UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

LIN CHI YU

CRIAÇÃO DE UM MÉTODO PARA REDUZIR A DIMENSÃO DA HIERARQUIA DE CLASSES DURANTE O TREINAMENTO DO ALGORITMO MHC-CNN

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA
2019

LIN CHI YU

CRIAÇÃO DE UM MÉTODO PARA REDUZIR A DIMENSÃO DA HIERARQUIA DE CLASSES DURANTE O TREINAMENTO DO ALGORITMO MHC-CNN

Trabalho de Conclusão de Curso apresentada como requisito parcial à obtenção do título de Bacharel em Ciência da Computação, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof.ª Dra. Helyane B. Borges

PONTA GROSSA

2019

Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Ponta Grossa
Diretoria de Graduação e Educação Profissional
Departamento Acadêmico de Informática
Bacharelado em Ciência da Computação

TERMO DE APROVAÇÃO

CRIAÇÂO DE UM MÉTODO PARA REDUZIR A DIMENSÃO DA HIERARQUIA DE CLASSES DURANTE O TREINAMENTO DO ALGORITMO MHC-CNN

por

LIN CHI YU

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 3 de junho de 2019 como requisito parcial para a obtenção do título de Bacharel em Ciência da Computação. O candidato foi arguido pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof.ª Dra. Helyane Bronoski Borges
Orientador(a)

_____
Prof.ª Dra. Simone de Almeida
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Membro titular

_____
Prof. MSc. Geraldo Ranthum
Responsável pelo Trabalho de Conclusão de Curso

_____
Prof. MSc. Saulo Jorge Beltrão de Queiroz
Coordenador do curso

RESUMO

LIN, CHI YU. **Criação De Um Método para Reduzir a Dimensão da Hierarquia de Classes durante o Treinamento do Algoritmo MHC-CNN**. 2019. 61 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2019.

Classificação de dados é uma tarefa comum em Mineração de Dados. Alguns problemas de classificação precisam levar em consideração sua hierarquia de dados, estabelecendo uma ordem entre suas classes, esses problemas são chamados de classificação hierárquica. Este trabalho apresenta um método para reduzir a dimensão da hierarquia de classes na fase de treinamento do classificador hierárquico MHC-CNN. O método proposto é chamado de HNR e visa aprimorar o algoritmo MHC-CNN. Para avaliação do método, 10 bases de dados de função de proteína foram utilizadas e, os seus resultados, comparados com resultados do algoritmo MHC-CNN. Os estudos iniciais mostram que HNR é uma alternativa que pode ser usada para problemas de classificação hierárquica global.

Palavras-chave: Classificação Hierárquica. Classificação Multirrótulo. Grafo Acíclico Dirigido. Classificação de proteína. Classificação Global. Classificação Hierárquica Multirrótulo.

ABSTRACT

LIN, CHI YU. **Development of a Method to Reduce the Hierarchical Class Dimension during the Algorithm MHC-CNN Training Phase**. 2019. 61 p. Work of Conclusion Course (Graduation in Computer Science) - Federal Technology University - Paraná. Ponta Grossa, 2019.

Data classification is a common task in Data Mining. Some classification problems need to take into account the data hierarchical taxonomy, establishing an order between classes, those are called hierarchical classification problems. This work presents a method to reduce the dimension of the hierarchical classes in the MHC-CNN hierarchical classifier training phase. The method is called HNR and seek to improve the algorithm MHC-CNN. The proposed method is called HNR and attempt to improve the MHC-CNN. To evaluate the method, ten protein functions datasets is used and the results compared to MHC-CNN's results. Preliminary studies show that the HNR is an alternative that can be used for global hierarchical classification.

Keywords: Hierarchical Classification. Multi-label Classification. Directed Acyclic Graph. Classify protein. Global Classification. Multi-label Hierarchical Classification.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

DAG	*Directed Acyclic Graph*
HNR	*Hierarchical Neighbour Reductor*
MHC-CNN	*Multi-label Hierarchical Classification using a Competitive Neural Network*
SOM	*Self-Organized Map*

SUMÁRIO

1 INTRODUÇÃO

A AM (Aprendizagem de máquina) é um campo da inteligência artificial que dá aos computadores a capacidade de obter conclusões genéricas (funções) que resolvem um problema a partir de um conjunto de dados (FACELI, 2011). Existem duas abordagens principais: a aprendizagem supervisionada e a não supervisionada (FACELI, 2011).

A aprendizagem supervisionada funciona da seguinte maneira: os resultados esperados são passados para o algoritmo na fase de treinamento, possibilitando uma adaptação de acordo com a entrada e saída esperada (MONARD; BARANAUSKAS, 2002). Como um exemplo pode-se citar uma base de dados de música com seus respectivos estilos musicais, esses dados, as músicas e seus estilos, são passados ao algoritmo na fase de treinamento, onde é feito o relacionamento de ambos e adaptado de acordo com o acerto (estilos iguais) ou erros (estilos diferentes).

Na aprendizagem não supervisionada, as categorias não são previamente repassadas ao algoritmo, cabendo ao algoritmo decidir como será feito a divisão dos dados, tentando correlacionar os dados de acordo com suas características. Tem-se nesse tipo de aprendizagem métodos como agrupamento ou *clustering* de dados (MONARD; BARANAUSKAS, 2002).

A classificação dos dados é um desafio para a AM, pois a grande quantidade de rótulos aliados à grande quantidade de dados não garante que as hipóteses geradas por meio da inferência indutiva retenham a verdade (MONARD; BARANAUSKAS, 2002). Por isso são desenvolvidos e projetados algoritmos que procuram solucionar o problema, buscando melhorar os resultados obtidos, seja na taxa de acerto da classificação (porcentagem em que a classe predita pelo algoritmo foi correta), seja na velocidade (custo computacional) para apresentação do resultado.

Classificação de dados no mundo real são naturalmente problemas de classificação hierárquica, onde as classes estão organizadas em uma hierarquia de classes (normalmente uma árvore ou um Grafo Acíclico Dirigido) (SILLA; FREITAS, 2011). Um DAG (*Directed Acyclic Graph)* é um grafo dirigido onde as conexões entre os nós não formam ciclo.

O problema de classificação multirrótulo está presente em diversas áreas, tais como: Categorização de Texto (BENNETT; NGUYEN, 2009), Predição de Função de Proteína (VALENTINI, 2009) e (BORGES, 2012), Classificação de Gênero Musical (LI; OGIHARA, 2005), Classificação de Discurso Emocional (XIAO et al, 2007), Classificação de Fonema (DEKEL et al, 2004). As classificações dos dados podem ser realizadas por algoritmos baseados em diferentes abordagens.

Na bioinformática, por exemplo, o problema de predição da função que uma proteína exerce em uma célula não é um problema banal, visto que proteínas com estruturas parecidas podem acabar exercendo funções diferentes, assim classificadas em classes diferentes. Além disso, proteínas que realizam as mesmas funções podem ter muitas sequências de aminoácidos diferentes (LESK, 2008). A informática pode auxiliar nesse processo de classificação hierárquica por meio de algoritmos de aprendizagem.

Borges (2012) propôs um algoritmo de classificação hierárquica multirrótulo denominado de MHC-CNN (*Multi-label Hierarchical Classification using a Competitive Neural Network*) usando a abordagem de classificação global que funciona em árvores e DAG. O algoritmo tem como base redes neurais competitivas no qual os neurônios competem entre si para serem ativados e, ao final, apenas um neurônio de saída será o “vencedor”.

Este trabalho propõe o desenvolvimento de um novo método chamado HNR (*Hierarchicial Neighbour Reductor)* que visa reduzir a dimensão da hierarquia de classes durante o treinamento do algoritmo MHC-CNN.

1.1 DESCRIÇÃO DO PROBLEMA

O algoritmo MHC-CNN proposto por Borges (2012) tem como base redes neurais competitivas no qual os neurônios competem entre si para serem ativados. Essa rede neural possui uma quantidade de neurônios que é proporcional à quantidade de rótulos que os dados apresentam. Assim, em grandes bases de dados o custo computacional cresce proporcionalmente.

Com o objetivo de melhorar o resultado da predição do MHC-CNN e também seu custo computacional foi implantado um método de redução de número de neurônios atualizados durante o treinamento, esse método foi inspirado pelo

algoritmo SOM descrito por Teuvo Kohonen em seu livro *Self Organizing Maps* (1990).

1.2 OBJETIVOS

Esta Seção apresenta o objetivo geral e os objetivos específicos.

1.2.1 Objetivo Geral

O objetivo geral do trabalho é desenvolver um método para reduzir a dimensão da hierarquia de classes na fase de treinamento do classificador MHC-CNN.

1.2.2 Objetivos Específico

Como objetivos específicos têm-se:

- Realizar uma revisão sistemática sobre classificação hierárquica multirrótulo utilizando a abordagem global para dados estruturados em DAG.
- Desenvolver um algoritmo de redução de dimensionalidade e implantar no classificador MHC-CNN.
- Realizar experimentos em bases de dados hierárquicas com o classificador adaptado.
- Analisar e comparar estatisticamente os resultados obtidos.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado em 6 Capítulos. O Capítulo 2 apresenta a fundamentação teórica da classificação de dados. No Capítulo 3 é apresentado os trabalhos correlatos que serviram como base para a realização do trabalho por meio de um método de revisão sistemática realizada na literatura. O Capítulo 4 descreve o MHC-CNN e o método proposto denominado de HNR (*Hierarchical Neighbour Reductor*) desenvolvido neste trabalho. O Capítulo 5 apresenta os resultados obtidos

por meio dos experimentos. Por fim, o Capítulo 6 apresenta a conclusão e trabalhos futuros.

2 CLASSIFICAÇÃO DE DADOS

Neste Capítulo são apresentadas informações sobre a classificação de dados. A Seção 2.1 aborda os conceitos fundamentais referentes à classificação. Na Seção 2.2 são descritos os tipos de classificação. A Seção 2.3 aborda os conceitos de classificação hierárquica e, por fim, a Seção 2.4 apresenta as considerações do Capítulo.

2.1 CONCEITOS FUNDAMENTAIS

Classificação de dados é um processo de organização dos dados em categorias (classes), cada exemplo de dado é descrito por valores de características, ou atributos, e o rótulo da classe associada.

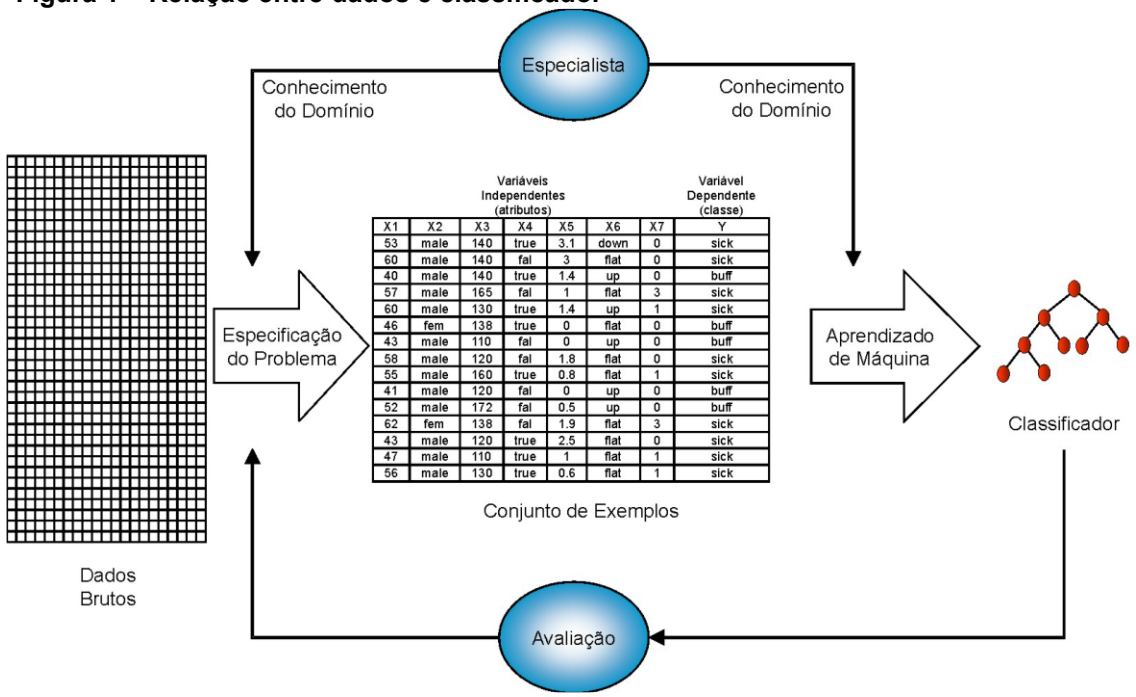
O objetivo do algoritmo de classificação é construir um classificador que possa determinar corretamente a classe de novos exemplos ainda não rotulados, ou seja, exemplos que não tenham o rótulo da classe (MONARD; BARANAUSKAS, 2002).

O processo de classificação é ilustrado na Figura 1. O conhecimento sobre o domínio pode ser usado para escolher os dados ou para fornecer alguma informação previamente conhecida como entrada ao indutor. Após induzido, o classificador é geralmente avaliado e o processo de classificação pode ser repetido, se necessário, por exemplo, adicionando outros atributos, exemplos ou mesmo ajustando alguns parâmetros no processo de indução (MONARD; BARANAUSKAS, 2002).

2.2 TIPOS DE CLASSIFICAÇÃO

A tarefa de classificação pode ser dividida em classificação plana, Figura 3b, na qual as classes estão separadas em apenas um único nível, e a classificação hierárquica, Figura 3a, em que classes podem possuir vários níveis, onde esses níveis definem, também, sua ancestralidade e dependência (SILLA; FREITAS, 2011).

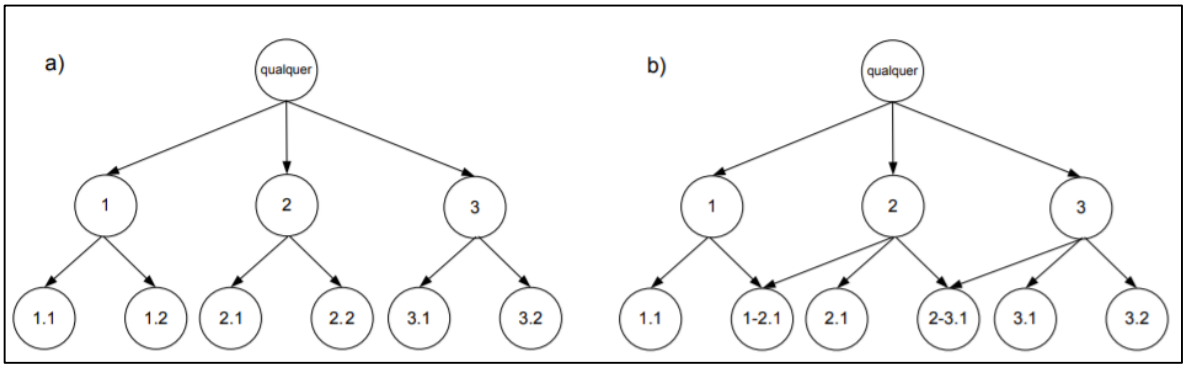
Figura 1 – Relação entre dados e classificador



Fonte: Monard e Baranauskas (2003)

Na classificação hierárquica, pode-se trabalhar em duas estruturas de grafos diferentes, a árvore e o Grafo Acíclico Dirigido (*Directed Acyclic Graph* - DAG). A diferença entre esses dois grafos é que a árvore é um grafo onde um vértice está conectado com apenas um nó ancestral (nó pai) como pode ser visto na Figura 2a, enquanto um vértice de uma DAG pode ter vários nós ancestrais, conforme a Figura 2b (SILLA; FREITAS, 2011).

