UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DEEP LEARNING PARA CLASSIFICAÇÃO HIERÁRQUICA DE ELEMENTOS TRANSPONÍVEIS

FELIPE KENJI NAKANO

ORIENTADOR: PROF. DR. RICARDO CERRI

São Carlos – SP

Agosto/2018

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DEEP LEARNING PARA CLASSIFICAÇÃO HIERÁRQUICA DE ELEMENTOS TRANSPONÍVEIS

FELIPE KENJI NAKANO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação, área de concentração: Inteligência Artificial
Orientador: Prof. Dr. Ricardo Cerri

São Carlos – SP

Agosto/2018

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Dissertação de Mestrado do candidato Felipe Kenji Nakano, realizada em 05/09/2018:

Prof. Dr. Ricardo Cerri
UFSCar

Profa. Dra. Heloisa de Arruda Camargo
UFSCar

Prof. Dr. Carlos Nascimento Silla Junior
PUC-PR

Certifico que a defesa realizou-se com a participação à distância do(s) membro(s) Carlos Nascimento Silla Junior e, depois das arguições e deliberações realizadas, o(s) participante(s) à distância está(ao) de acordo com o conteúdo do parecer da banca examinadora redigido neste relatório de defesa.

Prof. Dr. Ricardo Cerri

AGRADECIMENTOS

Primeiramente, agradeço a minha família por ter me dado a oportunidade de estudar e me incentivar durante toda a vida. Também sou grato a minha namorada por estar sempre do meu lado quando precisei.

Aos meus companheiros dos laboratórios BioMaL e CIG, pelas inúmeras discussões, muitas das quais contribuíram ativamente para minha pesquisa. Além disso, também gostaria de destacar os pesquisadores da Universidade Federal de Minas Gerais, Gisele Pappa e Walter José Pinto, por contribuir ativamente nesta pesquisa, principalmente na concepção do projeto e obtenção dos conjuntos de dados.

Aos amigos da Universidade Estadual de Londrina que tive o privilégio de conviver grande parte deste mestrado.

Agradeço também ao meu orientador, Ricardo Cerri, por sempre estar presente quando precisei, e valiosas conversas sobre a pesquisa e decisões dentro da vida acadêmica.

Da mesma maneira, não poderia deixar de mencionar a professora Celine Vens, e agradecela por ter me orientado durante o período de intercâmbio na Bélgica. Suas ideias e motivação foram essenciais para o sucesso deste projeto.

Agradeço à Fundação de Amparo à Pesquisa de São Paulo pelos auxílios financeiros indispensáveis para a concretização deste mestrado (Processos 2017/19264-2 e 2016/12489-2). Igualmente, agradeço ao fomento fornecido pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior durante os primeiros meses de desenvolvimento desta pesquisa.

Por fim, agradeço a todos os funcionários da Universidade Federal de São Carlos e KU Leuven KULAK que contribuíram de maneira direta ou indireta para o sucesso deste projeto.

RESUMO

Elementos Transponíveis (TEs) são sequências de DNA que podem se mover de um local para outro dentro do genoma de uma célula. Eles contribuem para a diversidade genética das espécies, e seus mecanismos de transposição podem afetar a funcionalidade dos genes. A correta identificação e classificação de TEs é útil para a compreensão de seus efeitos nos genomas. Como existem propostas na literatura para organizar os TEs em uma taxonomia hierárquica, com superclasses e subclasses, este trabalho investigou métodos de Classificação Hierárquica (CH) de TEs utilizando Aprendizado de Máquina (AM), e Redes Neurais Artificiais (RNAs) do paradigma Deep Learning (DP). RNAs profundas têm-se mostrado promissoras em diversos campos de estudo, incluindo bioinformática. Inicialmente, sequências com TEs previamente identificados foram coletadas e estruturadas de acordo com taxonomias hierárquicas. Em seguida, as redes neurais Restricted Booltzman Machine, Denoising Auto-encoder, MultiLayer Perceptron e suas versões empilhadas foram testadas. De posse dos conjuntos de dados, diferentes métodos de classificação foram propostos, e comparados com métodos existentes na literatura. Como resultados dessa pesquisa, foram propostas duas novas estratégias, chamadas nLLCPN (Classificador Local por Nó Pai não Folha) e LCPNB (Classificador Local por Nó Pai e Ramo). Ambas adaptam a estratégia LCPN (Classificador Local por Nó Pai), de maneira que classificações em nós internos sejam permitidas. Adicionalmente, as redes profundas apresentam desempenho superior ou competitivo à arquiteturas rasas na maioria dos casos.

Palavras-chave: Elementos Transponíveis, Classificação Hierárquica, Deep Learning

ABSTRACT

Transposable Elements (TEs) are DNA sequences that can change its location within a cell's genome. They contribute directly to the genetic variety of species. Besides, their transposition mechanisms can affect the functionality of genes. The correct identification and classification of TEs play a central role in comprehension of genomes. Generally, identification and classification of TEs are performed using tools that employs homology, by comparing a sequence to many sequences from a labeled TE database. Since the literature proposes hierarchical taxonomies to classify TEs according to classes and subclasses, this project aims to develop new classification methods employing Machine Learning (AM) and Artificial Neural Networks (RNA) trained using Deep Learning (DP) concepts. Deep Neural Networks have extend the state-of-art of many field of study, including bioinformatics. As the first step, DNA sequences labelled with previously identified TEs will be collected and mapped according to hierarchies provided by the literature. Next, Deep Learning's neural networks Restricted Booltzman Machine, Auto-encoders, MultiLayer Perceptrons and their stacked version were tested. With these datasets, different classification methods are proposed and compared with literature's methods. As contributions, two new strategies were proposed, nLLCPN (*non-Leaf Local Classifier per Parent Node*) and LCPNB (*Classifier per Parent Node and Branch*). Both of then adapt LCPN (Local Classifier per Parent Node) in order to allow classifications in inner nodes. Additionally, the deep neural networks presented superior or competitive results in most of the cases.

