREITAS, A. A.; JOHNSON, C. G. A hierarchical classification ant colony algorithm for predicting gene ontology terms. In: *Proceedings of the 7th European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics*. Berlin, Heidelberg: Springer-Verlag, 2009. (EvoBIO '09), p. 68–79. ISBN 978-3-642-01183-2.

OTERO, F. E.; FREITAS, A. A.; JOHNSON, C. G. A hierarchical multi-label classification ant colony algorithm for protein function prediction. *Memetic Computing*, v. 2, n. 3, p. 165–181, 2010.

PEARSON, W. R.; LIPMAN, D. J. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, v. 85, n. 8, p. 2444–8, abr. 1988. ISSN 0027-8424.

RIGDEN, D. J.; MELLO, L. V. Anotação Funcional e Computacional de Proteínas. *biotecnologia.com.br*, v. 25, p. 64–70, 2002.

ROMAO, L. M.; NIEVOLA, J. C. Prediction of Protein Function Using Learning Classifier Systems. *IADIS International Conference on Applied Computing*, IADIS Press, Rio de Janeiro, BR, p. 395–401, 2011.

ROMAO, L. M.; NIEVOLA, J. C. Hierarchical Classification of Gene Ontology with Learning Classifier Systems. *Proceedings of IBERAMIA 2012 Lecture Notes in Computer Science/Lecture Notes in Artificial Intelligence LNCS/LNAI 7637*, Springer-Verlag, Berlin Heidelberg, p. 120–129, 2012.

ROMAO, L. M.; NIEVOLA, J. C. Predicting GPCR and Enzymes Function with a Global Approach Based on LCS. *Proceedings of the 2012 IEEE 12th International Conference on Bioinformatics and Bioengineering (BIBE)*, Larnaca, Cyprus, 2012.

RUEPP, A. et al. The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. *Nucleic acids research*, v. 32, n. 18, p. 5539–45, jan. 2004.

SALZBURGER, W. et al. Annotation of expressed sequence tags for the East African cichlid fish Astatotilapia burtoni and evolutionary analyses of cichlid ORFs. *BMC genomics*, v. 9, p. 96, jan. 2008. ISSN 1471-2164.

SECKER, A. et al. An experimental comparison of classification algorithms for the hierarchical prediction of protein function. In: *3rd UK Data mining and Knowledge Discovery Symposium (UKKDD 2007), Canterbury*. [S.l.]: Citeseer, 2007. p. 13–18.

SIGAUD, O.; WILSON, S. W. Learning classifier systems: a survey. *Soft Comput.*, Springer-Verlag, Berlin, Heidelberg, v. 11, n. 11, p. 1065–1078, maio 2007. ISSN 1432-7643.

SILLA, C. N.; FREITAS, A. A. A Global-Model Naive Bayes Approach to the Hierarchical Prediction of Protein Functions. *2009 Ninth IEEE International Conference on Data Mining*, p. 992–997, dez. 2009.

SILLA, C. N.; FREITAS, A. A. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 22, n. 1-2, p. 31–72, jan. 2011. ISSN 1384-5810.

STOLZMANN, W. Anticipatory classifier systems. *Genetic Programming '98*, University of Wisconsin, Morgan Kaufmann, Madison, Wisconsin, p. 658–664, 1998.

SU, C. et al. A novel approach for prediction of multi-labeled protein subcellular localization for prokaryotic bacteria. *2005 IEEE Computational Systems Bioinformatics Conference - Workshops*, IEEE Computer Society, Los Alamitos, CA, USA, v. 0, p. 79–82, 2005.

SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. [S.l.]: The MIT Press, 1998. Hardcover. ISBN 0262193981.

TAN, P.; STEINBACH, M.; KUMAR, V. *Introduction to Data Mining, (First Edition)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2005. ISBN 0321321367.

TOMLINSON, A.; BULL, L. A corporate classifier system. In: EIBEN, A. et al. (Ed.). *Parallel Problem Solving from Nature - PPSN V*. [S.l.]: Springer Berlin / Heidelberg, 1998, (Lecture Notes in Computer Science, v. 1498). p. 550–559. ISBN 978-3-540-65078-2.

TOMLINSON, A.; BULL, L. A corporate xcs. In: LANZI, P.; STOLZMANN, W.; WILSON, S. (Ed.). *Learning Classifier Systems*. [S.l.]: Springer Berlin / Heidelberg, 2000, (Lecture Notes in Computer Science, v. 1813). p. 195–208.

TU, K. et al. Learnability-based further prediction of gene functions in gene ontology. *Genomics*, v. 84, n. 6, p. 922 – 928, 2004.

UNOLD, O. Grammar-based classifier system: a universal tool for grammatical inference. *W. Trans. on Comp.*, World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, v. 7, n. 10, p. 1584–1593, out. 2008. ISSN 1109-2750.

URBANOWICZ, R. J.; MOORE, J. H. Learning classifier systems: a complete introduction, review, and roadmap. *J. Artif. Evol. App.*, Hindawi Publishing Corp., New York, NY, United States, v. 2009, p. 1:1–1:25, jan. 2009. ISSN 1687-6229.

VALENTINI, G. True path rule hierarchical ensembles. In: *Proceedings of the 8th International Workshop on Multiple Classifier Systems*. Berlin, Heidelberg: Springer-Verlag, 2009. (MCS '09), p. 232–241. ISBN 978-3-642-02325-5.

VENS, C. et al. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 73, n. 2, p. 185–214, nov. 2008. ISSN 0885-6125.

WILSON, S. W. Zcs: A zeroth level classifier system. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 2, n. 1, p. 1–18, mar. 1994. ISSN 1063-6560.

WILSON, S. W. Classifier fitness based on accuracy. *Evol. Comput.*, MIT Press, Cambridge, MA, USA, v. 3, n. 2, p. 149–175, jun. 1995. ISSN 1063-6560.

WILSON, S. W. Get real! xcs with continuous-valued inputs. In: *Learning Classifier Systems, From Foundations to Applications*. London, UK, UK: Springer-Verlag, 2000. p. 209–222. ISBN 3-540-67729-1.

WILSON, S. W. Function approximation with a classifier system. *Genetic and Evolutionary Computation Conference, GECCO 2001*, Morgan Kaufmann, San Francisco, CA, p. 974–981, 2001.

WU, F.; ZHANG, J.; HONAVAR, V. Learning Classifiers Using Hierarchically Structured Class Taxonomies. *Lecture notes in computer science*, v. 3607, p. 313–320, jan. 2005.