Figura 2 – Exemplo de hierarquia de classe estruturada em árvore e em DAG



Fonte: Borges (2012)

O desafio da classificação no mundo real é que muitos são naturalmente problemas de classificação hierárquica, onde as classes estão organizadas em uma

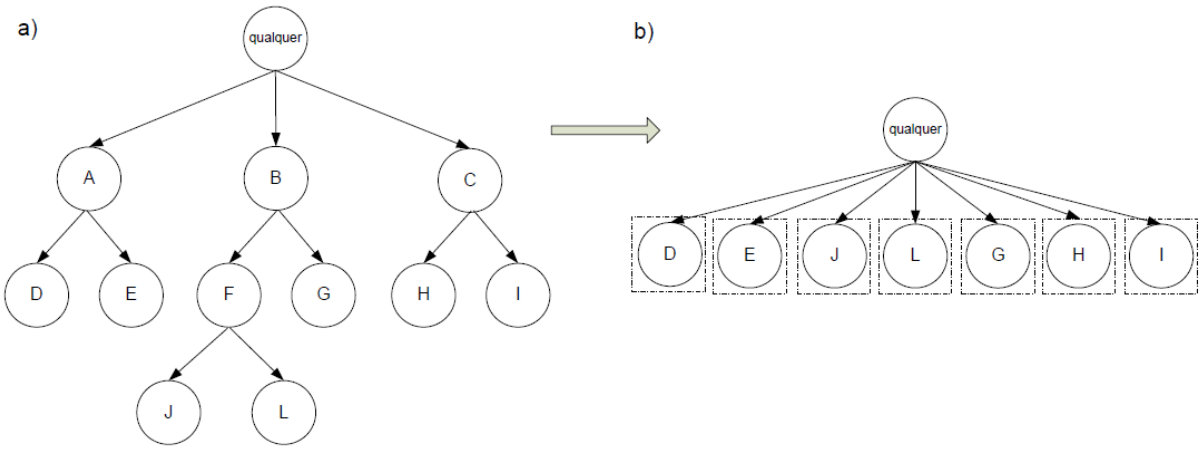
hierarquia de classes (normalmente uma árvore ou um Grafo Acíclico Dirigido) (SILLA; FREITAS, 2011). Na classificação plana essa relação entre as classes, que pode ser um fator importante, é perdida.

A geração dos rótulos para as classes hierárquicas pode ser feita por meio de um método automatizado, como proposto por Freitas (2007). Porém, para o escopo desse trabalho a classificação hierárquica para somente os problemas com classes hierárquicas previamente definidas, não para classes criadas por similaridades dos dados.

A classificação hierárquica pode ser separada quanto ao tipo de predição das classes na hierarquia, que podem ser: predição obrigatória em nós-folha (*Mandatory Leaf-Node Prediction*), onde as classes preditas devem sempre pertencer aos nós folhas, e predição opcional em nós-folha (*Non-Mandatory Leaf-Node Prediction*), onde o classificador pode considerar parar a classificação em qualquer nó em qualquer nível da hierarquia (FREITAS; CARVALHO, 2007).

É possível realizar a transformação de um problema de classificação hierárquica para um problema de classificação. Nesse caso, é desconsiderado o conceito de ancestralidade e descendência ignorando a hierarquia de classe. A Figura 3a mostra uma hierarquia de classes, sendo esta transformada em um problema de classificação plana, em que são considerados apenas os nós folha, conforme ilustra a Figura 3b.

Figura 3 – Exemplo de classificação hierárquica sendo transformada para classificação plana



Fonte: Adaptado de Borges (2012)

2.3 ABORDAGENS DA CLASSIFICAÇÃO HIERÁRQUICA

Para tratar problemas de classificação hierárquica existem na literatura duas abordagens: a classificação hierárquica local e a classificação hierárquica global (SILLA; FREITAS, 2011).

2.3.1 Classificação Hierárquica Local

A classificação hierárquica local consiste em resolver o problema em exemplos de classes em uma determinada região da hierarquia. Esse tipo de classificação apenas determina se um exemplo pode ou não ser associado à classe para a qual aquele classificador foi treinado. A classificação hierárquica local pode ser subdividida em três grupos.

- Classificação Hierárquica Local para Cada Nó

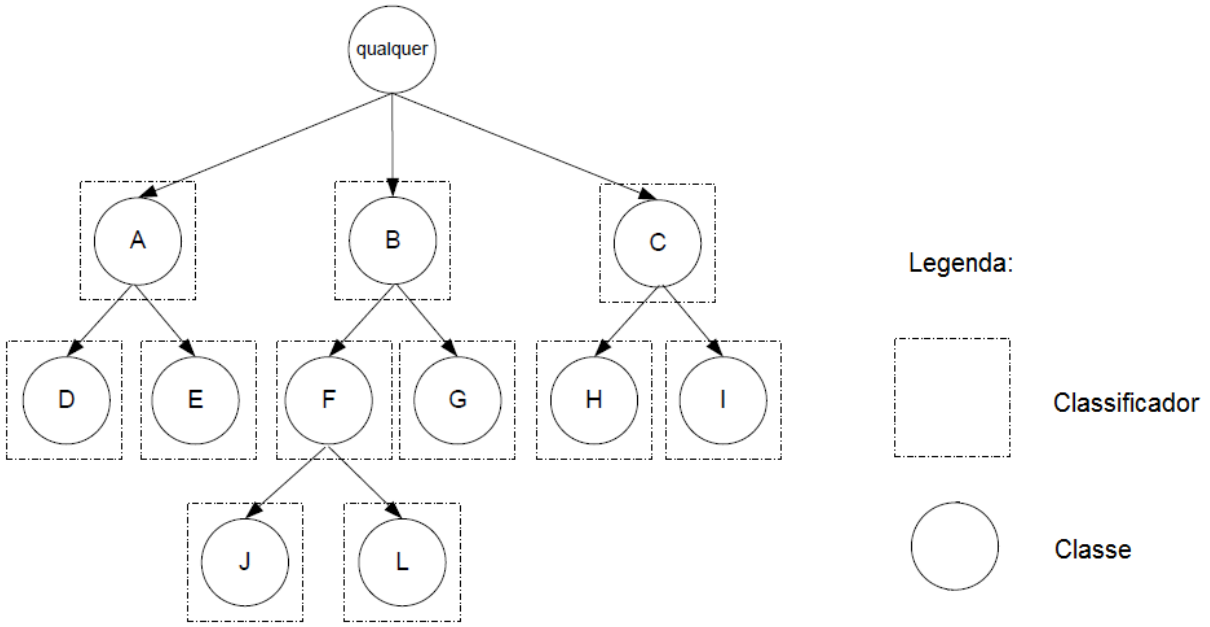
A classificação hierárquica local para cada nó consiste em treinar um classificador binário para cada nó da hierarquia de classes indicando se determinado exemplo pertence ou não sua classe que está sendo predita (SILLA; FREITAS, 2011). Dessa forma, a quantidade de classificadores locais é equivalente a quantidade de classes da hierarquia. A Figura 4 ilustra um exemplo desta abordagem. Nota-se que nessa figura existem 11 classes definidas pelas letras A, B, C, D, E, F, G, H, I, J e L. e a mesma quantidade de classificadores.

Um problema que esse tipo de abordagem pode apresentar é a possibilidade de classificar uma classe como “verdadeira” (pertencente à classe) porém classificar sua classe pai como “falso” (não pertence à classe), causando inconsistência na classificação em entre diferentes níveis (SILLA; FREITAS, 2011). Assim, é necessário utilizar algum método para corrigir esse tipo de inconsistência.

- Classificação em Nós Pais

Outro tipo de abordagem que utiliza informações locais é o treinamento de um classificador para cada um dos nós pais da hierarquia, esse classificador é treinado para diferenciar entre seus filhos qual será o próximo a ser classificado, utilizando o método *top-down*.

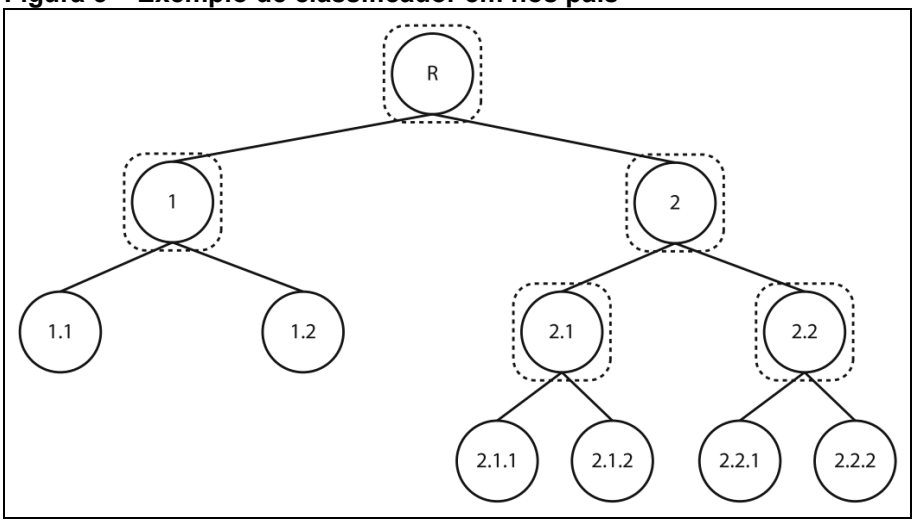
Figura 4 – Exemplo de classificação hierárquica local por nó



Fonte: Adaptado de Borges (2012)

Visualizando a árvore de hierarquias da Figura 5, caso o classificador do nível mais alto da hierarquia (que se encontra na classe R) atribua o exemplo de entrada para a classe 2. O classificador da classe 2, o qual foi treinado apenas com os filhos (neste caso 2.1 e 2.2), será ativado e irá atribuir o exemplo para o próximo nível, e assim por diante até encontrar os nós folha. Esse tipo de abordagem elimina inconsistências nas predições e respeita a hierarquia natural das classes.

Figura 5 – Exemplo de classificador em nós pais



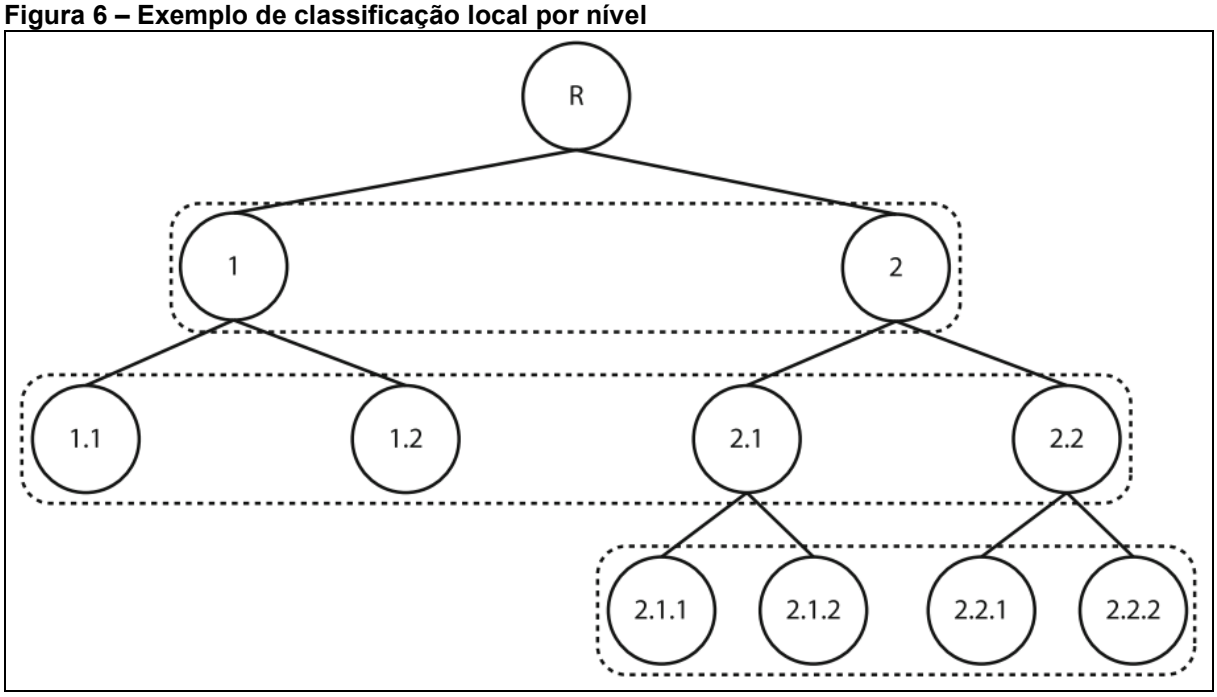
Fonte: Silla; Freitas (2011)

Esse tipo de classificação é melhor empregado em estruturas do tipo árvore, visto que em uma DAG um nó poderia ter múltiplos pais, os quais podem se encontrar em diferentes níveis da hierarquia, causando redundância na classificação (SILLA; FREITAS, 2011).

- Classificação Local Por Nível

A classificação local por nível consiste no treinamento de um classificador multi-classe para cada nível da hierarquia. A Figura 6 ilustra esse tipo de classificação. Considerando o exemplo dessa figura, três classificadores seriam treinados (um classificador para cada nível da hierarquia), onde cada classificador seria treinado para predizer uma ou mais classes (dependendo se o problema é multirrótulo ou não) do nível ao qual corresponde.

Figura 6 – Exemplo de classificação local por nível



Fonte: Silla; Freitas (2011)

Esse tipo de classificação pode também gerar o mesmo tipo de inconsistência da classificação local por nó. Visto que os classificadores de cada nível da Figura 6 podem gerar respostas do tipo: classe 2 no primeiro nível, classe 1.1 no segundo nível e classe 2.1.2 no terceiro nível. Assim, esse tipo de abordagem

deve ser complementado com algum método de pós-processamento que procure corrigir esse tipo de inconsistência (SILLA; FREITAS, 2011).