Keywords: Transposable Elements, Hierarchical Classification, Deep Learning

GLOSSÁRIO

AE – *Auto-Encoder*

AM – *Aprendizado de Máquina*

BM – *Booltzman Machine*

BP – *Back-Propagation*

CD – *Constrastive Divergence*

DAE – *Denoising Auto Encoder*

DAG – *Grafo Acíclico Direcionado*

DBN – *Deep Belief Network*

DL – *Deep Learning*

DNA – *Ácido Desoxirribonucleico*

GD – *Gradiente Descendente*

HMC – *Hierarchical Multi-Label Classification*

HSC – *Hierarchical Single-Label Classification*

LCL – *Classificador Local por Nível*

LCN – *Classificador Local por Nó*

LCPNB – *Classificador Local por Nó Pai e Ramo*

LCPN – *Classificador Local por Nó Pai*

MLP – *Multi-Layer Perceptron*

PCT – *Predictive Clustering Trees*

RBM – *Restricted Booltzman Machine*

RM – *Repeat Masker*

RNA – *Redes Neurais Artificiais*

ReLu – *Rectified Linear Unit*

ReLu – *Rectified Linear Unit*

SAE – *Stacked Denoising Auto-Encoder*

SP – *Simple Prune*

SWV – *Sum of Weighted Votes*

TEs – *Elementos Transponíveis*

nlLCPN – *Classificador Local por Nó Pai Não Folha*

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

Capítulo 1

INTRODUÇÃO

Este capítulo apresenta o problema de classificação de TEs, introduzindo os principais conceitos e a motivação por trás de seu estudo, juntamente com as limitações dos métodos existentes na literatura. Também são apresentadas a hipótese e um breve resumo dos resultados atingidos nesta pesquisa.

Elementos Transponíveis (TEs) são sequências de DNA capazes de se mover dentro do genoma de um organismo. Durante muito tempo, TEs eram denominados como *Junk* DNA (DNA lixo), pois acreditava-se que sua presença não se manifestava no genoma (OHNO, 1972). Entretanto, trabalhos recentes revelam que a sua presença pode produzir diversos efeitos (KIM; LEE; HAN, 2012).

Até o momento, a classificação de TEs é realizada através da homologia, um método que busca alinhar sequências com outras previamente identificadas, todavia este apresenta deficiências. Segundo Costa (COSTA et al., 2013), a homologia utiliza informações estruturais e repetições nas sequências, fato que acarreta na propagação de erros e inutilização de propriedades bioquímicas. Ainda, vários destes são limitados a somente um subgrupo de TEs, e, o mais agravante, ignoram aspectos hierárquicos.

Com o intuito de facilitar sua identificação, e consequentemente, classificar TEs, taxonomias (hierarquias) que agrupam TEs são propostas. Atualmente a taxonomia de Wicker é considerada a mais adequada (WICKER et al., 2007). Esta agrupa TEs de acordo com suas características de maneira que TEs semelhantes sejam alocados em classes próximas na hierarquia.

Considerando as limitações citadas acima, e a possibilidade de empregar a taxonomia de Wicker, neste trabalho é proposto o estudo da classificação de TEs como um problema de Aprendizado de Máquina (AM), mais precisamente um problema de Classificação Hierárquica (CH).

De maneira similar a classificação tradicional, a CH estuda problemas cujas classes são estruturadas de acordo com uma hierarquia, como, por exemplo, a taxonomia de Wicker. Além dos benefícios do AM como criação automática de modelos e flexibilidade as características, ao incorporar CH, é possível utilizar as relações hierárquicas diretamente na solução (SILLA; FREITAS, 2009b).

A literatura oferece duas abordagens de CH: Global e Local. Usando somente um classificador, a abordagem Global adapta algoritmos da literatura, como *Naive-Bayes* ou Árvores de Decisão para lidar com a hierarquia do problema (SILLA; FREITAS, 2009a; VENS et al., 2008). Por sua vez, a abordagem Local reduz o problema para vários sub-problemas cuja complexidade é menor, em seguida, combina as suas soluções para resolver o problema de classificação. Ambas são consolidadas, entretanto a Local permite a utilização de qualquer classificador da literatura, incorporando seus pontos positivos.

No contexto de classificação, trabalhos recentes em AM mostram o potencial do novo paradigma *Deep Learning* (DL). Sua ideia central consiste em construir Redes Neurais Artificiais (RNAs) compostas por múltiplas camadas ocultas as quais são capazes de aprenderem distribuições mais representativas sobre os dados de entrada (HINTON; SALAKHUTDINOV, 2006; HINTON; OSINDERO; TEH, 2006; BENGIO et al., 2009). Esta capacidade impulsionou muitos campos de pesquisa, no contexto de Bioinformática, vários trabalhos já mostraram o seu potencial (MIN; LEE; YOON, 2016), entretanto poucas pesquisas estudaram sua aplicação em CH e TEs.

Nesta pesquisa, optou-se por utilizar três RNAs clássicas de DL: *Denoising* Auto-Encoders (SDA), Deep Multi-Layer Perceptrons (MLPs) e *Restricted Boltzmann Machine* (RBM). As duas primeiras são redes do tipo *Feed-Forward*, isso é, sinapses são propagadas sempre em direção a saída. Por sua vez, a RBM é uma rede recorrente cujas sinapses são propagadas somente entre as camadas de entrada e oculta.

Para atingir desempenho superior, SDAs e RBMs podem ser empilhadas, no sentido em que as representações apreendidas são sequencialmente refinadas com o intuito de facilitar a classificação. Por sua vez, Deep MLPs são extensões de suas versões rasas nas quais parâmetros especialmente desenvolvidos para *Deep Learning*, e naturalmente mais camadas ocultas, são utilizadas.

Desta maneira, neste trabalho investigou-se o problema de Classificação Hierárquica de TEs usando RNAs do paradigma DL associado à abordagem Local, ou seja, estratégias desta abordagem serão implementadas utilizando redes profundas como classificadores locais. Ao explorar estes tópicos de pesquisa, novos métodos de CH foram propostos e testados utilizando DL, seus resultados mostraram superioridade à homologia.

Sendo assim, neste mestrado, uma nova área de pesquisa é estudada, concomitantemente a aplicação de DL neste contexto é documentada. O restante deste capítulo é organizado da seguinte maneira: na Seção 1.1, são destacadas com mais detalhes as motivações desta pesquisa; as novas estratégias propostas são descritas na Seção 1.2; Em seguida, nossa hipótese e objetivo são introduzidos na Seção 1.3; Por fim, um resumo dos resultados, assim como a organização da dissertação são apresentados nas Seções 1.4 e 1.5 respectivamente.

1.1 Motivação

Atualmente sabe-se que TEs estão presentes em grande parte dos seres vivos, constituindo de 4% a 60% do genoma dos vertebrados (SOTERO-CAIO et al., 2017). Ao se transpor, um TE pode causar a remoção, inversão ou duplicação no genoma (SOTERO-CAIO et al., 2017), fato que altera significativamente o tamanho de um genoma, assim como sua evolução. Ainda, TEs podem funcionar como sequências regulatórias, forçando um genoma a responder a um estímulo externo (HAYASHI; YOSHIDA, 2009). No âmbito da agricultura, TEs também são explorados para potencializar o desenvolvimento da cultura de cana de açúcar cujo genoma contém muitos TEs ativos (JúNIOR et al., 2004; VETTORE et al., 2003). Sendo assim, a correta identificação e classificação apresenta potencial acadêmico e econômico.