Silla e Freitas (2011) afirmam que esse tipo de classificador pode ser utilizado tanto para estruturas tipo árvore quanto para estruturas do tipo DAG. Porém, os tratamentos para tipos DAG seriam mais complexos, pois poderiam ter mais do que um único caminho entre dois nós, fazendo com que uma classe possa ser considerada pertencente a mais de um nível, causando redundância entre os classificadores.

É importante ressaltar que a classificação hierárquica local difere da classificação plana no quesito em que, no primeiro, a hierarquia é levada em conta trabalhando-se na perspectiva de informações locais (SILLA; FREITAS, 2011), já a classificação plana ignora-se a hierarquia das classes.

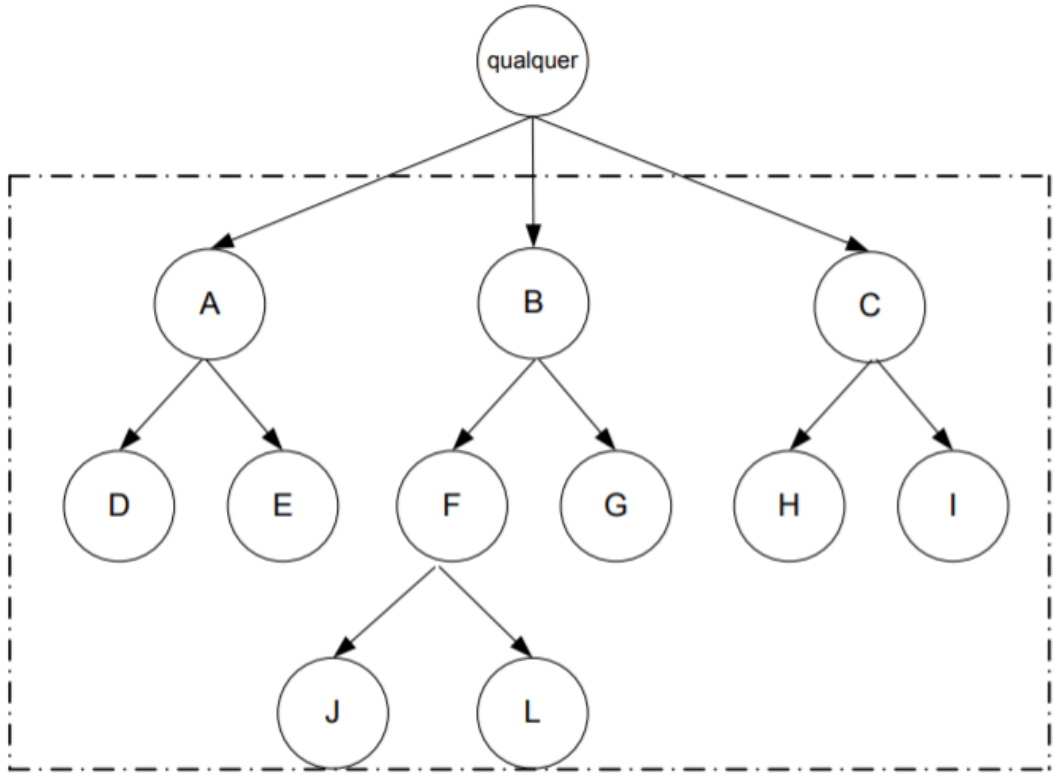
2.3.2 Classificação Global (ou *Big-Bang*)

Já a classificação hierárquica global, representado na Figura 7, consiste em construir um único classificador que leve em consideração a hierarquia de classes de todo o conjunto de treinamento (BORGES, 2012). Nesta abordagem é necessário apenas treinar um único classificador o qual irá realizar a predição em qualquer nível da hierarquia. O problema neste tipo de classificação é a complexidade em desenvolver um classificador, pois esse deve levar em consideração a hierarquia de classes como um todo durante a execução do algoritmo de classificação.

Silla e Freitas (2011) afirmam que o tamanho (espaço utilizado) de um modelo global é consideravelmente menor se comparado com o tamanho de todas as abordagens utilizadas nos modelos locais. Adicionalmente, pode-se levar em consideração todas as relações existentes entre as classes de forma natural e direta, podendo até ser explicitada (ilustrada) caso desejado.

Apesar desse modelo ter como uma das vantagens o aprendizado, como esse modelo considera todas as classes, a complexidade computacional desse algoritmo é maior durante a fase de treinamento.

Figura 7 – Exemplo de classificação global



Fonte: Borges (2012)

2.4 CONSIDERAÇÕES DO CAPÍTULO

Neste Capítulo foi apresentada uma breve descrição dos conceitos fundamentais sobre classificação e classificação hierárquica multirrótulo. As abordagens para esses problemas envolvem métodos de classificação local, que podem ser subdivididas em local por nó, local por nível e por nós pais, e a classificação global.

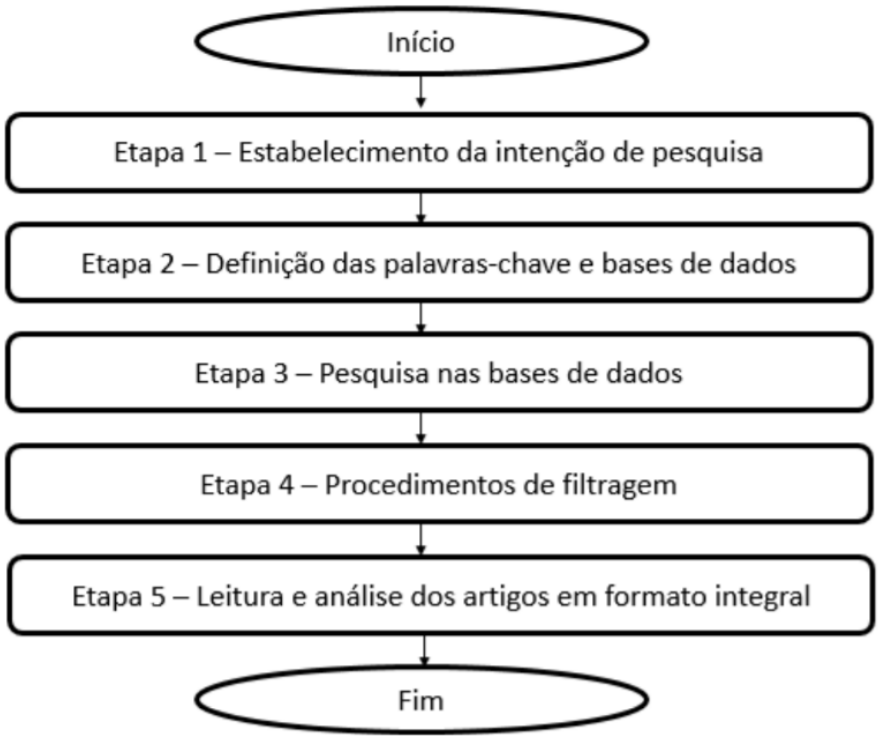
3 REVISÃO SISTEMÁTICA DA LITERATURA

Neste Capítulo é apresentado uma revisão sistemática da literatura abordando os trabalhos relacionados com o tema deste trabalho. Para isso, a Seção 3.1 descreve o método de revisão sistemática que será utilizada. A Seção 3.2 apresenta a aplicação da revisão sistemática em artigos científicos. A Seção 3.3 descreve o método base do algoritmo que irá ser aprimorado. Por fim, a Seção 3.4 contém as considerações do Capítulo.

3.1 DESCRIÇÃO DO MÉTODO

O método de revisão sistemática utilizado nesse trabalho é baseado no método proposto por Pagani, Kovaleski e Resende (2015) chamado de *Methodi Ordinatio*. Esse método de revisão é dividido em cinco etapas, como demonstrado na Figura 8, e busca refinar e diminuir a quantidade de trabalhos que devem ser lidos na íntegra.

Figura 8 – Etapas da Revisão Sistemática



Fonte: Adaptado de Almeida (2017)

Na primeira etapa deve ser definida a intenção de pesquisa, estabelecendo o tema a ser trabalhado e formando as questões de pesquisa às quais serão respondidas ao final da revisão sistemática. Também é fixada nessa etapa algumas características como tipo de trabalho, ano, local e idioma da publicação, que um trabalho deve conter para que seja considerado nesta revisão sistemática.

Após definido o tema da pesquisa, na segunda etapa, deve-se realizar uma combinação de palavras-chave com o intuito de pesquisar e validar os resultados em bases de dados definidos pelo pesquisador, capturando a existência de outros termos relacionados ao objetivo de pesquisa e verificando a necessidade de alteração ou recombinação das mesmas. Um minucioso trabalho nesta etapa, prestando atenção a detalhes e utilizando os filtros corretamente, acarreta a eliminação de documentos que não estão relacionados à intenção de pesquisa (PAGANI; KOVALESKI; RESENDE, 2015).

Na terceira etapa define-se um gerenciador de bibliografia para organizar resultados obtidos da busca realizada nas bases de dados, com as palavras-chave definidas na etapa anterior. Desses resultados, na quarta etapa, é feita uma filtragem descartando artigos duplicados, artigos que apresentem informações não relevantes em seu título, resumo ou palavras-chave. Outros filtros que não estão definidos no método, mas que o pesquisador considere significativos também devem ser aplicadas nesta etapa.

Para a última etapa, primeiramente é realizada a busca pelos artigos selecionados. Se o artigo for encontrado, então é feito a leitura e análise, caso não seja encontrado o mesmo é descartado. O objetivo da leitura é responder as perguntas definidas na etapa um.

3.2 APLICAÇÃO DO MÉTODO DE REVISÃO SISTEMÁTICA

Para a aplicação do método foram realizadas as cinco etapas conforme descritas na Seção 3.1. Definindo a intenção de pesquisa, as perguntas a serem respondidas e as palavras chaves. Logo após foram realizadas pesquisas nas bases de dados, os devidos filtros e a leitura dos artigos.

3.2.1 Etapa 1 – Estabelecimento da intenção de pesquisa

O objetivo do trabalho é desenvolver um método para reduzir a dimensão da hierarquia de classes na fase de treinamento do classificador MHC-CNN. Dessa forma, é necessário um estudo sobre algoritmos de classificação hierárquica, que utilizam a abordagem global e redes neurais. No Quadro 1 são expostas as perguntas definidas para essa intenção de pesquisa.

Quadro 1 – Definição de perguntas da revisão sistemática

ID	Perguntas
P1	Quais os algoritmos de classificação hierárquica global existentes?
P2	Qual foi a contribuição científica dos autores em cada trabalho escrito por eles? (criação de novo classificador ou aprimoramento de um método existente)
P3	Como eram organizados os dados (Árvore ou DAG)?
P4	Quais foram as áreas e as bases de dados em que a classificação hierárquica foi aplicada?
P5	Como foi feita avaliação do classificador hierárquico?

Fonte: Autoria própria

O período delineado para a revisão foi dos últimos 16 anos, considerando, assim, os trabalhos realizados entre os anos de 2003 até 2019. Serão considerados apenas artigos, desconsiderando nessa revisão os livros que forem encontrados durante a etapa de pesquisa nas bases de dados.

3.2.2 Etapa 2 – Definição das palavras-chaves e bases de dados

O Quadro 2 apresenta as palavras chaves selecionadas de acordo com o escopo dessa revisão sistemática.

Quadro 2 – Definição das palavras-chave

ID	Língua portuguesa	Língua inglesa
1	Classificação Hierárquica Multirrótulo	Multi-label Hierarchical Classification
2	Classificação Hierárquica Global	Global Hierarchical Classification
3	Grafo Acíclico Dirigido	Directed Acyclic Graph (DAG)

Fonte: Autoria própria

As palavras-chaves foram reorganizadas com operadores lógicos formando frases que serão utilizadas para realização das pesquisas conforme mostra o Quadro 3.

Quadro 3 – Palavras-chave reorganizadas com operadores lógicos

ID	*Strings* para pesquisa
S1	Multi-label hierarchical classification AND global approach
S2	Multi-label hierarchical classification AND (DAG OR directed acyclic graph)

Fonte: Autoria própria

A escolha das bases de dados foi realizada baseada em algumas sugestões do site da CAPES[1]. O Quadro 1 apresenta a relação das bases de dados em que foram realizadas as pesquisas e suas respectivas páginas da Internet.

Quadro 4 – Bases de dados

Base de dados	Sites
Science Direct	<http://www.sciencedirect.com>
Inderscience	<http://www.inderscience.com>
Springer	<http://link.springer.com>
Emerald Insight	<http://www.emeraldinsight.com>
ArXiv.org	<https://arxiv.org>
IEEEXplore	<http://ieeexplore.ieee.org>
SciELO.ORG	<http://www.scielo.org>
Scopus	<https://www.scopus.com >
Google Scholar	<https://scholar.google.com.br>

Fonte: Autoria própria

3.2.3 Etapa 3 – Pesquisas nas bases de dados

Para cada base de dados definidos na Etapa 2, foram inseridas as frases da intenção de pesquisa definidas na Etapa 2. O número dos resultados disponibilizados em cada base de dados está demonstrado conforme a Tabela 1.

[1] https://www.periodicos.capes.gov.br/index.php

Tabela 1 – Número de resultados por base

Base de dados	S1	S2
Science Direct	4	5
Inderscience	0	0
Springer	3	8
Emerald Insight	0	0
ArXiv.org	0	0
IEEEXplore	7	5
SciELO.ORG	0	2
Scopus	6	9
Google Scholar	32	63

Fonte: Autoria própria

Um total de 144 artigos foram encontrados os quais foram importados para o gerenciador de bibliografia Zotero[2].

3.2.4 Etapa 4 – Procedimentos de filtragem

Após a pesquisa nas bases, foram removidos os resultados duplicados. Além disso, fez-se a leitura do título, palavras-chave e resumo dos artigos resultando em 25 artigos.

3.2.5 Etapa 5 – Leitura e análise dos artigos em formato integral

Na sequencia foi feita a leitura integral dos 25 artigos resultantes da Etapa 4. Desses, 16 artigos foram selecionados como relevantes ao tema deste trabalho. Para simplificar a citação dos artigos encontrados, cada trabalho recebeu um número de identificação (ID) conforme o Quadro 5.

[2] https://www.zotero.org/

Quadro 5 – Artigos selecionados para a revisão sistemática

ID	Autor	Ano	Título
1	KIRITCHENKO, S.; MATWIN, S.; FAMILI, F.	2005	*Functional Annotation of Genes Using Hierarchical Text Categorization*
2	JUNG, J.; THON, M. R.	2008	*Gene function prediction using protein domain probability and hierarchical Gene Ontology information*
3	ALVES, R. T.; DELGADO, M. R.; FREITAS, A. A.	2008	*Multi-label Hierarchical Classification of Protein Functions with Artificial Immune Systems*
4	VENS, C.; STRUYF, J.; SCHIETGAT, L.; DŽEROSKI, S.; BLOCKEEL, H.	2008	*Decision Trees for Hierarchical Multi-label Classification*
5	SILLA JR, C. N.; FREITAS, A. A.	2009	*A Global-Model Naive Bayes Approach to the Hierarchical Prediction of Protein Functions*
6	BORGES, H. B.; NIEVOLA, J. C.	2012	*Multi-Label Hierarchical Classification using a Competitive Neural Network for protein function prediction*
7	ROMAO, L. M.; NIEVOLA, J. C.	2012	*Predicting GPCR and enzymes function with a global approach based on LCS*
8	CERRI, R.; BARROS, R. C.; DE CARVALHO, A. C. P. L. F.; FREITAS, A. A.	2013	*A grammatical evolution algorithm for generation of Hierarchical Multi-Label Classification rules*
9	FERRANDIN, M.; NIEVOLA, J. C.; ENEMBRECK, F.; SCALABRIN, E. E.; KREDENS, K. V.; ÁVILA, B. C.	2013	*Hierarchical classification using fca and the cosine similarity function*
10	SANTOS, A.; CANUTO, A.	2014	*Applying semi-supervised learning in hierarchical multi-label classification*
11	MENEZES, R. A.; NIEVOLA, J. C.	2014	*RCMDE-GMD: Predicting gene ontology terms using differential evolution*
12	PRABHA, G. M.; BALRAJ, E.	2014	*A HM Ant Miner Using Evolutionary Algorithm*
13	FERRANDIN, M.; ENEMBRECK, F.; NIEVOLA, J. C.; SCALABRIN, E. E.; ÁVILA, B. C.	2015	*A Centroid-based Approach for Hierarchical Classification.*
14	FERRANDIN, M.; ROMAO, L. M.	2016	*Hierarchical Classification with Jumping Emerging Patterns*
15	SOHRAB, M. G.; MIWA, M.; SASAKI, Y.	2016	*IN-DEDUCTIVE and DAG-Tree Approaches for Large-Scale Extreme Multi-label Hierarchical Text Classification*
16	CERRI, R.; BASGALUPP, M. P.; BARROS, R. C.; DE CARVALHO, A. C.	2019	*Inducing Hierarchical Multi-label Classification rules with Genetic Algorithms*

Fonte: Autoria própria

A partir dos 16 artigos é iniciada a análise para que possa ser possível responder as perguntas elencadas na Etapa 1.

P1: Quais os algoritmos de classificação hierárquica global existentes? (ver Quadro 6)

Quadro 6 – Respostas da pergunta P1

ID	Respostas da pergunta P1
1	*AdaBoost*
2	*Bayesian multi-label classifier*
3	*Multi-label Hierarchical Classification with an Artificial Immune System (MHC-AIS)*
4	*Árvore de Decisão baseada em PCT (Predictive Clustering Trees)*
5	*Global Naive Bayes*
6	*Multi-label Hierarchical Classification with Competitive Neural Network (MHC-CNN)*
7	*Hierarchical Learning Classifier System (HLCS)*
8	*Grammatical Evolution Algorithm for Generation of Hierarchical Multi-Label Classification Rules (GEHM)*
9	*Hierarchical Multi-label Classifier System - Formal Concept Analysis with Similarity Cosine (HMCS-FCA-SC)*
10	Novo: *Hierarchical Multi-label Classification using Random k-Labelsets (HMC-RAkEL)* Aprimorados: *Hierarchical Multi-label Classification using Semi-Supervised Binary Relevance (HMC-SSBR)* *Hierarchical Multi-label Classification using Semi-Supervised Label Powerset (HMC-SSLP)* *Hierarchical Multi-label Classification using Semi-Super-vised Random k-Labelsets (HMC-SSRAkEL)*
11	*Rule Construction Method Using Differential Evolution Global Multi-Directed Acyclic Graph (RCMDEGMD)*
12	*Hierarchical MultiLabel Classification Ant-Miner (hmAnt-Miner)*
13	*Hierarchical Centroid-Based Classifier System (HCCS) e Hierarchical Centroid-Based Classifier System intra-class (HCCSic)*
14	*Jumping emerging patterns*
15	*IN-DEDUCTIVE - large-scale hierarchical inductive learning and deductive classification*
16	*Hierarchical Multi-label Classification with a Genetic algorithm (HMC-GA)*

Fonte: Autoria própria

P2: Qual foi a contribuição científica dos autores em cada trabalho escrito por eles? (Desenvolvimento de novo classificador ou aprimoramento de um método existente) (ver Quadro 7)

Quadro 7 – Respostas da pergunta P2

ID	Respostas da pergunta P2
1	Novo classificador
2	Aprimoramento
3	Novo classificador
4	Aprimoramento
5	Novo classificador
6	Novo classificador
7	Novo classificador
8	Novo classificador
9	Aprimoramento
10	Novo classificador
11	Novo classificador e aprimoramentos
12	Aprimoramento
13	Novo classificador
14	Novo classificador
15	Novo classificador
16	Aprimoramento

Fonte: Autoria própria

P3: Como eram organizados os dados (Árvore ou DAG)? (ver Quadro 8)

Quadro 8 – Respostas da pergunta P3

ID	Respostas da pergunta P3
1	DAG
2	DAG
3	DAG
4	Árvore e DAG
5	Árvore
6	Árvore e DAG
7	Árvore e DAG
8	Árvore e DAG
9	Árvore e DAG
10	Árvore
11	DAG
12	Árvore
13	Árvore
14	Árvore
15	DAG
16	Árvore

Fonte: Autoria própria

P4: Quais foram as áreas e as bases de dados em que a classificação hierárquica foi aplicada? (ver Quadro 9)

Quadro 9 – Respostas da pergunta P4

ID	Respostas da pergunta P4
1	Função genética: processo biológico, função molecular e componente celular. Retirados da Saccharomyces Genome Database.
2	Função de proteína do reino Fungi. Retirados da database da Uniprot.
3	Função de proteína. Retirados da database da Uniprot.
4	Função de proteína disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Church, Derisi, Eisen, Expr, Gasch1, Gasch2, Pheno, Seq e Spo.
5	Função de proteína *G-Protein Coupled Receptor* (GPCR) e Enzimas. Disponibilizados no link https://sites.google.com/site/carlossillajr/resources. Bases de dados: Interpro, Pfam, Prints e Prosite.
6	Função de proteína disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Church, Derisi, Eisen e Spo.
7	Função de proteína *G-Protein Coupled Receptor* (GPCR) e Enzimas. Disponibilizados no link https://sites.google.com/site/carlossillajr/resources. Bases de dados: Interpro, Pfam, Prints e Prosite.
8	Função de proteína disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Derisi, Eisen, Gasch1 e Gasch2.
9	Função de proteína *G-Protein Coupled Receptor* (GPCR) e Enzimas. Disponibilizados no link https://sites.google.com/site/carlossillajr/resources. Bases de dados: Interpro, Pfam, Prints e Prosite.
10	Função de proteína do fungo Saccharomyces cerevisiae, disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Church, Derisi, Eisen e Pheno.
11	Função de proteína disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Church, Derisi, Eisen, Expr, Gasch1, Gasch2, Pheno, Seq e Spo.
12	Função de proteína disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Church, Derisi, Eisen, Expr, Gasch1, Gasch2, Pheno, Seq e Spo.
13	Função de proteína *G-Protein Coupled Receptor* (GPCR) e Enzimas. Disponibilizados no link https://sites.google.com/site/carlossillajr/resources. Bases de dados: Interpro, Pfam, Prints e Prosite.
14	Função de proteína *G-Protein Coupled Receptor* (GPCR) e Enzimas. Disponibilizados no link https://sites.google.com/site/carlossillajr/resources. Bases de dados: Interpro, Pfam, Prints e Prosite.
15	Textos utilizados em *PASCAL Challenge on Large-Scale Hierarchical Text Classification* (LSHTC). Bases de dados: *Wikipedia Medium Dataset* e *Wikipedia Large Dataset*.
16	Função de proteína disponível no repositório de *KU Leuven Declarative Languages and Artificial Intelligence Group*. Bases de dados: Cellcycle, Church, Derisi, Eisen, Expr, Gasch1, Gasch2, Pheno, Seq e Spo.

Fonte: Autoria própria

P5: Como foi feita avaliação do classificador hierárquico? (ver Quadro 10)

Quadro 10 – Respostas da pergunta P5

ID	Respostas da pergunta P5
1	Precisão e revocação.
2	Medida-F.
3	Correção preditiva com base na Medida-F, área da curva precisão e revocação e simplicidade do conjunto de regras descobertas.
4	Precisão e revocação e AUCPR.
5	Precisão hierárquica, revocação hierárquica e Medida-hF.
6	Distância e Medida-hF, acompanhados pelos testes estatísticos de Friedman e Nemenyi.
7	Precisão hierárquica, revocação hierárquica e Medida-hF.
8	AUPRC.
9	Método estatístico Rank-Sum de Wilcoxon e Curva-PR.
10	Testes estatísticos de Friedman e de Nemenyi.
11	Curva-PR, precisão hierárquica, revocação hierárquica e Medida-hF.
12	Não foi feita avaliação do classificador.
13	Precisão hierárquica, revocação hierárquica e Medida-hF, acompanhados pelo teste estatístico de Friedman.
14	Teste estatístico de Friedman usando Medida-hF.
15	Taxa de acerto, *example-based F1 measure, label-based Macro-average F1 measure, label-based Micro-average F1 measure* e *hierarchical F1 measure.*
16	Teste estatístico de Friedman.

Fonte: Autoria própria

As medidas de precisão hierárquica, revocação hierárquica e Medida-F hierárquica (Medida-hF), que são versões adaptadas, para cenários de classificação hierárquica, das já conhecidas medidas de precisão, revocação e Medida-F (SILLA; FREITAS, 2009).

Dos 16 artigos, 6 deles são classificadores para estruturas do tipo árvore, cinco para estruturas do tipo DAG e outros cinco tratam os dois tipos de estruturas, conforme mostra o Quadro 8.

Kiritchenko et. al. (2005) usa a hierarquia para aumentar o número de instâncias de treinamento em cada nó, aumentando a consistência entre os dados de treinamento com sua ontologia. Jung e Thon (2008) utiliza a hierarquia de classes como representação da Rede Neural Bayesiana.

Alves et al (2008) trabalha com *Artificial Immune Systems*, gerando resultados que são interpretáveis pelo usuário, no formato de regras SE-ENTÃO.

Romão e Nievola (2012) criam o HLCS que é baseado no LCS (*Learning Classifier System*), que é um método que gera resultados em um conjunto de regras SE-ENTÃO de mais fácil interpretação pelo usuário.

Vens et al (2008) trabalha com PCT (*Predictive Clustering Trees*) que visualiza uma árvore de decisão como uma hierarquia de *clusters*. Sohrab et al (2016) apresenta uma abordagem que gera uma hierarquia no formato árvore-DAG, que incorpora árvore e DAG juntos buscando reduzir o tempo de treinamento e teste e melhorar a performance da classificação.

Borges e Nievola (2012) propuseram uma rede neural competitiva, formado por uma camada de entrada e uma camada de saída. A distância entre os nós da hierarquia com a instância de treinamento é calculada. Os neurônios com os menores valores de distância eram considerados vencedores e a atualização de seus pesos sinápticos influenciam seus antecessores.

Cerri et al (2013) propôs um classificador baseado em regras geradas a partir do método de evolução gramatical, que combina as vantagens de programação genética baseado em gramática com algoritmo genético. Cerri et al (2019) utiliza algoritmo genético para induzir regras de classificação hierárquica.

Menezes e Nievola (2014) criam um método baseado em *Differential Evolution*, que consiste em uma população de N indivíduos, que evolvem até passar por G gerações.

Ferrandin et al (2013) cria um classificador que utiliza técnicas de FCA (*Formal Conceptual Analysis*) combinado com a função de *cosine similarity*. Santos e Canuto (2014) usa aprendizagem semi-supervisionada, adaptando-o para o conceito de classificação hierárquica multirrótulo.

Prabha e Balraj (2014) se baseiam no conceito de Colônia de Formigas, para encontrar boas soluções para otimização de problemas, adaptando para a classificação hierárquica.

Ferrandin et al (2015) usa a técnica baseada em centroides para realizar a classificação, seu método é uma adaptação de algoritmos baseados em centroides usados em categorização de texto.

Ferrandin et al (2016) cria um classificador baseado em JEP (*Jumping Emerging Patterns)*, que possui a vantagem de criar resultados facilmente compreendidos pelo usuário.

Dentre os artigos lidos, três deles utilizam rede neural para realizar a classificação hierárquica, são eles o *Bayesian multi-label classifier* (ALVES et. al, 2008), *Global Naive Bayes* (SILLA; FREITAS, 2009) e *Multi-label Hierarchical Classification with Competitive Neural Network* (BORGES, 2012).

3.3 CONSIDERAÇÕES DO CAPÍTULO

Neste Capítulo foi apresentada uma revisão sistemática envolvendo classificação hierárquica multirrótulo global. A classificação hierárquica global baseado em redes neurais ainda é pouco explorada, seu maior desafio é o custo computacional que redes neurais necessitam. O HNR apresentado neste trabalho pode ajudar a reduzir o custo computacional do classificador MHC-CNN.

4 DESENVOLVIMENTO DO MÉTODO HNR

Este Capítulo apresenta informações sobre a adaptação a ser feita no algoritmo MHC-CNN. A Seção 4.1 descreve o algoritmo MHC-CNN (BORGES, 2012). A Seção 4.2 aborda uma descrição das características gerais do SOM (KOHONEN, 1995) e como serviu de inspiração para a origem do método HNR (*Hierarchical Neighbour Reductor*). Por fim, a Seção 4.4 apresenta as considerações do Capítulo.