Como especificado com mais detalhes no Capítulo 4, a literatura ainda não apresenta métodos de CH que reduzem a propagação de erros e permitem classificações *Não obrigatória* concomitantemente. Desta maneira, há grande motivação para a criação de métodos com estas características, visando sua aplicação em TEs.

Por fim, a revisão bibliográfica revelou que grande parte dos trabalhos de DL lidam com classificação tradicional. Motivados pelo seu alto desempenho, também buscou-se investigar como RNAs do paradigma DL se comportam neste contexto.

1.2 Métodos Propostos

Nesta pesquisa foram propostas duas nova estrategias de CH. Ambas tomam como base a estratégia Classificador Local Por Nó Pai (Subseção 4.2.2) no qual um classificador é treinado para cada nó pai da hierarquia. Apesar de ser consolidada na literatura, esta estratégia é sujeita a propagação de erros e não é capaz de predizer classificações *Não obrigatória*. Sendo assim, adaptações foram propostas.

Para possibilitar classificações *Não obrigatória*, a primeira estratégia, Classificador Local

por Nó Pai não Folha (nlLCPN), adiciona um nó fictício para cada classificador associado a um nó pai que também é uma classificação *Não obrigatória*, ou seja, supondo um nó pai 2.1, este é treinado para distinguir entre seus nós filhos e a si mesmo. Na fase de teste, utiliza-se o método tradicional da estratégia, porém caso um classificador prediga a classe associada a si mesmo, a fase de teste se encerra, e obtém-se uma classificação *Não obrigatória* automaticamente.

De maneira complementar, a segunda estratégia, denominada Classificador Local por Nó Pai e Ramo (LCPNB), utiliza a mesma técnica que a primeira para sua fase de treino (adição de nó fictício). A diferença está na fase de teste. Numa tentativa de reduzir a propagação de erros, um novo exemplo é testado por todos os classificadores, em seguida obtém-se a média das predições de probabilidade para todas as possíveis classificações. A classificação com maior média é dada como final.

1.3 Hipótese e Objetivo

Como mencionado anteriormente, o objetivo principal desta pesquisa consistiu em investigar DL para CH de TEs. Para tanto, trabalhou-se com duas hipóteses:

- Métodos de AM são preferíveis à homologia: ao utilizar AM, pretende-se provar que a criação de modelos automaticamente associada a capacidade de adicionar aspectos hierárquicos à solução do problema produz resultados estatisticamente superiores;
- Métodos DL são aplicáveis em problemas de CH: assume-se que RNAs com múltiplas camadas ocultas são substancialmente superiores a RNAs tradicionais no contexto de CH.

Considerando ambas as hipóteses, esta pesquisa foi desenvolvida de acordo com as seguintes etapas.

- Revisão Bibliográfica: durante o decorrer desta pesquisa, trabalhos relacionados foram constantemente estudados. Este levantamento permitiu uma análise comparativa, juntamente com a descoberta de lacunas na literatura;
- Coleta de Dados: considerando que esta pesquisa foi uma das primeiras a estudar a classificação de TEs como um problema de AM, primeiramente realizou-se a coleta e pré-processamento de conjunto de dados;
- Implementação de novos métodos: após realizar as etapas anteriores, observou-se deficiências nos métodos da literatura que poderiam inviabilizar a CH de TEs. Desta maneira, novos métodos que buscam saná-las foram propostos;

- Implementação de métodos de DL: com os resultados da etapa anterior, métodos de DL foram implementados e seu desempenho no contexto de CH foi testado;
- Avaliação de Resultados: levando em conta as peculiaridades do problema em questão, foi necessário a escolha de uma métrica de avaliação específica. De maneira complementar, testes estatísticos para validação dos métodos também foram investigados.

1.4 Resumo dos Resultados

Os experimentos desta pesquisa foram divididos em duas etapas:

- Homologia vs AM: inicialmente, como verificação de hipótese, investigou-se como métodos de AM, juntamente com as novas estratégias propostas, comportam-se ao ser comparados com a homologia. Os resultados favorecem claramente a aplicação de AM, mostrando superioridade estatística;
- Experimentos com *Deep Learning*: nesta experimentação, considerou-se como a adição de mais camadas pode melhorar o desempenho dos métodos de CH. Ao comparar DL com redes neurais rasas, comprovou-se que DL apresentam resultados estatisticamente superiores em alguns casos.

1.5 Organização da dissertação

Esta dissertação está organizada da seguinte maneira:

- Capítulo 2: apresenta uma descrição mais detalhada sobre TEs, juntamente com sua perspectiva histórica e relevância acadêmica;
- Capítulo 3: introduz o conceito de *Deep Learning*, assim como suas principais arquiteturas e o funcionamento de suas Redes Neurais;
- Capítulo 4: trata de aspectos fundamentais da Classificação Hierárquica, apresentando suas abordagens e trabalhos relacionados;
- Capítulo 5: contém uma descrição sobre as duas novas estratégias de CH propostas nesta pesquisa;
- Capítulo 6: traz os materiais utilizados com detalhes sobre obtenção do conjunto de dados, implementações e medidas de avaliação;

- Capítulo 7: discute os experimentos e seus resultados;
- Capítulo 8: encerra a dissertação com conclusões e trabalhos futuros.

Capítulo 2

ELEMENTOS TRANSPONÍVEIS

Este capítulo apresenta mais detalhadamente o conceito de Elementos Transponíveis, apontando sua importância em ambos âmbito acadêmico e econômico. Da mesma maneira, também apresenta os métodos para classificação de TEs na literatura. Estes possuem a homologia como base e ignoram aspectos fundamentais dos TES.

Elementos Transponíveis (TEs) são sequências de DNA capazes de se mover (transpor) dentro do genoma de uma célula. Sua descoberta é decorrente do trabalho (MCCLINTOCK, 1950) no qual variações na pigmentação de sementes de milho foram estudadas. A transposição de TEs dentro de um genoma contribui para a variabilidade genética de espécies, assim como para possíveis alterações na funcionalidades dos genes.

Apesar do extenso trabalho de McClintock, suas pesquisas ainda não possuíam reconhecimento acadêmico. Consequentemente, TEs foram ignorados entre as décadas de 60 e 90, e denominações como DNA lixo (*Junk DNA*) (OHNO, 1972), e DNA egoísta (*Selfish DNA*) (ORGEL; CRICK, 1980) eram utilizadas, todavia pesquisas recentes apontam que grande parte do DNA de diversas especies é composto por TEs, atingindo aproximadamente 45% em humanos (LANDER et al., 2001).

Adicionalmente, evidências apontam que TEs, através da evolução, tornaram-se componentes essenciais na modelagem de genomas (NAKAYASHIKI, 2011). No trabalho de Hayashi (HAYASHI; YOSHIDA, 2009), a inserção de um TE promoveu a ativação de um gene responsável pela defesa do organismo. Segundo Biemont (BIÉMONT; VIEIRA, 2006), TEs também podem afetar negativamente uma espécie, aproximadamente de 0.5% à 1% das doenças, incluindo tipos de câncer, são originadas pela transposição de TEs.