4.1 DESCRIÇÃO DO MHC-CNN

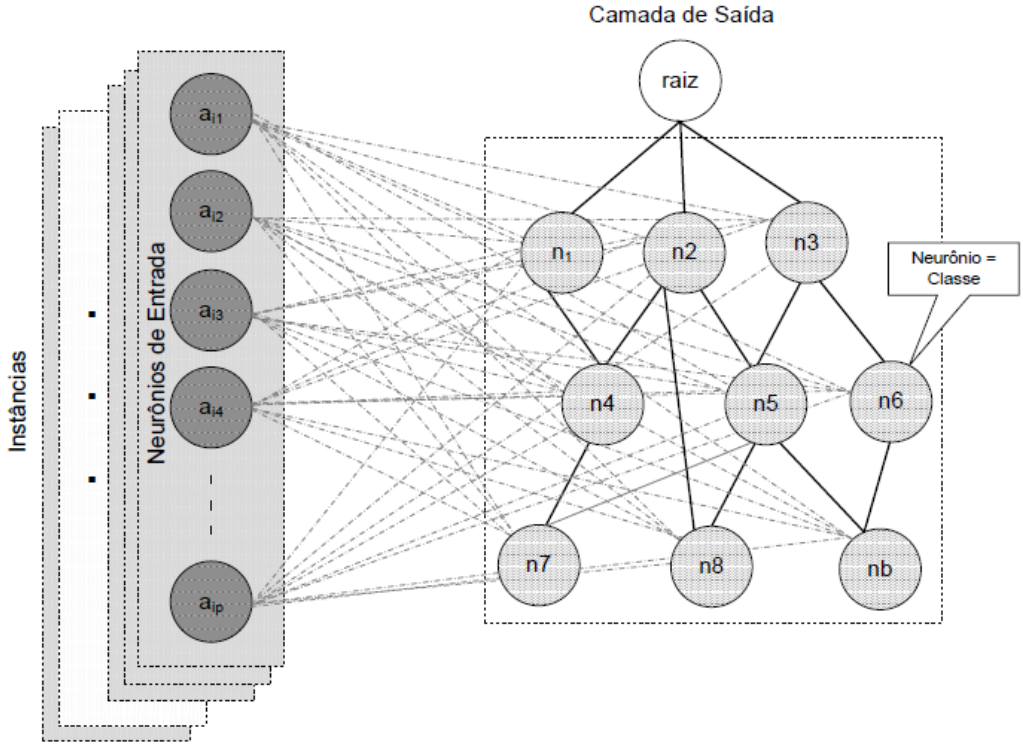
O algoritmo de classificação MHC-CNN é baseado em uma rede neural artificial competitiva, esse tipo de rede tem a habilidade de realizar mapeamentos que preservam a topologia entre entrada e saída de um neurônio. A rede neural utilizada pelo algoritmo é uma rede neural de duas camadas, a camada de entrada e a camada de saída.

A camada de entrada da rede é conectada a um vetor de entrada do conjunto de dados, já os neurônios da camada de saída são criados conforme o número de classes que cada base de dados possui. Cada neurônio possui um vetor que representa sua posição, chamado de pesos sinápticos. Pode-se observar as duas camadas conforme a Figura 9.

A rede é competitiva pois em seu processo de aprendizagem, os neurônios da camada de saída competem entre si para serem ativados de forma que apenas um neurônio de saída será o “vencedor”. Essa competição é realizada a partir de estímulos que são feitos pelos exemplos de entrada e é considerado o vencedor, ou vencedores em casos multirrótulo, os neurônios que mais se assemelham à instância de entrada selecionada. O nível de semelhança é medido por meio de medidas de distância.

Em redes neurais competitivas tradicionais, como a rede de Kohonen, os neurônios da camada de saída mapeiam a topologia da rede em grades, que podem ser retangulares, hexagonais, triangulares, entre outras. No algoritmo MHC-CNN a topologia da rede tem a estrutura de uma DAG, como mostra a camada de saída da Figura 9.

Figura 9 – Modelo do MHC-CNN



Fonte: Borges (2012)

O treinamento da rede necessita de um determinado número de épocas (ep), uma época representa um ciclo do algoritmo após tratar todas as instâncias, uma taxa de aprendizagem (inicial e final) que irá decrescer exponencialmente ao longo das épocas, essa taxa serve como um multiplicador de conversão dos pesos sinápticos dos neurônios, uma função de distância e uma função de atualização da rede.

A camada de entrada é criada por meio das instâncias da base de dados. Onde cada neurônio armazena os valores dos atributos de cada instância e também a(s) classe(s) que esses atributos representam.

A camada de saída consiste de neurônios que carregam pesos sinápticos, onde o número de pesos representa o número de atributos de entrada das instâncias.

A distância utilizada para realizar a verificação de semelhança entre as entradas e o vetor de pesos é dado pela Distância Euclidiana, conforme mostra a Equação 1.

$$d_{ik} = \sum_{i=1}^{l-1} \sqrt{(e_{ij} - n_{ijk})^2} \qquad (1)$$

Em que *e* representa cada elemento (atributo) da camada de entrada, e *n* representa cada neurônio da camada de saída. A distância é calculada para todas as *k* classes dos neurônios de saída.

Para problemas multirrótulo, deve-se identificar os neurônios de saída que apresentaram as menores distâncias em relação à instância. O número de neurônios selecionados se equipara ao número de classes da instância. Por exemplo, caso a instância pertença à três classes, deverá ser selecionado os três neurônios que obtiveram a menor distância.

Após identificar os neurônios "vencedores" é necessário atualizar seus pesos sinápticos e também de sua vizinhança, que são os ancestrais do neurônio. Assim, com o passar das épocas os neurônios vão se adaptando às instâncias e mapeando as classes. A atualização de pesos é feita pela Equação 2.

$$n_{ijk}(t+1) = \begin{cases} (n_{ijk} + (e_i(t) - n_{ij}(t)) * Ap(t) * Dist(t)), \ se\ a\ classe\ for\ correta \\ (n_{ijk} - (e_i(t) - n_{ij}(t)) * Ap(t) * Dist(t)), \ caso\ contrário \end{cases} \quad (2)$$

A classe é considerada como correta caso o neurônio vencedor pertença à uma das classes que representam a instância de entrada.

Ap(t) é a taxa de aprendizagem para o instante de tempo (t+1) a qual é obtida pela Equação 3.

$$Ap(t) = (\mu_i - \mu_f) * e^{-\frac{t_{atual}}{C}} \quad (3)$$

Em que μ_i é a taxa de aprendizagem inicial, μ_f é a taxa de aprendizagem final, t_{atual} é a iteração atual (a iteração varia de 1 até o número máximo de épocas definido) e C é uma constante definida para que a função exponencial decresça lentamente.

Dist (t) é a distância do neurônio vencedor até o seu ancestral que está sendo atualizado e é obtida pela Equação 4.

$$Dist(t) = 1/(k+1) \quad (4)$$

Em que *k* a distância (número de ligações em nós) entre o neurônio vencedor e o neurônio que está tendo seus pesos ajustados.

Esses passos ocorrem para todas as instâncias de entrada. Quando todas as instâncias são verificadas se conclui uma época, a próxima época é então iniciada e os passos se repetem até que o número total de épocas seja executada.

Após a última interação obtém-se os pesos sinápticos que serão utilizados na fase de testes do algoritmo. O Quadro mostra o algoritmo de treinamento do MHC-CNN de forma simplificada.

Quadro 11 – Algoritmo de treinamento do MHC-CNN

ENTRADA DO ALGORITMO
- Conjunto de dados de treinamento $BD_{trein} = [e_1\ e_2\ e_3 \ldots e_q]$ de dimensão m.
PASSO 1: INICIALIZAÇÃO
- Inicializar a taxa de aprendizagem inicial μ_i e a final μ_f. - Determinar o número de épocas ep. - Inicializar os pesos sinápticos da rede: $RN = [n_1\ n_2\ n_3\ n_b]$ onde b é o número de classes que existe na hierarquia de classes.
PASSO 2: CRITÉRIO DE PARADA
- Número de épocas.
PASSO 3: TREINAMENTO
- Selecionar uma instância e_i do conjunto de dados de entrada $BD_{trein} = [e_1\ e_2\ e_3 \ldots e_q]$. **FASE COMPETIVA:** - Calcular a distância entre a instância v_i com os neurônios da rede $RN = [n_1\ n_2\ n_3\ n_b]$. - Encontrar o(s) neurônio(s) n_j que apresentaram a(s) menor(es) distância(s), os quais serão considerados vencedores. **FASE COOPERAÇÃO:** - Encontrar os ancestrais do neurônio j. **FASE ADAPTAÇÃO:** - Atualizar os pesos sinápticos do neurônio(s) vencedor(es) e de seus ancestrais: $n_{ijk}(t+1) = \begin{cases} (n_{ijk} + (v_i(t) - n_{ij}(t)) * Ap(t) * Dist(t)), \text{ se a classe for correta} \\ (n_{ijk} - (v_i(t) - n_{ij}(t)) * Ap(t) * Dist(t)), \text{ em caso contrário} \end{cases}$ - $Dist(t) = 1$ se a classe verdadeira for igual a classe predita. - Calcular a taxa de aprendizagem $Ap(t) = (\mu_i - \mu_f) * e^{-\frac{t_{atual}}{C}}$ em que t_{atual} é a iteração atual e C é uma constante definida para que a função exponencial decresça lentamente.
PASSO 4: ATUALIZAÇÃO
- Atualizar a taxa de aprendizagem $Ap(t)$. - Inicia o PASSO 1.
SAÌDA DO ALGORITMO
- Conjunto de pesos considerado adequado.

Fonte: Borges (2012)

Na fase de teste (Quadro 12) o procedimento é muito parecido com a fase de treinamento, mas dessa vez os neurônios de entrada recebem os valores das instâncias de testes, os neurônios de saída recebem os pesos sinápticos obtidos ao final da fase de treinamento e esses valores não recebem atualizações.

Quadro 12 – Algoritmo de teste do MHC-CNN

ENTRADA DE DADOS
- Conjunto de dados de teste BD_{test}, representadas por $BD_{test} = [x_1\ x_2\ x_3\ \dots\ x_g]$ onde g é o total de instâncias de teste.
PASSO 1: CRITÉRIO DE PARADA
- Até que todas as instâncias tenham sido selecionadas e testadas.
PASSO 3: TESTE
- Selecionar uma instância x_i do conjunto de dados de entrada $BD_{test} = [x_1\ x_2\ x_3\ \dots\ x_g]$. - Calcular a distância entre a instância x_i com os neurônios da rede $RN = [n_1\ n_2\ n_3\ n_b]$. - Encontrar o(s) neurônio(s) n_j que apresentaram a(s) menor(es) distância(s), os quais serão considerados vencedores. - Encontrar os ancestrais do neurônio j. - Comparar predição. - Atribuir resultado para a matriz de confusão.
SAÍDA DO ALGORITMO
- Taxa de acerto obtido pelo algoritmo.

Fonte: Borges (2012)

4.2 DESCRIÇÃO DO MÉTODO HNR

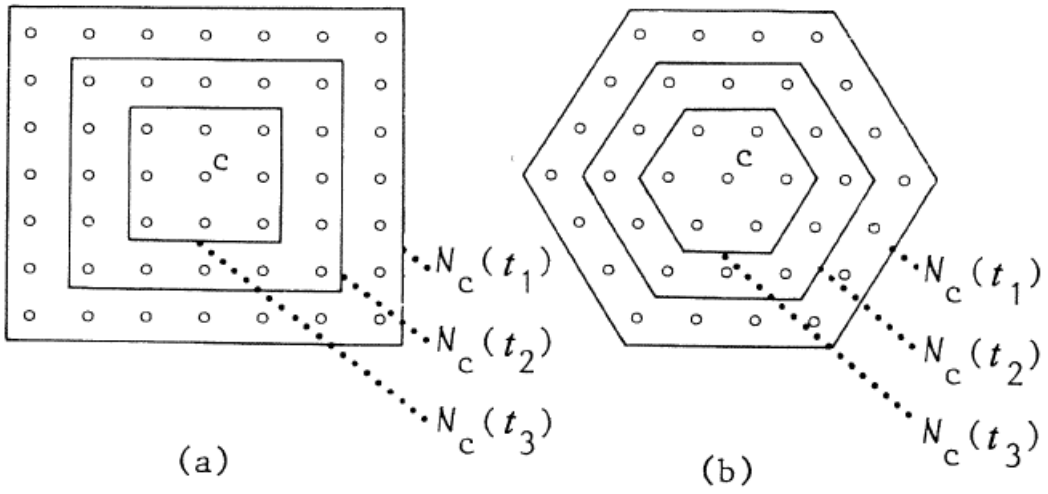
O método HNR (*Hierarchical Neighbour Reductor*) proposto neste trabalho é baseado na atualização dos pesos dos neurônios da rede neural artificial SOM (KOHONEN, 1995). O algoritmo SOM (KOHONEN, 1995) possui variáveis como a constante de aprendizagem, que decresce com o tempo, a função de vizinhança, que varia conforme a distância do nó “vencedor” com seus nós vizinhos. No método proposto visa explorar o conceito da função de vizinhança, que define uma distância máxima entre o nó “vencedor” e seu vizinho. Caso seu vizinho ultrapasse essa distância, o mesmo não será atualizado.

Pode-se observar na Figura 10 um exemplo onde *C* é o neurônio “vencedor”, e *Nc* representa o Raio de vizinhança do neurônio *C*. Esse raio pode

decrescer com o tempo, assim como a constante de aprendizagem. Na Figura 10, *Nc* decresce de acordo com o tempo t_1, t_2, e t_3.

O valor do raio deve ser inicializado juntamente com a constante de aprendizagem e deve ser atualizado a cada época do algoritmo. Durante a atualização dos pesos sinápticos dos vizinhos do neurônio “vencedor”, deve ser feita a verificação se a distância (em nós) entre o “vencedor” e o seu vizinho ultrapassou o valor do raio daquela época, caso negativo, atualiza-se o vizinho normalmente, em casos positivos, ignora-se a atualização desse vizinho e de todos os seus ancestrais.

Figura 10 – Exemplo de vizinhança topológica

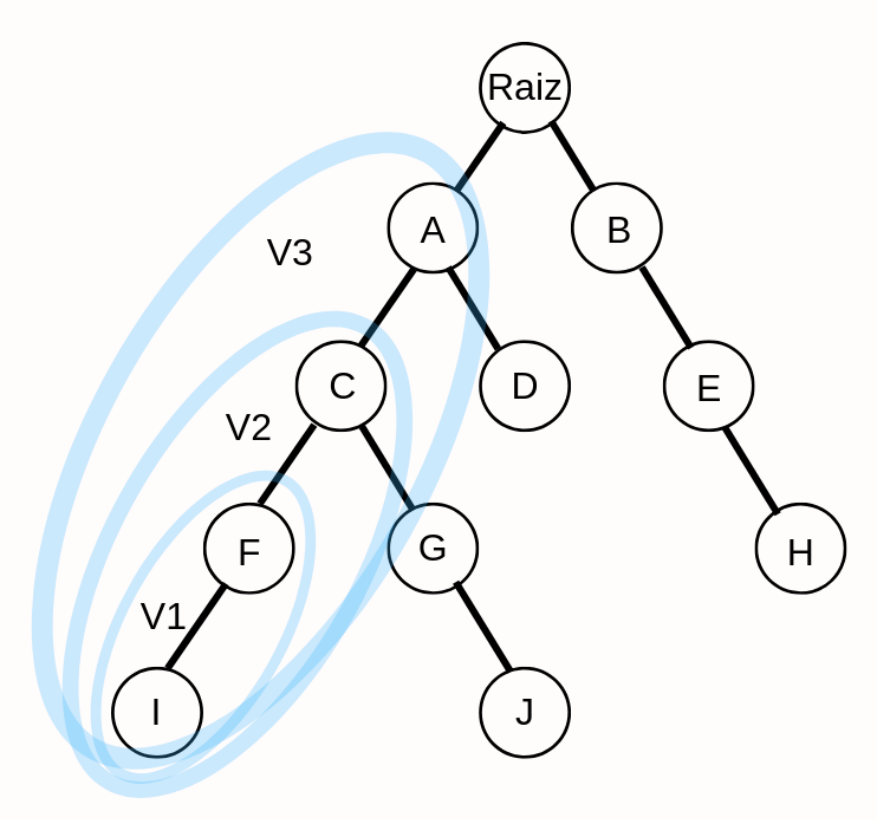


Fonte: Kohonen (1995)

O método HNR visa decrementar o número nós a serem atualizados, deduzindo o tempo gasto na fase de treinamento do modelo de classificação. Como o problema de classificação hierárquica trabalha com estruturas em árvore ou DAG, os vizinhos são os nós ancestrais ou descendentes conectados.

A Figura 11 apresenta um exemplo de vizinhança em uma hierarquia, considerando o nó *I*, *V1* seria o grupo de vizinhos que possui distância máxima um, *V2* o grupo de vizinhos que possui distância máxima 2, e *V3* o grupo de vizinhos que possui distância máxima 3.

Figura 11 – Vizinhança em uma árvore hierárquica



Fonte: Autoria própria

Para manter certa consistência dentro do algoritmo, a função utilizada para o decremento do raio ao longo das épocas é a mesma função utilizada na constante de aprendizagem. Assim, dado um tempo *t* (em épocas), a distância máxima de vizinhança, conforme mostra a Equação 5.

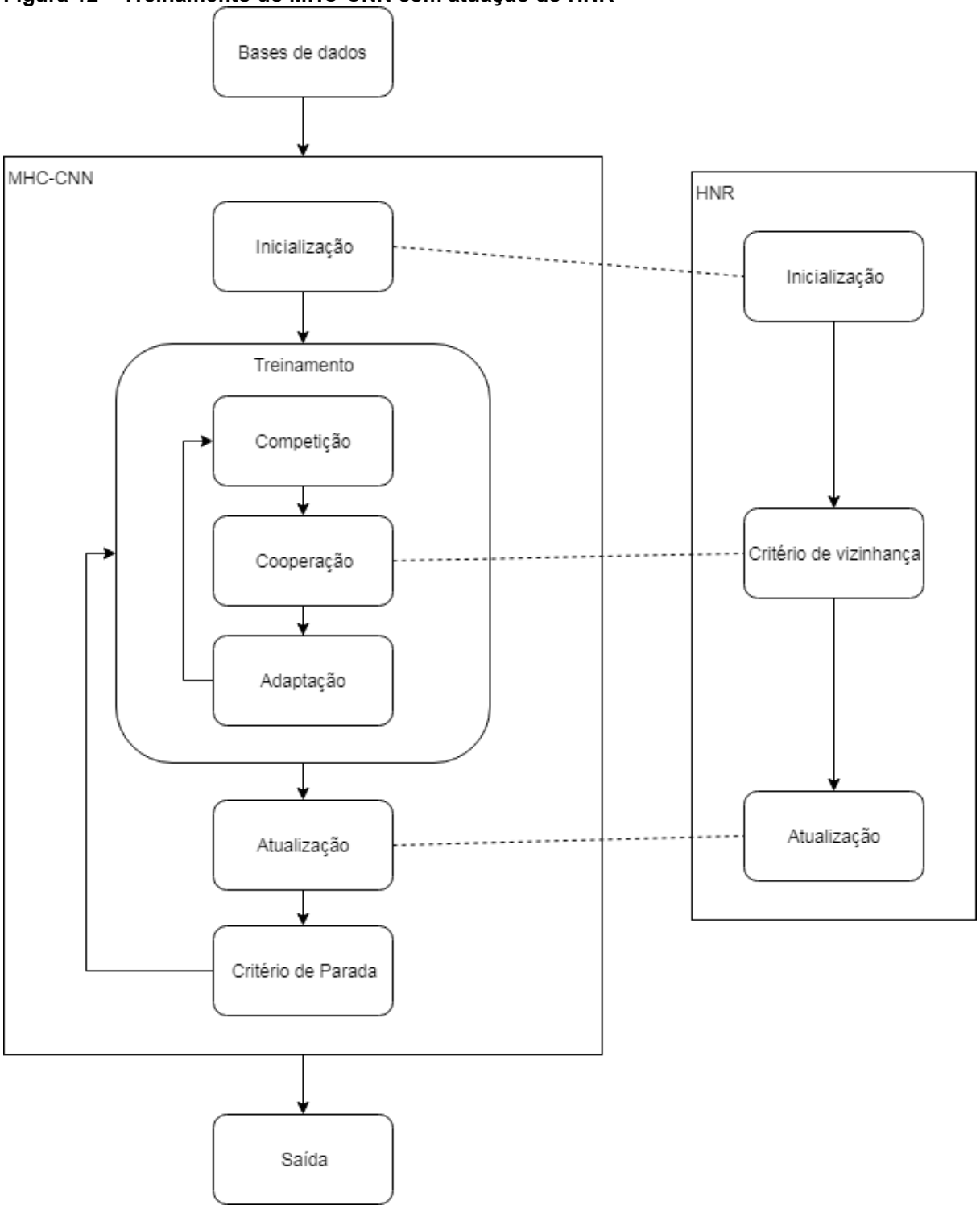
$$D_{max}(t) = (D_{mIni} - D_{mFim}) * e^{\frac{t_{atual}}{C}} \quad (5)$$

Em que, D_{mIni} é a distância máxima inicial, D_{mFim} é a distância máxima final e *C* é uma constante definida para que a função exponencial decresça.

4.3 APLICAÇÃO DO MÉTODO HNR NO ALGORITMO MHC-CNN

Para realizar a validação do método HNR feita a modificação do algoritmo MHC-CNN conforme ilustra A Figura 12

Figura 12 – Treinamento do MHC-CNN com atuação do HNR



Fonte: Autoria própria

O Quadro 13 apresenta o pseudocódigo do MHC-CNN adaptado com o método HNR (destaque em azul). A inicialização do MHC-CNN, que envolve a definição dos parâmetros de aprendizagem inicial, aprendizagem final, número de

épocas, e pesos sinápticos, passa a inicializar também a distância máxima inicial e final de vizinhança.

Quadro 13 – Pseudocódigo do MHC-CNN ∪ HNR

ENTRADA DO ALGORITMO
- Conjunto de dados de treinamento $Bd_{trein} = [e_1e_2e_3...e_q]$ de dimensão m.
PASSO 1: INICIALIZAÇÃO
- Inicializar a taxa de aprendizagem inicial e a final. - Inicializar a distância de vizinhança inicial e a final. - Determinar o número de épocas ep. - Inicializar os pesos sinápticos da rede: $RN = [n_1n_2n_3...n_b]$ onde b é o número de classes que existe na hierarquia de classes
PASSO 2: CRITÉRIO DE PARADA
- Número de épocas
PASSO 3: TREINAMENTO
- Selecionar uma instância e_i do conjunto de dados de entrada $Bd_{trein} = [e_1e_2e_3...e_q]$. **FASE COMPETITIVA:** - Calcular a distância entre a instância v_i com os neurônios da rede $RN = [n_1n_2n_3...n_b]$. - Encontrar o(s) neurônio(s) n_j que apresenta(ram) a(s) menor(es) distância(s), os quais serão considerados vencedores. **FASE COOPERAÇÃO:** - Encontrar os ancestrais do neurônio j. - Ignorar ancestrais do neurônio j que possuem distância maior do que o valor de vizinhança máxima. **FASE ADAPTAÇÃO:** - Atualizar os pesos sinápticos do(s) neurônio(s) vencedor(es) e de seus ancestrais.
PASSO 4: ATUALIZAÇÃO
- Atualizar a taxa de aprendizagem. - Atualizar o valor de vizinhança máxima. - Inicia o PASSO 2.
SAÍDA DO ALGORITMO
- Conjunto de pesos considerado adequado.

Fonte: Adaptado de Borges (2012)

No passo de treinamento, a competição encontra os neurônios vencedores. A cooperação gera os ancestrais destes neurônios vencedores. Os neurônios ancestrais devem fazer parte do grupo de vizinhos definidos por HNR para então realizar a atualização do neurônio vencedor e de seus ancestrais (fase de adaptação).

Por fim, é feita a atualização da taxa de aprendizagem, juntamente com a atualização da taxa de vizinhança de HNR, repetindo o ciclo até que o critério de parada seja satisfeito.

4.4 CONSIDERAÇÕES DO CAPÍTULO

Esse Capitulo apresentou o método proposto HNR e mostrou o seu funcionamento quando aplicado ao classificador MHC-CNN, visando reduzir a dimensionalidade da vizinhança, o SOM foi utilizado como inspiração do HNR.

5 EXPERIMENTOS E RESULTADOS

Este Capítulo apresenta informações dos experimentos e resultados realizados. A Seção 5.1 aborda as bases de dados utilizados para os experimentos. Na Seção 5.2 é exposta a metodologia utilizada para a realização dos experimentos. A Seção 5.3 apresenta os resultados obtidos pelo algoritmo. Por fim a Seção 5.4 compara os resultados obtidos com os resultados do algoritmo MHC-CNN.

5.1 BASE DE DADOS

Para a realização dos experimentos foram utilizadas dez bases de dados GO (*Gene Ontology*), representando genes e atributos de funções proteicas. Essas bases são as mesmas utilizadas por Vens et. al (2008), Alves et. al (2010) e Borges (2012). A Tabela 2 apresenta algumas características dessas bases.

Tabela 2 – Características das Bases de Dados GO

Base de Dados	Quant. Amostras	Quant. Atributos	Quant. Classes	Quant. Max. Níveis	Quant. Min/Max Classes por Amostras	Quant. Min/Max Amostras por Classe
Cellcycle (SPELLMAN et al., 1998)	3751	77	4125	13	3/28	0/785
Church (ROTH et al., 1998)	3749	27	4125	13	3/28	0/786
Derisi (DERISI et al, 1997)	3719	63	4119	13	3/28	0/781
Eisen (EISEN et al., 1998)	2418	79	3573	13	3/28	0/492
Expr (CLARE, 2003)	3773	551	4131	13	3/28	0/789
Gasch1 (GASCH et al., 2000)	3758	173	4125	13	3/28	0/786
Gasch2 (GASCH et al., 2001)	3773	52	4131	13	3/28	0/789
Pheno (CLARE, 2003)	1586	69	3127	13	3/21	0/388
Seq (CLARE, 2003)	3900	478	4133	13	3/28	0/791
Spo (CHU et al., 1998)	3697	80	4119	13	3/28	0/775

Fonte: Borges (2012)

5.2 METODOLOGIA PARA REALIZAÇÃO DOS EXPERIMENTOS

A metodologia utilizada para a realização dos experimentos é formada por duas etapas: realização dos experimentos e a avaliação dos resultados obtidos. Na sequência é descrita detalhadamente cada umas dessas etapas.

5.2.1 Realização dos experimentos

Para a execução dos experimentos com o método HNR, foi necessário aplicá-lo ao método proposto HNR no classificador MHC-CNN. Os parâmetros adotados serão os mesmos adotados por Borges (2012). Taxa de aprendizagem inicial igual 0,1, a taxa de aprendizagem final igual a 0,01 e a constante C igual a 1000.

Para a redução dos neurônios vizinhos atualizados pelo algoritmo MHC-CNN∪HNR foi utilizada a fórmula presente na Equação 5, com vizinho inicial de 17 e final de 0 e a constante C igual a 1000. O valor 17 foi selecionado após alguns testes iniciais, o mesmo apresentou bons resultados pois garante que as épocas iniciais atualizem todos os ancestrais, recomenda-se o uso da maior distância da hierarquia acrescidos de 30% desse valor.

Com estes parâmetros, o algoritmo MHC-CNN∪HNR foi treinado, e quando atingido as épocas 50, 500 e 1000, foram realizados os testes para coletar as medidas de Distância e a Medida-Fh.

5.2.2 Avaliação dos resultados

Os resultados foram avaliados utilizando o teste de hipótese de Friedman (FRIEDMAN, 1940). Esse teste é sugerido por Demsar (2006) por ser um teste não paramétrico que compara o desempenho de vários algoritmos em diferentes bases de dados. Além disso será utilizado o uso da estatística FF distribuída de acordo com a distribuição F de Snedecor, conforme sugerido por (IMAN; DAVENPORT, 1980).

A comparação é feita com os algoritmos de Borges (2012), treinados em 50, 500 e 1000 épocas.

5.3 RESULTADOS

A Tabela 3 apresenta os resultados obtidos do algoritmo pela medida de Distância e Medida-Fh, cada uma com 50, 500 e 1000 épocas. Nessas medidas, quanto maior, melhor.

Tabela 3 – Resultados do MHC-CNN∪HNR na Base de Dados

	MHC-CNN∪HNR					
	Distância			Medida-Fh		
	50	500	1000	50	500	1000
CECYCLE	25,9%	27,5%	28,1%	18,4%	20,9%	21,3%
CHURCH	21,4%	24,3%	19,5%	17,8%	20,7%	16,8%
DERISI	23,8%	25,7%	27,0%	19,9%	22,1%	23,7%
EISEN	27,6%	28,2%	30,8%	20,9%	22,1%	24,9%
EXPR	29,6%	29,3%	32,5%	26,3%	26,0%	30,5%
GASCH1	29,8%	30,3%	30,2%	25,7%	27,9%	23,7%
GASCH2	26,6%	26,3%	27,2%	20,3%	22,0%	20,2%
PHENO	27,8%	22,7%	23,5%	26,6%	20,0%	22,2%
SEQ	27,7%	28,4%	27,3%	24,4%	24,9%	26,5%
SPO	24,5%	25,9%	27,5%	21,4%	23,1%	25,0%

Fonte: Autoria própria

5.4 COMPARAÇÃO DOS RESULTADOS ENTRE MHC-CNN∪HNR COM MHC-CNN

Nesta seção são apresentados os comparativos feitos entre os algoritmos MHC-CNN∪HNR e MHC-CNN de Borges (2012) utilizando o teste de Friedman para realizar a avaliação estatística entre os resultados de medida de distância e da medida-Fh.