No contexto da Bioenergia, principalmente quanto à produção de Bioetanol, o estudo de TEs também é de grande importância. O projeto Genoma da cana-de-açúcar (Sucest) (JúNIOR et

al., 2004; VETTORE et al., 2003), revelou que existem muitos TEs ativos (expressos) nessa planta, considerada atualmente essencial para o desenvolvimento da economia brasileira.

Numa tentativa de facilitar a compreensão de TEs, uma organização foi proposta em (FINNEGAN, 1989) e (FINNEGAN, 1992). Esta divide os TEs em Classe I e Classe II de acordo com seus mecanismos de transposição.

Elementos da Classe I, também chamados de *retrotransposons*, utilizam um mecanismo "copia e cola". No primeiro passo, os *retrotransposons* são transcritos para Ácido ribonucleico [1], e em seguida, a enzima transcriptase reversa converte os elementos para DNA. Por fim, o TE é novamente inserido no genoma.

Por sua vez, integrantes da Classe II, *transposons*, movem-se por mecanismo de "recorta e cola". Diferentemente da Classe I, esse grupo se transpõe diretamente no DNA através da enzima transposase. A Figura 2.1 ilustra ambos os mecanismos de tranposição

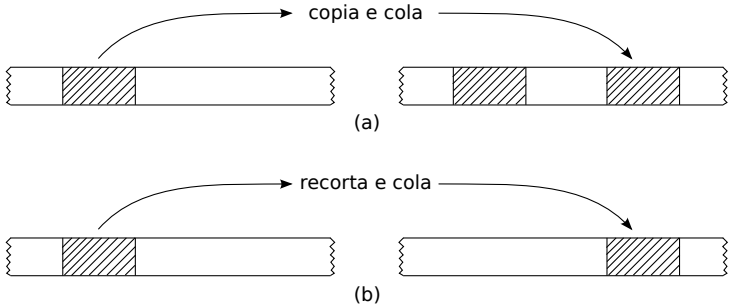


Figura 2.1: Ilustração dos mecanismos de transposição dos TEs: (a) transposição dos *retrotransposons* ("copia e cola"); (b) transposição dos *transposons* ("recorta e cola")

Além de elementos das Classes I e II, os TEs passaram a ser classificados em outras subclasses. Com o intuito de agrupá-los corretamente, taxonomias hierárquicas que estendem o trabalho de Finnegam foram criadas. Em (WICKER et al., 2007), é proposta uma taxonomia de TEs na qual suas classes são estruturadas em uma hierarquia. Esta hierarquia é apresentada parcialmente na Figura 2.2, na qual cada nó corresponde a um tipo de TE. De maneira similar à proposta de Finnegan, a taxonomia proposta por (WICKER et al., 2007) primeiramente divide os TEs de acordo com seus mecanismos de transposição (Classe I e Classe II). A Classe I é posteriormente subdividida em categorias denominadas ordens, de acordo com características como organização e filogenia da enzima Transcriptase Reversa. As ordens são posteriormente subdivididas em diferentes superfamílias. Os elementos da Classe II são subdivididos em duas subclasses I e II, também de acordo com características de transposição. Cada subclasse é posteriormente subdividida em ordens e superfamílias.

[1] Para evitar conflitos de sigla com Redes Neurais Artificiais (RNA), o Ácido ribonucleico é referenciado por sua nomenclatura completa

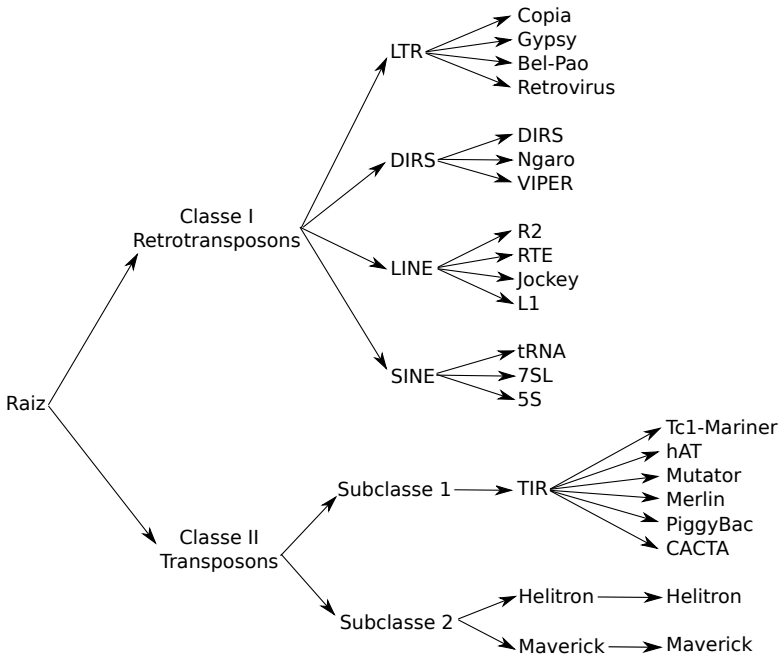


Figura 2.2: Taxonomia hierárquica para classificação o de TEs (NAKANO et al., 2017a)

A taxonomia de Wicker pode ser vista como uma tentativa de criar um sistema unificado para a classificação de TEs, entretanto Piegu (PIéGU et al., 2015) apresenta limitações. De acordo com os autores, em uma classificação biológica, é esperada a formação de grupos com base em relacionamentos evolutivos. Quando relacionamentos não podem ser encontrados entre indivíduos sendo classificados, os mesmos devem ser separados em grupos distintos e não relacionados como grupos de origens independentes. A taxonomia de Wicker falha quanto a isso, pois contém grupos de classes e subclasses não relacionadas filogeneticamente.

Seberg (SEBERG; PETERSEN, 2009) alega que, na taxonomia proposta por (WICKER et al., 2007), os limites entre alguns grupos perdem conexão com a filogenia. Por exemplo, TEs autônomos e não-autônomos pertencentes a uma mesma família são classificados em diferentes subfamílias. Wicker se defende, alegando que muitos TEs não-autônomos são derivados de TEs autônomos, ou então parasitam TEs autônomos. Outros TEs não-autônomos também parasitam TEs não relacionados (WICKER et al., 2009). Apesar de possuir limitações, a taxonomia de Wicker é a mais utilizada na literatura.

No contexto de classificação automática de TEs, grande parte dos métodos utilizam homologia (BERGMAN; QUESNEVILLE, 2007). Todavia, tais métodos possuem limitações, pois consideram que as sequências são muito similares (propagam classificações feitas incorretamente) ou são específicos para um determinado tipo de elemento (COSTA et al., 2013). Ainda, a homologia entre sequência ignora propriedades bioquímicas e relacionamento hierárquicos entre as famílias e superfamílias de TEs.