Tabela 4 – Comparativo entre o MHC-CNN∪HNR e o MHC-CNN usando a medida de distância

	MHC-CNN∪HNR			MHC-CNN		
	50	500	1000	50	500	1000
CECYCLE	25,9%	27,5%	28,1%	26,5%	27,3%	27,5%
CHURCH	21,4%	24,3%	19,5%	28,2%	26,8%	23,7%
DERISI	23,8%	25,7%	27,0%	24,3%	25,1%	24,5%
EISEN	27,6%	28,2%	30,8%	27,8%	28,1%	26,8%
EXPR	29,6%	29,3%	32,5%	31,9%	31,9%	31,8%
GASCH1	29,8%	30,3%	30,2%	30,0%	29,0%	29,6%
GASCH2	26,6%	26,3%	27,2%	26,6%	25,6%	26,1%
PHENO	27,8%	22,7%	23,5%	21,5%	23,7%	24,9%
SEQ	27,7%	28,4%	27,3%	21,5%	30,8%	31,2%
SPO	24,5%	25,9%	27,5%	24,6%	24,8%	24,4%

Fonte: Autoria própria

A Tabela 4 mostra os resultados dos dois algoritmos utilizando a medida de distância. Realizando o cálculo dos postos médio e aplicado esses dados na equação de Friedman (1940), obtém-se o qui-quadrado de 7,86. Em seguida,

usando a estatística (Ff) para a correção do qui-quadrado com n = 6 e k = 10, tem-se (Ff) = 0,8526.

De acordo com o valor tabelado de F de Snedecor para os graus de liberdade k-1 e (k-1)*(n-1) tem-se F(9;45) = 2,09. Assim, como (Ff) < F(9;45) a hipótese nula não pode ser rejeitada, indicando que não há diferença estatística entre os resultados dos algoritmos comparados.

Utilizando a medida-Fh para comparação entre os algoritmos, como mostrado na Tabela 5, tem-se qui-quadrado como 25,18 e (Ff) = 4,37. Com os mesmos graus de liberdade tem-se F(9;45) = 2,09. Como Ff > F(9;45) determina-se que a hipótese nula pode ser rejeitada, indicando que há diferença estatística entre os resultados dos algoritmos.

Com a rejeição da hipótese nula, deve-se identificar quais algoritmos possuem diferenças significativas, pela distância crítica (CD). Considerando o nível de significância de 95% tem-se o valor de CD = 20,55. Comparando a diferença entre a soma dos postos médios entre os algoritmos, é possível afirmar que o algoritmo é estatisticamente superior a outro se essa diferença for maior que a distância crítica.

Na Tabela 5 é possível afirmar que os resultados do algoritmo MHC-CNN∪HNR, nos casos de 500 e 1000 épocas, é estatisticamente superior aos resultados do algoritmo MHC-CNN nos três casos selecionados: 50, 500 e 1000 épocas.

Tabela 5 – Comparativo entre o MHC-CNN∪HNR e o MHC-CNN usando a medida-Fh

	MHC-CNN∪HNR			**MHC-CNN**		
	50	**500**	**1000**	**50**	**500**	**1000**
CECYCLE	18,4%	20,9%	21,3%	19,0%	18,8%	19,2%
CHURCH	17,8%	20,7%	16,8%	21,6%	18,2%	16,6%
DERISI	19,9%	22,1%	23,7%	16,1%	17,0%	17,5%
EISEN	20,9%	22,1%	24,9%	21,0%	21,0%	20,8%
EXPR	26,3%	26,0%	30,5%	24,8%	25,0%	25,2%
GASCH1	25,7%	27,9%	23,7%	22,4%	21,9%	22,1%
GASCH2	20,3%	22,0%	20,2%	18,8%	18,3%	18,6%
PHENO	26,6%	20,0%	22,2%	13,9%	15,7%	15,2%
SEQ	24,4%	24,9%	26,5%	13,9%	24,3%	24,6%
SPO	21,4%	23,1%	25,0%	17,2%	17,6%	17,6%

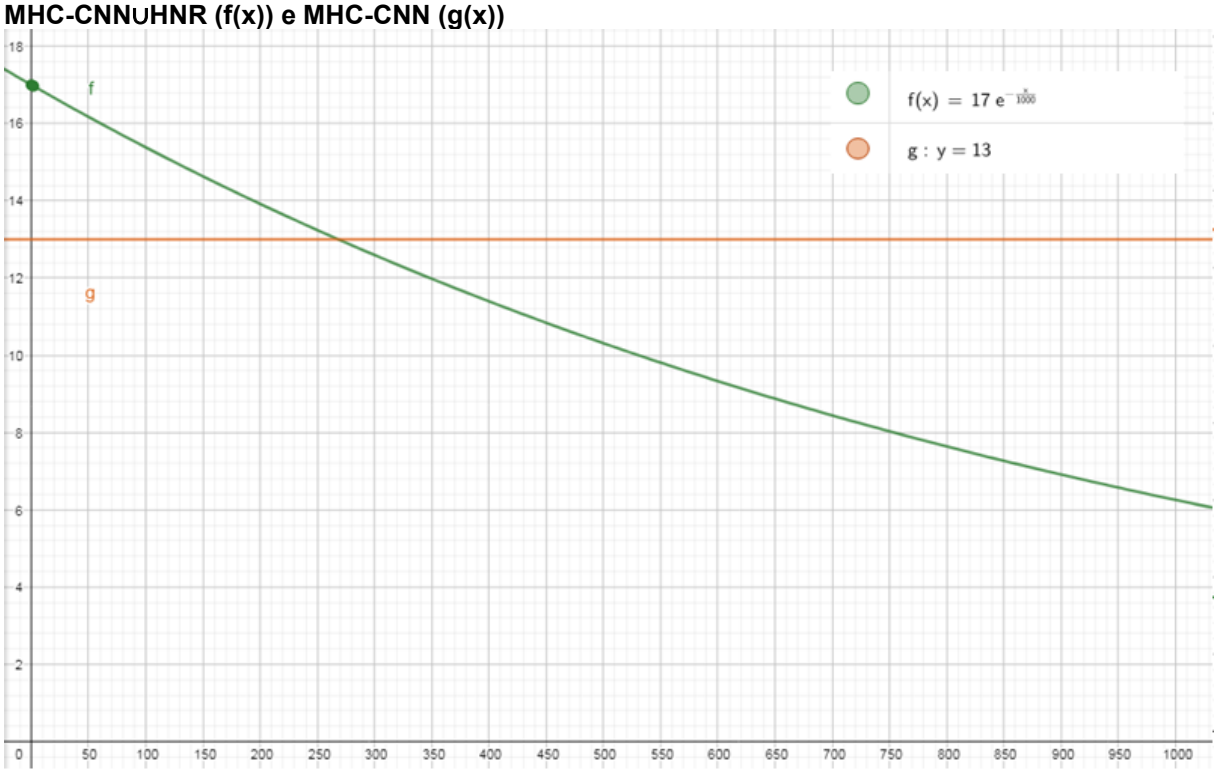
Fonte: Autoria própria

Como a ideia do método HNR é a redução dos número de neurônios ajustados durante a fase de treinamento, o Gráfico 1 mostra visualmente o quanto essa redução representa em número de neurônios acessados em cada época. O eixo y representa a distância máxima de vizinhança que é considerada, e o eixo x representa o número de épocas.

A função g(x) representa o treinamento do MHC-CNN e a função f(x) representa o treinamento do MHC-CNN∪HNR. Com isso é possível verificar visualmente a quantidade de neurônios vizinhos considerados em cada época. Pro exemplo, na época 500, o MHC-CNN considera como vizinho máximo a distância de 13 níveis, enquanto o MHC-CNN∪HNR considera 10 níveis.

Pode-se encontrar a diferença entre a redução do número de acessos aos neurônios por meio da diferença de área abaixo da curva dos dois algoritmos.

Gráfico 1 – Comparação de número de acessos aos neurônios na fase de treinamento entre MHC-CNN∪HNR (f(x)) e MHC-CNN (g(x))



Fonte: Autoria própria

Para encontrar a área abaixo da curva, calcula-se a integral da função *f* entre os valores de x entre 268 e 1000, conforme mostra a Equação 6.

$$\int_{268}^{1000} 17e - \frac{z}{1000} dx \tag{6}$$

Sendo a área total de g(x) = 13000 e a área total aproximada de f(x) = 10233. Pode-se afirmar que para o algoritmo quando executado com 1000 épocas há uma redução de 21,28% no número de acessos de atualizações dos pesos sinápticos dos neurônios durante o processo de treinamento.

Não há diferença no tempo de treinamento dos algoritmos de 50 épocas, e há redução de 4,97% entre os algoritmos de 500 épocas.

Além disso é também possível reduzir a constante para que a função exponencial decresça mais rapidamente, digamos que seja utilizado o valor 650 como constante da função de atualização do MHC-CNN∪HNR. O Gráfico 2 apresenta a diferença entre o MHC-CNN∪HNR e o MHC-CNN. Nesse caso, a redução do número de acesso aos neurônios vizinhos em 1000 épocas é de 35,81%.

Gráfico 2 – Comparação de número de acessos aos neurônios na fase de treinamento entre MHC-CNN∪HNR (f(x)) e MHC-CNN (g(x)) com constante 650

Fonte: Autoria própria

5.5 CONSIDERAÇÕES DO CAPÍTULO

Neste Capítulo foi apresentado os resultados obtidos do classificador MHC-CNN∪HNR, e sua comparação com o algoritmo MHC-CNN. Além disso cálculos matemáticos foram feitos para apresentar redução no tempo de treinamento do algoritmo.

6 CONCLUSÃO

Neste trabalho foi apresentado um novo método chamado HNR para o problema de classificação hierárquica para dados organizados em árvores ou DAG. O algoritmo proposto é baseado no SOM (KOHONEN, 1995) e aplicado no algoritmo MHC-CNN (BORGES, 2012).

A classificação foi realizada considerando a abordagem global da hierarquia, podendo realizar uma classificação multirrótulo em dados organizados em árvores ou DAG. A maior desvantagem desse tipo de classificação é o tempo de processamento devido ao seu comportamento exponencial, mas HNR tenta reduzir esse problema no algoritmo MHC-CNN.

O MHC-CNN∪HNR foi aplicado em 10 bases de dados de função de proteína. Os seus resultados foram coletados e comparados com os resultados do algoritmo MHC-CNN utilizando teste estatístico de Friedman. Os resultados coletados são promissores mostrando que a atualização de nós ancestrais pode ser limitada até certo nível de vizinhança.

Com isso conclui-se que o método proposto neste trabalho possibilita às redes neurais redução no tempo de treinamento do algoritmo, sem perder o desempenho, sendo um fator importante para futuras implementações de classificadores hierárquicos multirrótulo globais baseados em redes neurais artificiais.

6.1 TRABALHOS FUTUROS

Novos testes com alterações nos parâmetros iniciais, como os valores da distância da vizinhança e taxa de aprendizagem, poderão ser realizados em trabalhos futuros. Refinando esses valores podem gerar melhoras no custo computacional e nos resultados.

O método pode ser avaliado em novas bases de dados de classificação hierárquica, como as bases da Uniprot, utilizadas em Jung e Thon (2008) e Alves et al (2008), ou Interpro, Pfam, Prints e Prosite, utilizadas em Silla e Freitas (2009), Romão e Nievola (2012), Ferrandin et al (2013), Ferrandin et al (2015) e Ferrandin e Romão (2016).

Além disso, estudos poderão ser feitos para adaptar o método para outros algoritmos baseados em redes neurais como o *Bayesian multi-label classifier* (ALVES et. al, 2008) e *Global Naive Bayes* (SILLA; FREITAS, 2009).

REFERÊNCIAS

ALVES, R. T.; DELGADO, M. R.; FREITAS, A. A. Multi-label Hierarchical Classification of Protein Functions with Artificial Immune Systems. In: A. L. C. Bazzan; M. Craven; N. F. Martins (Orgs.); **Advances in Bioinformatics and Computational Biology**. v. 5167, p.1–12, 2008. Berlin, Heidelberg: Springer Berlin Heidelberg.

BENNETT, P. N.; NGUYEN, N. Refined experts: improving classification in large taxonomies. Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval. **Anais...** p.11–18, 2009. ACM.

BORGES, H. B. **Classificador hierárquico multirrótulo usando uma rede neural competitiva.**, 2012. PhD Thesis, Pontifical Catholic University of Paraná, Curitiba, Brazil.

BORGES, H. B.; NIEVOLA, J. C. Multi-Label Hierarchical Classification using a Competitive Neural Network for protein function prediction. The 2012 International Joint Conference on Neural Networks (IJCNN). **Anais...** p.1–8, 2012.

CERRI, R.; BASGALUPP, M. P.; BARROS, R. C.; DE CARVALHO, A. C. Inducing Hierarchical Multi-label Classification rules with Genetic Algorithms. **Applied Soft Computing**, v. 77, p. 584–604, 2019.

CERRI, R.; BARROS, R. C.; DE CARVALHO, A. C. P. L. F.; FREITAS, A. A. A grammatical evolution algorithm for generation of Hierarchical Multi-Label Classification rules. 2013 IEEE Congress on Evolutionary Computation. **Anais...** p.454–461, 2013.

DEKEL, O.; KESHET, J.; SINGER, Y. An online algorithm for hierarchical phoneme classification. International Workshop on Machine Learning for Multimodal Interaction. **Anais...** p.146–158, 2004. Springer.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. **Journal of Machine learning research**, v. 7, n. Jan, p. 1–30, 2006.

FACELI, K.; LORENA, A. C.; GAMA, J.; CARVALHO, A. C. P. DE L. **Inteligência Artificial: Uma abordagem de aprendizado de máquina**. Rio de Janeiro: LTC, 2011.

FERRANDIN, M.; ENEMBRECK, F.; NIEVOLA, J. C.; SCALABRIN, E. E.; ÁVILA, B. C. A Centroid-based Approach for Hierarchical Classification. ICEIS (1). **Anais...** p.25–33, 2015.

FERRANDIN, M.; ROMAO, L. M. Hierarchical Classification with Jumping Emerging Patterns. **IEEE Latin America Transactions**, v. 14, n. 9, p. 4143–4149, 2016.

FERRANDIN, M.; NIEVOLA, J. C.; ENEMBRECK, F.; SCALABRIN, E. E.; KREDENS, K. V.; ÁVILA, B. C. Hierarchical classification using fca and the cosine similarity function. Proceedings on the International Conference on Artificial Intelligence. **Anais...** p.1, 2013.

FREITAS, A.; CARVALHO, A. A tutorial on hierarchical classification with applications in bioinformatics. **Research and trends in data mining technologies and applications**. p.175–208, 2007. IGI Global.

JUNG, J.; THON, M. R. Gene function prediction using protein domain probability and hierarchical Gene Ontology information. 2008 19th International Conference on Pattern Recognition. **Anais...** p.1–4, 2008.