As próximas Seções apresentam o conceito de homologia, e também dois métodos muito utilizados para a detecção de sequências homólogas, BLAST (ALTSCHUL et al., 1990) e RepeatMasker (SMIT; HUBLEY; GREEN, 2015). Em seguida, outros métodos para classificação de TEs encontrados na literatura são revisados.

2.1 Homologia

Na bioinformática, a homologia, ou alinhamento de sequências, consiste de métodos para identificação de sequências similares de nucleotídeos ou aminoácidos (MOUNT, 2004). Esta assume que regiões similares são decorrentes de relações funcionais, estruturais ou evolutivas. Consequentemente, sequências que apresentam regiões com subsequências similares podem ser utilizadas para identificação de sequências homologas (NG; HENIKOFF, 2001).

O alinhamento de sequências curtas pode ser realizado de maneira manual, todavia aplicações do mundo real normalmente lidam com sequências longas e em grande quantidade. Desta maneira, pesquisas empregam abordagens computacionais.

Atualmente a literatura apresenta dois tipos métodos de homologia: global e local. Métodos globais buscam otimizar o alinhamento total da sequência, i.e. todos os nucleotídeos de sua extensão são levados em consideração. Por sua vez, métodos locais procuram subsequências similares. De maneira geral, métodos globais são preferíveis quando as sequências do conjunto de dados possuem mesmo comprimento, enquanto métodos locais podem ser aplicados em genes parcialmente anotados e também são mais eficientes em sequências que compartilham apenas regiões isoladas de similaridade (POLYANOVSKY; ROYTBERG; TUMANYAN, 2011).

Métodos de homologia também podem ser separados de acordo com a quantidade de sequências comparadas. Caso a comparação seja feita entre duas sequências, tem-se um alinhamento pareado. No caso de múltiplas sequências, tem-se um alinhamento múltiplo.

Como mencionado anteriormente, diversos métodos para classificação de TEs já foram propostos, entretanto, em grande parte, não são apropriados para esta pesquisa. Muitos destes são limitados somente para um subgrupo específico de TEs, por exemplo os métodos LTRDigest e LTR_FINDER são específicos para classificação de *Long Terminal Repeats Retrotransposons*. Da mesma maneira, os métodos RetroPred e TEClass limitam seu escopo as classes SINE e LINE. Ainda, alguns métodos são dependentes de vários componentes cuja acessibilidade é reduzida devido necessidade de adquirir *Software* específico. Por fim, e mais agravante segundo a perspectiva de Aprendizado de Máquina, os métodos são disponibilizados treinados de antemão usando um vasto conjunto de dados, inviabilizando sua aplicabilidade segundo a metodologia

clássica de treino e teste.

Devido a estes motivos, nesta pesquisa, optou-se por estudar os métodos mais populares de homologia aplicáveis a TEs, BLAST (*Basic Local Alignment Search Tool*) e RepeatMasker. Ambos são métodos locais e utilizam alinhamento múltiplo.

2.1.1 BLAST

Nos últimos anos, a quantidade de sequências de aminoácidos e DNA anotadas tem crescido rapidamente (POSADA, 2009), consequentemente sua anotação manual é inviável. Como alternativa computacional o método baseado em homologia, BLAST, destaca-se devido a sua eficiência.

Neste âmbito, o método BLAST, juntamente com suas variantes, é a ferramenta mais utilizada para a homologia (CASEY, 2005). Dada uma sequência desconhecida e um conjunto de dados com sequências previamente anotadas, o BLAST é capaz de encontrar uma sequência cuja composição se assemelha da sequência buscada (ALTSCHUL et al., 1990). Outras técnicas como (SMITH; WATERMAN, 1981) também são capazes de encontrarem sequências homólogas, entretanto não são aplicáveis em determinados cenários devido a sua alta complexidade computacional. Quando comparado à técnica de Smith-Waterman (SMITH; WATERMAN, 1981), o BLAST é consideravelmente mais rápido, por outro lado não garante o alinhamento ótimo (melhor solução).

O BLAST pode ser executado de maneira local ou online. Na opção online, os desenvolvedores disponibilizam um serviço no Website[2] no qual sequências podem ser submetidas para análise. Na versão local, o usuário também consegue fazer a análise, entretanto é necessário a utilização da ferramenta FormatDB para a conversão das sequências.

Como entrada, o BLAST requer um arquivo do tipo FASTA. Este lista todas sequências utilizando duas linhas para cada. Na primeira linha, o carácter $>$ é inserido juntamente com um identificador (id) para sequência. Na segunda linha, localiza-se a sequência associada ao identificador da linha anterior. A Figura 2.3 apresenta um exemplo de arquivo FASTA.

```
1 >ID_sequencia1
2 ATCGATCGATACGATACAT
3 >ID_sequencia2
4 ATCTAATCGGATC
5 >ID_sequencia3
6 AGGGTCGATACCGTAS
```

Figura 2.3: Exemplo de arquivo FASTA

[2]https://blast.ncbi.nlm.nih.gov/Blast.cgi

Quanto ao arquivo de saída, o BLAST apresenta várias possibilidades. Na versão online, a saída padrão é o HTML. Em sua versões local, outros formatos como XML e padrão texto podem ser utilizadas.

Como descrito por Mount (MOUNT, 2004), o funcionamento do BLAST pode ser sumarizado em três etapas:

- Remoção de regiões de baixa complexidade presentes na sequência buscada
- Encontrar regiões de alinhamento
- Reportar todas as sequências encontradas cuja pontuação é superior a um limiar

Inicialmente, regiões de baixa complexidade são removidas da sequências. Estas regiões não são representativas por possuir pouca variedade, consequentemente sua remoção é necessária para evitar alinhamentos triviais.

Em seguida, dada a sequência a ser alinhada, o BLAST vasculha o seu conjunto de dados de subsequências semelhantes. Logo após, as subsequências com maior pontuação (segundo uma matriz de pesos) são mantidas e expandidas em busca de alinhamentos maiores e também com maior pontuação. Por fim, os alinhamentos pontuação superior a um limiar são reportadas.

2.1.2 Repeat Masker

Repeat Masker é um software desenvolvido por A.R.A Smith capaz de detectar TEs de maneira eficiente. Segundo (TEMPEL, 2012), o Repeat Masker é a ferramenta mais utilizada para identificação de TES. Seu funcionamento consiste basicamente em analisar sequências de DNA comparando-as com um conjunto de dados previamente anotado por meio de algum motor de busca (por exemplo, BLAST), seguido de verificações de confiabilidade.

Atualmente, quatro motores de busca, todos baseados no BLAST, são disponibilizados para o Repeat Masker: AB-BLAST[3], RMBlast[4], Cross_Match[5], DeCypher[6] e HMMER (http://hmmer.org/).

A técnica AB-Blast consiste em uma variação do BLAST cuja velocidade de processamento e sensibilidade é superior. Por sua vez, a técnica RMBlast é equivalente ao Blast original com adição de recursos específicos para sua utilização pelo RepeatMasker. A técnica Cross_Match emprega uma versão otimizada do algoritmo Smith-Waterman (SMITH; WATERMAN, 1981). A

[3] http://blast.advbiocomp.com
[4] http://www.repeatmasker.org/RMBlast.html
[5] http://www.phrap.org/phredphrapconsed.html
[6] http://www.timelogic.com/catalog/755/decypher

técnica DeCypher é de propriedade da empresa TimeLogic e seus detalhes não são de livre acesso. A técnica HMMER implementa a técnica probabilística profile Cadeia Oculta de Markov para a busca de sequência homologas.

De maneira similar ao BLAST, o Repeat Masker utiliza arquivos no formato FASTA, e é disponibilizado em versões online e local. Ambas as versões são providas pelo Website[7]. A versão online é disponibilizada diretamente no site no qual sequências podem submetidas para análise. Esta versão não requer nenhuma instalação de software, entretanto possui limitações quanto a tamanho de sequências e motores de busca (a técnica DeCypher não está disponível). Por outro lado, a versão local permite sequências de qualquer tamanho, porém requer a instalação de todos os componentes do Repeat Masker.

O funcionamento do Repeat Masker é dividido em três etapas:

1. Execução do motor de busca;
2. Processamento dos resultados;
3. Gerar arquivos de saída.

Na primeira etapa, o RepeatMasker verifica configurações e o formato dos arquivos de entrada. Em seguida, separa as sequências em fragmentos, e executa o modos de busca utilizando parâmetros definidos automaticamente e padroniza seus arquivos de saída.

Na segunda etapa, os resultados do motor de busca são processados pelo módulo ProcessRepeats. Primeiramente, os fragmentos gerados na etapa anterior são pre-processados, eliminando redundâncias e alinhamentos triviais. Posteriormente, os fragmentos restantes são aglomerados para a formação de sequências. Por fim, sequências com pontuação, dada por Smith-Watermann (SMITH; WATERMAN, 1981), acima de um limiar são reportadas como solução.

2.2 Classificadores de Elementos Transponíveis

Diferentes métodos já foram propostos na literatura para a classificação de TEs. O método RetroPred (PK MITTAL VK, 2008) identifica e classifica TEs das classes LINE e SINE. Utilizando diversos softwares de detecção de TEs (PALS, PILER e MEME) juntamente com bancos de dados de referência, o RetroPred constrói uma matriz de pesos que serve como entrada para uma rede neural cuja saída corresponde às ordens LINE ou SINE. Apesar de cumprir seu propósito, este método é limitado somente a LINE e SINE.

[7] http://www.repeatmasker.org/

O método CENSOR (JURKA et al., 1996; KOHANY et al., 2006) classifica TEs usando sequências previamente anotadas. Como passo inicial, uma comparação rápida usando homologia é realizada. Neste passo, a nova sequência é comparada com uma banco de dados de sequência anotadas. Em seguida, um processo denominado *censoring* é realizado. Este consiste em marcar fragmentos de sequência homologas. Por fim, o algoritmo de busca local proposto em (SMITH; WATERMAN, 1981) é aplicado.

O método LTRDigest (STEINBISS et al., 2009) é específico para *Long Terminal Repeats Retrotransposons*. Inicialmente, LTR Retrotransposons são anotados com domínios proteicos (utilizando modelos ocultos de Markov) e outras regiões estruturais que buscam localizar a posição dos LTR Retrotransposons. O método pode então ser utilizado para a classificação não supervisionada, ou seja, encontrar grupos nos LTR Retrotransposons sem nenhum esquema de classificação pré-definido. Para avaliar se os grupos resultantes representam famílias conhecidas de LTR Retrotransposons, os autores compararam sequências representativas dos grupos de um conjunto de referência contendo sequências conhecidas.

O método LTR_FINDER (XU; WANG, 2007) também possui como foco LTRs *Long Terminal Repeats Retrotransposons*. Seu funcionamento é descrito em quatro etapas. Na primeira, pares de possíveis LTRs são selecionados. Estes são definidos através da busca de subsequências cuja semelhança é superior a um determinado valor pré-estipulado. Na segunda etapa, o algoritmo de Smith-Waterman (SMITH; WATERMAN, 1981) é utilizado para definir os limites de alinhamento. Em seguida, a terceira etapa detecta *Primer Binding Sites* e *Polypurine Tract* por meio de alinhamentos com t-Ácido ribonucleico e janelamentos de tamanho 15. A quarta etapa busca domínios (regiões de *Reverse Transcriptase*). Por fim, o LTR_FINDER reporta sequências com possíveis LTR anotados.

O método apresentado por (ROUAULT et al., 2009) toma como base um algoritmo de agrupamento hierárquico combinado com medidas de distância para aglomerar e classificar TEs de acordo com a hierarquia da superfamília *Mariner*. Inicialmente, todas as sequências são igualmente separadas. De maneira gradativa grupos são formados de acordo com a similaridade de cada cadeia de DNA. Por fim, para classificar um novo elemento, o algoritmo percorre cada classe da hierarquia. Caso a distância entre o novo elemento e um nó da hierarquia seja menor que um limiar, a subclasse é explorada, caso contrário é ignorada. Este processo se repete até que não existam mais nós a serem explorados.

Em (FESCHOTTE et al., 2010), foi proposto o método RepClass que consiste de três módulos independentes de classificação: um módulo baseado em homologia, um módulo que busca por características estruturais tais como LTRs e TIR (*Inverted Terminal Repeats*) e um módulo que

busca por duplicações em sítios alvo. Os três geram classificações em diferentes níveis de granularidade. Por fim, um módulo de integração tem a finalidade de comparar, ranquear, e combinar os resultados obtidos pelos três módulos anteriores, obtendo uma única classificação.

No método ncRNAclassifier (TEMPEL; POLLET; TAHI, 2012), os TEs presentes em *hairpins* de Ácido ribonucleico são detectados e classificados. Primeiramente, um *pipeline* de programas de identificação de TEs. Em seguida, com base em limiares definidos pelo usuário, a classificação é feita. Apesar de ser efetivo em vários tipos de genomas (sapos, humanos, ratos, entre outros), o método não lida com a relação hierárquica entre as famílias e superfamílias de TEs, e é limitado somente a TEs associados ao Ácido ribonucleico.