KIRITCHENKO, S.; MATWIN, S.; FAMILI, F. Functional Annotation of Genes Using Hierarchical Text Categorization. **Proceedings of BioLink SIG, ISMB**, 2005.

KOHONEN, T. **Self-Organizing Maps**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995.

LESK, A. M. **Introdução à bioinformática**. 2a ed. Porto Alegre: Artmed, 2008.

LI, T.; OGIHARA, M. Music genre classification with taxonomy. Proceedings (ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005. **Anais...** v. 5, p.v–197, 2005. IEEE.

MENEZES, R. A.; NIEVOLA, J. C. RCMDE-GMD: Predicting gene ontology terms using differential evolution. 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD). **Anais...** p.518–524, 2014.

MONARD, M. C.; BARANAUSKAS, J. A. Conceitos Sobre Aprendizado de Máquina. **Sistemas Inteligentes Fundamentos e Aplicações**. 1º ed, p.89–114, 2003. Barueri-SP: Manole Ltda.

PAGANI, R. N.; KOVALESKI, J. L.; RESENDE, L. M. Methodi Ordinatio: a proposed methodology to select and rank relevant scientific papers encompassing the impact factor, number of citation, and year of publication. **Scientometrics**, v. 105, n. 3, p. 2109–2135, 2015.

PRABHA, G. M.; BALRAJ, E. A HM Ant Miner Using Evolutionary Algorithm. **Int. J. Innov. Res. Sci. Eng. Technol**, v. 3, n. 3, p. 1687–1692, 2014.

ROMAO, L. M.; NIEVOLA, J. C. Predicting GPCR and enzymes function with a global approach based on LCS. 2012 IEEE 12th International Conference on Bioinformatics & Bioengineering (BIBE). **Anais...** p.151–156, 2012.

SANTOS, A.; CANUTO, A. Applying semi-supervised learning in hierarchical multi-label classification. **Expert Systems with Applications**, v. 41, n. 14, p. 6075–6085, 2014.

SILLA JR, 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. **Anais...** p.992–997, 2009.

SILLA, C. N.; FREITAS, A. A. A survey of hierarchical classification across different application domains. **Data Mining and Knowledge Discovery**, v. 22, n. 1–2, p. 31–72, 2011.

SOHRAB, M. G.; MIWA, M.; SASAKI, Y. IN-DEDUCTIVE and DAG-Tree Approaches for Large-Scale Extreme Multi-label Hierarchical Text Classification. **Polibits**, n. 54, p. 61–70, 2016.

VALENTINI, G.; RE, M. Weighted True Path Rule: a multilabel hierarchical algorithm for gene function prediction. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases: 1st International Workshop on learning from Multi-Label Data. **Anais...** 2009.

VENS, C.; STRUYF, J.; SCHIETGAT, L.; DŽEROSKI, S.; BLOCKEEL, H. Decision trees for hierarchical multi-label classification. **Machine Learning**, v. 73, n. 2, p. 185, 2008.

XIAO, Z.; DELLANDREA, E.; DOU, W.; CHEN, L. Hierarchical classification of emotional speech. **IEEE Transactions on Multimedia**, v. 37, 2007.

ANEXO A - Avaliação dos Classificadores Hierárquicos (Medida Distância e Medida-Fh)

AVALIAÇÃO DOS CLASSIFICADORES HIERÁRQUICOS

Para avaliar problemas de classificação hierárquica, os métodos convencionais de avaliação de desempenho devem ser adaptadas, pois os mesmos não são capazes de identificar um relacionamento hierárquico entre as classes.

Além disso, não há um consenso geral sobre os métodos que devem ser adotados para esse tipo de avaliação (SILLA; FREITAS, 2010). Neste trabalho serão utilizadas as medidas baseadas em distância e as medidas baseadas na relação de ancestralidade e descendência.

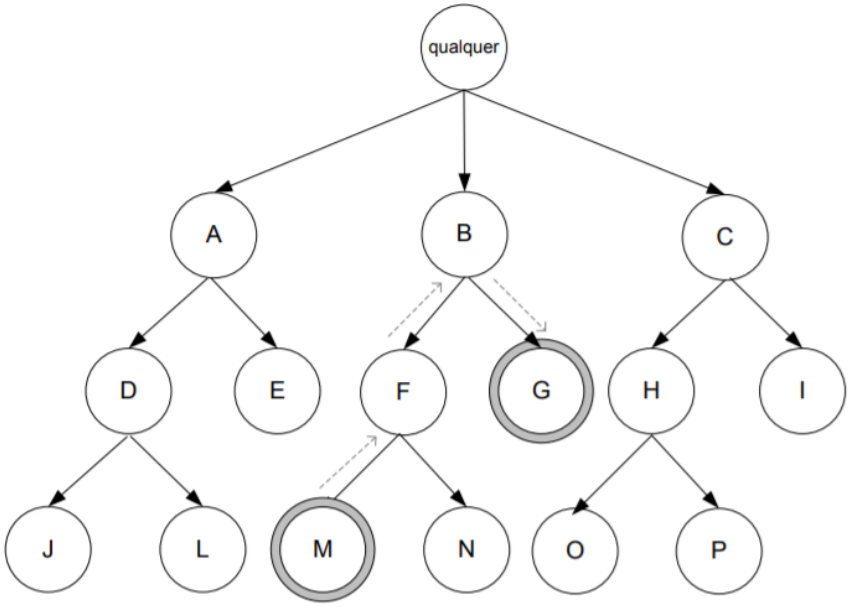
Medidas Baseada em Distância

Essa medida de desempenho consiste em atribuir um custo que é proporcional à distância entre a classe predita e a classe verdadeira. A distância adotada é o número de ligações entre os nós do menor caminho que liga as duas classes. Essa medida pode ser dividida em duas subcategorias: medida independente da profundidade e medida dependente da profundidade.

Na estrutura hierárquica em árvore o cálculo da distância é simples. Um exemplo dessa medida é mostrado na Figura 13. Observa-se que a distância entre classes M e G na hierarquia nessa estrutura é igual a três.

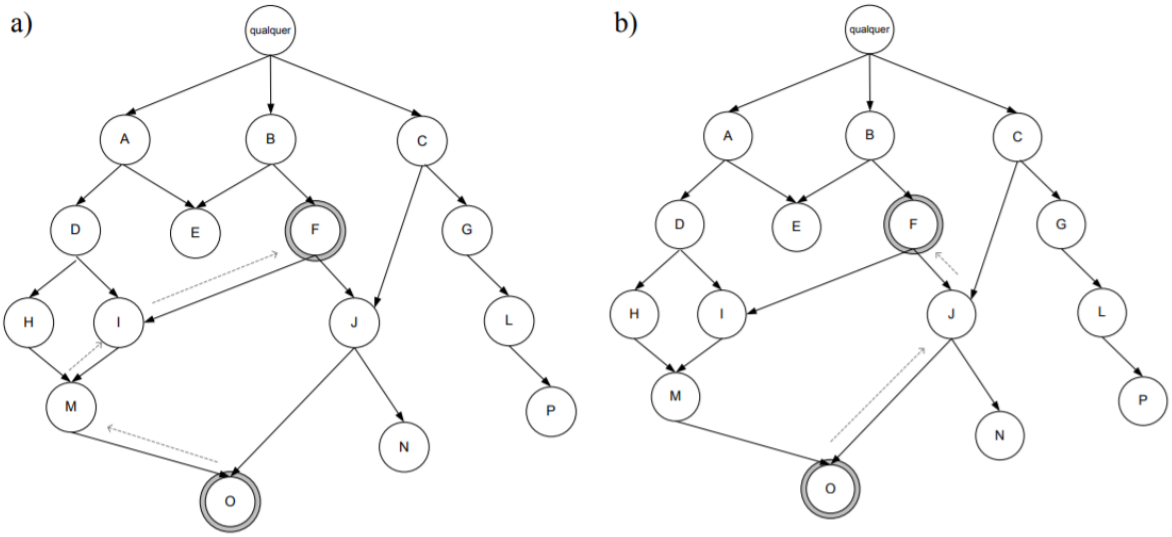
A situação é diferente quando fizermos o tratamento em um DAG, pois pode ter múltiplos caminhos entre o nó verdadeiro e o nó predito. Neste caso, é necessário a escolha de um critério para selecionar o caminho entre o nó de classe verdadeira e o nó de classe predito. Um critério que pode ser usado é o do menor caminho entre os dois nós. Por exemplo, há dois caminhos entre o nó de classe O e o nó F na Figura 14, o caminho selecionado seria o da Figura 14b cuja distância é igual a dois.

Figura 13 – Medida de Distancia em Árvore



Fonte: Borges (2012)

Figura 14 – Medida de Distância em DAG



Fonte: Borges (2012)

Medidas Baseadas na Relação de Ancestralidade e Descendência

No conceito de classes descendentes, cada subárvore é formada pela classe em si e pelas classes descendentes, ou seja, as suas subclasses. Segundo Ipeirotis (2001) as medidas de precisão e sensibilidade da hierarquia podem ser calculadas através da intersecção dessas subárvores. A medida de Precisão

Hierárquica Ph, é obtida pela divisão do número de descendentes comuns pelo número de descendentes da classe predita.

A medida de Sensibilidade Hierárquica Sh é obtida pela divisão do número de descendentes comuns pelo número de descendentes da classe verdadeira.

$$Ph = \frac{\left|Desc(C_v) \cap Desc(C_p)\right|}{\left|Desc(C_p)\right|}$$

$$Sh = \frac{\left|Desc(C_v) \cap Desc(C_p)\right|}{\left|Desc(C_v)\right|}$$

em que $Desc(C_v)$ representa a subárvore das classes descendentes a partir da classe verdadeira C (incluindo C), e $Desc(C_p)$ representa as classes descendentes (incluindo C) contidas na subárvore em que C é a classe raiz da classe predita. Essas medidas são então utilizadas no cálculo de uma extensão hierárquica da Medida-F.

$$Medida - Fh = \frac{(\beta^2 + 1) * Ph * Sh}{1 * Ph + Sh}$$

onde β geralmente recebe o valor igual a 1.

O problema dessa medida é que ela assume que a classe predita é uma classe descendente ou uma classe ancestral da classe verdadeira. Quando essas classes estão no mesmo nível, a intersecção é um conjunto vazio.

Em Kiritchenko (2004) é usado o conceito de ancestralidade. A medida de precisão Ph é obtida pela divisão do número de ancestrais comuns pelo número de ancestrais da classe predita. Já a medida de sensibilidade Sh é obtida pela divisão do número de ancestrais comuns pelo número de ancestrais da classe verdadeira. Em ambas as medidas, o nó raiz da árvore não é considerado ancestral da classe C.

$$Ph = \frac{\left|Anc(C_v) \cap Anc(C_p)\right|}{\left|Anc(C_p)\right|}$$

$$Sh = \frac{\left|Anc(C_v) \cap Anc(C_p)\right|}{\left|Anc(C_v)\right|}$$

ANEXO B - Teste Estatístico para Validação dos Resultados (Teste de Friedman)

TESTE ESTATÍSTICO PARA VALIDAÇÃO DOS RESULTADOS

Para validar os dados é necessário recorrer a testes estatísticos para comparar os experimentos realizados, pois estes fornecem respaldo científico, garantindo aceitabilidade no meio científico.

Os testes podem ser divididos em testes paramétricos e não paramétricos. Os testes paramétricos são baseados em suposições sobre a distribuição por trás da população em que os dados foram obtidos. Esses tipos de teste envolvem parâmetros populacionais, onde esses parâmetros são qualquer medida que descreva uma população (HOSKIN, 2011). Nos casos onde determinado conjunto de dados está distribuído de forma normal (distribuição normal), testes paramétricos podem ser aplicados.

Testes não paramétricos não dependem de nenhuma suposição sobre a forma ou os parâmetros de distribuição da população dos dados, assim, os testes realizados ficam livres de possíveis erros que poderiam ser causados por conta de premissas feitas sobre a distribuição (HOSKIN, 2011). Os testes não paramétricos são obtidos testando outras situações que não sejam parâmetros populacionais, como relacionamentos, modelos, dependência ou independência e aleatoriedade.

Teste de Friedman

O teste de Friedman é um teste não paramétrico (FRIEDMAN, 1940). Seu princípio básico é comparar proporções (possíveis divergências) entre frequências.

Este teste utiliza os rank dos dados ao invés de seus valores brutos para o cálculo da estatística de teste. Sendo o melhor algoritmo recebendo o rank 1, o segundo melhor o rank 2, e assim por diante. Em caso de empate, o rank atrelado será a soma dos ranks dos elementos empatados divididos pelo número de elementos somados. Exemplo: rank 2 e rank 3 empataram, logo ambos recebem o rank 2+3/2 = 2,5 como valor de rank.

Para o teste são necessários os seguintes dados:

- Hipótese Nula (H_0) – hipótese que é apresentada sobre determinados factos estatísticos, cuja falsidade se tenta provar.

- Hipótese Alternativa (H_1) – hipótese que é tomada como aceitável quando a hipótese nula é rejeitada.
- Nível de Significância (α) – significa o risco de se rejeitar uma hipótese verdadeira. Usualmente fixado em 5% (P=0,05).
- Graus de Liberdade (G.L.) – é a diferença entre o numero de classes de resultados e o número de informações da amostra que são necessários ao cálculo dos valores esperados nessas classes (normalmente se adota k-1).
- Número de algoritmos comparados (k) – quantidade de algoritmos que estão sendo comparados.
- Número de bases de dados (b) – quantidade de amostras (base de dados) colhidas sobre os algoritmos.

Com isso será possível aplicar a fórmula do Qui-quadrado de Friedman representado por S:

$$S = \frac{12b}{k(k+1)} \sum_{j=1}^{k} \left(\frac{R_j}{b} - \frac{k+1}{2} \right)^2 = \left[\frac{12}{bk(k+1)} \sum_{j=1}^{k} R_j^2 \right] - 3b(k+1)$$

Segundo Iman e Davenport (1980) citado por Demsar (2006) o teste de Friedman é considerado muito conservador, por isso sugere a utilização da estatística F_F distribuída de acordo com a distribuição F de Snedcor com k −1 e (k −1) ∗ (n −1) graus de liberdade.

$$F_F = \frac{(n-1)X_F^2}{n(k-1) - X_F^2}$$

Se a hipótese nula é rejeitada, Demsar (2006) sugere que seja feito o teste de Nemenyi, que calcula a distância crítica (CD - critical distance) entre desempenhos através da equação:

$$CD = q_a \sqrt{\frac{k(k+1)}{6N}}$$

em que os valores de q_a são tabelados conforme o nível de significância. Um algoritmo é considerado estatisticamente superior a outro se a diferença entre os postos médio de cada um for maior que a distância critica calculada.