No método PASTEC (HOEDE et al., 2014), diferentes características são combinadas para a classificação de TEs, como características estruturais (comprimento das sequências, a presença de LTRs ou TIRs, e presença de uma única sequência repetida), homologia, e domínios conservados obtidos em bancos de perfis de modelos ocultos de Markov. Por fim, todas as classificações da hierarquia de Wicker são verificadas, aquela com maiores similaridade segundo suas características é dada como classificação.

Capítulo 3

REDES NEURAIS ARTIFICIAIS

Este capítulo introduz o conceito de Redes Neurais Artificiais, apresentando as inspirações biológicas envolvidas em seu desenvolvido. O paradigma Deep Learning também é apresentado, juntamente com as redes neurais Restricted Booltzman Machine e Auto-encoder.

3.1 Introdução

Na literatura, diversas definições de Redes Neurais Artificiais (RNA) são encontradas. Segundo Haykin (HAYKIN, 1998), uma RNA pode ser definida como um processador distribuído e paralelo composto por várias unidades simples de processamento cujas propriedades naturais são capazes de armazenar conhecimento provido de experiências passadas. No trabalho de Grossberg (GROSSBERG, 1988), RNA são definidas como algoritmos adaptativos que processam dados de maneira não linear e combinam unidades de processamento em diversas camadas. Segundo Russel (RUSSELL; NORVIG, 2003), uma rede neural corresponde a um conjunto de neurônios interligados, por sua vez, neurônios são denominados como elementos capazes de computar simples operações matemáticas.

Utilizadas amplamente em problemas de Aprendizado de Máquina, RNAs tem seu funcionamento inspirado no cérebro humano. O cérebro humano é composto de aproximadamente 10^{11} neurônios que estão conectados com cerca de 10^4 outros neurônios (DEMUTH et al., 2014). Por sua vez, um neurônio é composto por três componentes principais: dendritos, corpo celular e axônios. Suas funcionalidades são descritas abaixo:

- Dendritos: fibras nervosas responsáveis por receberem e levarem estímulos até o corpo celular;

- Corpo Celular: unidade de processamento responsável por somar e estabelecer um limiar dos estímulos recebidos;
- Axônio: uma longa fibra nervosa responsável por levar o sinal produzido pelo corpo celular até outro neurônio.

De maneira complementar, o ponto de contato entre o axônio de um neurônio e o dendrito de outro é denominado sinapse. Sendo assim, o arranjo de neurônios e a força das sinapses determinam o funcionamento de uma rede neural. A Figura 3.1 apresenta os componentes de um neurônio.

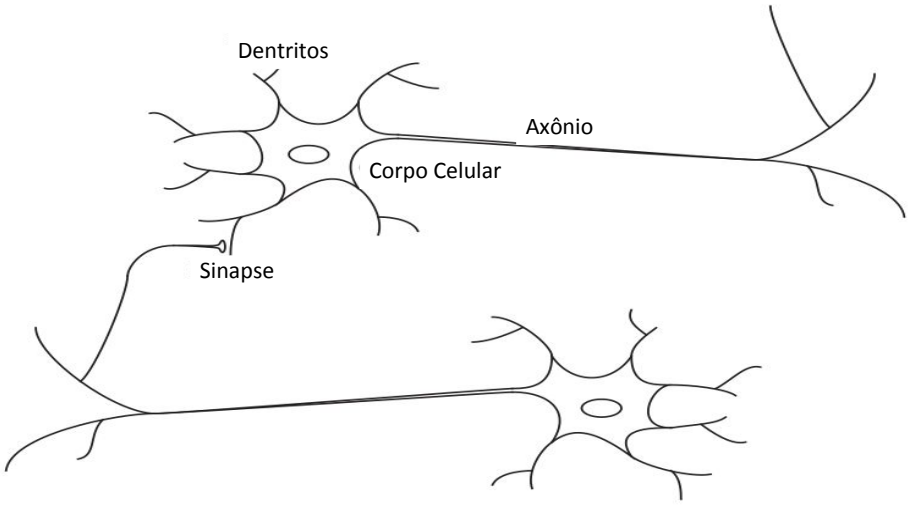


Figura 3.1: Representação de um neurônio. Adaptado de (DEMUTH et al., 2014)

Atualmente, a literatura apresenta uma vasta quantidade de tipos de redes neurais, entretanto o primeiro trabalho apresentava um modelo muito simples denominado Perceptron. A Seção a seguir traz mais detalhes sobre esta RNA.

3.1.1 Perceptron

A primeira RNA, denominada Perceptron, foi proposta por McCulloch (MCCULLOCH; PITTS, 1943). Esta é considerada como a primeira abordagem que utiliza a ideia de neurônios como unidades de processamento. De maneira simples, sua arquitetura é constituída somente de um neurônio que recebe estímulos do ambiente (dados de entrada) e os utiliza para construir um hiperplano capaz de separar dados em duas classes.

Como a Figura 3.2 mostra, o funcionamento do Perceptron consiste em associar um peso sináptico para cada atributo j do conjunto de dados. Para cada exemplo, os pesos são multiplicados pelo valor do atributo associado e somados, este valor somado corresponde à sinapse do

neurônio. No contexto de RNAs, a sinapse é denominada como valor de ativação. Também é necessário a adição de um peso extra, nomeado *Bias* (b), responsável por aumentar ou diminuir o valor de ativação. A Equação 3.1 representa o cálculo do valor de ativação.

$$\gamma = \sum_{j=0}^{m} w_j x_j \tag{3.1}$$

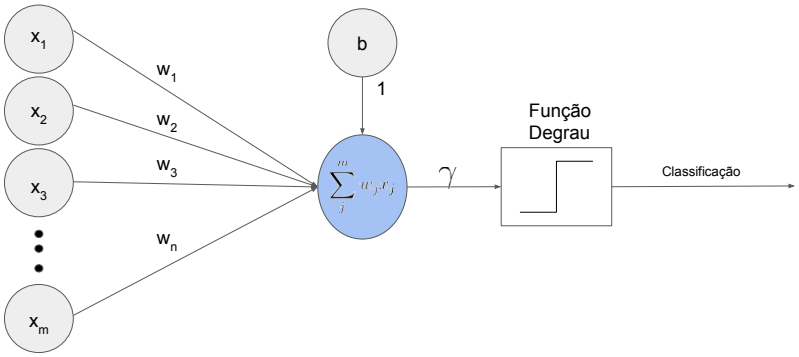


Figura 3.2: Funcionamento da RNA Perceptron

Nesta situação, w^0 corresponde ao peso associado ao *Bias*. Após obter o valor de ativação, γ, aplica-se uma função de ativação. No caso do Perceptron, normalmente, utiliza-se a função degrau definida na Equação 3.2, para obter a saída do Perceptron, $\upsilon(n)$.

$$\begin{cases} 1 & \text{se } \sum_{j=0}^{m} w_j x_j \geq 0 \\ 0, & \text{se } \sum_{j=0}^{m} w_j x_j < 0 \end{cases} \tag{3.2}$$

Durante a fase de treino do Perceptron, os valores de ativação são calculados para cada exemplo e com base no erro obtido, por meio da Equação 3.3, os pesos sinápticos são atualizados. Neste caso, n corresponde ao exemplo de treinamento, $e(n)$ ao erro, $d(n)$ ao valor desejado (classe do exemplo) e $\upsilon(n)$ ao valor obtido pelo Perceptron.

$$e(n) = d(n) - \upsilon(n) \tag{3.3}$$

O ajuste dos pesos segue a Equação 3.4 na qual $w_{anterior}$ corresponde ao peso anterior, w_{novo} o peso novo, $e(n)$ ao Erro (Equação 3.3) e x_j ao valor do atributo do exemplo. A Equação (3.4)

é aplicada a cada peso do Perceptron.

$$w_{novo_j} = w_{anterior_j} + e(n)x_j \tag{3.4}$$

Para classificar um novo exemplo, também utiliza-se a Equação 3.1, porém, aplica-se uma função de ativação (Equação 3.2). Normalmente, para Perceptrons, aplica-se a função degrau, definida na Equação 3.2. Caso o valor de saída seja maior que 0 atribui-se uma classificação, em caso contrário, outra classificação é obtida.

Apesar do Perceptron ter iniciado pesquisas com RNAs, sua aplicação mostrou-se precária em vários contextos, principalmente naqueles que envolvem problemas não linearmente separáveis. Sendo assim, avanços levaram a criação de estruturas mais complexas. Em especial, arquiteturas que utilizam uma ou mais camadas intermediárias, como a Multi-Layer Perceptron.

3.1.2 Multi-Layer Perceptron

No trabalho de Rumelhart (RUMELHART; HINTON; WILLIAMS, 1986), foi proposta uma rede neural com várias camadas de Perceptrons chamada MLP (*Multi Layer Perceptron*). Esta é denominada como *Feed-Forward* devido a propagação de sinapses em direção a camada de saída. Sua capacidade de resolução de problemas é maior, entretanto requer algoritmos de atualização de pesos mais complexos, como o *Back Propagation*. Normalmente a arquitetura de MLPs consiste em três camadas: inicial, intermediária e de saída, totalmente conectadas por pesos sinápticos (W). A Figura 3.3 retrata uma arquitetura para MLPs.

3.1.2.1 Back Propagation

O treinamento de MLPs é realizado com algoritmos de otimização de funções, como o *Back Propagation* (BP) (RUMELHART; HINTON; WILLIAMS, 1986). Seu funcionamento é separado em duas etapas, a fase de propagação e a fase de ajustes de pesos retro-propagação (HAYKIN, 1998).

Na fase de propagação, os neurônios da camada inicial são responsáveis por receber os estímulos do ambiente (dados) e propagá-los em direção à camada intermediária. Como MLPs apresentam mais camadas, é adicionado um índice i para referencia-las. Sendo assim, o valor de ativação é calculado usando a Equação 3.5.

$$\gamma = \sum_{j=0}^{m} w_{ji}x_{ji} \tag{3.5}$$

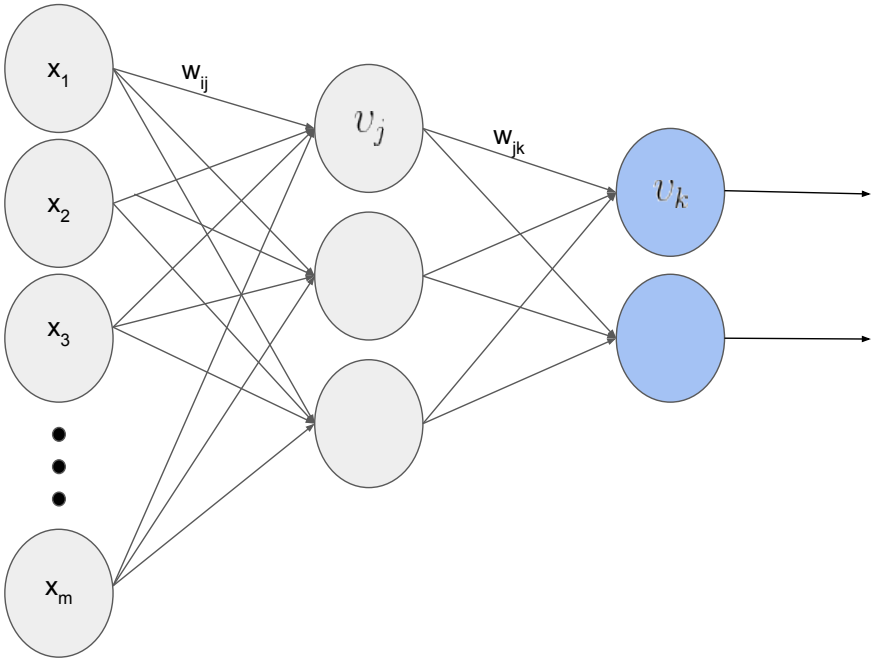


Figura 3.3: Arquitetura tradicional da Multi Layer Perceptron.

Ambas as camadas intermediária e final recebem estímulos das camadas anteriores, entretanto aplica-se uma função de ativação (φ) nos valores de ativação (Equação 3.5). Estas definem a amplitude da sinapse do neurônio e influenciam o ajuste dos pesos. Na Equação 3.6, o termo $\upsilon_j(n)$ representa a sinapse produzida pela função de ativação φ aplicada sobre o valor de ativação γ.

$$\upsilon_j(n) = \varphi_j(\gamma_j(n)) \tag{3.6}$$

No caso de MLPs, normalmente a função sigmoide (Equação 3.7) é utilizada (HAYKIN, 1998). Desta maneira, a camada intermediária recebe as saídas provenientes da camada inicial, combina-as, utilizando a Equação 3.6, aplica a Equação 3.7 e propaga as sinapses para a camada final.

$$sigmoide(n) = \frac{1}{(1+e^{-n})} \tag{3.7}$$

Por sua vez, a camada final realiza o mesmo processo, porém usa as sinapses da camada intermediária como entrada para produzir a classificação da MLP. A Equação 3.8, no qual w_{jk} representa o peso sináptico associado ao neurônio j da camada k, e $\gamma_j(n)$ a sinapse obtida da camada anterior, apresenta o cálculo do valor de ativação para neurônios da camada final. No

caso de uma arquitetura com mais de uma camada, a Equação 3.8 também é aplicada para propagação entre camadas ocultas.

$$\upsilon_k(n) = \sum_{j=0}^{m} w_{jk}(n)\gamma_j(n) \tag{3.8}$$

Após obter as sinapses da camada de saída, inicia-se a fase de ajustes e pesos. Da mesma maneira que o Perceptron, primeiramente calcula-se o erro produzido pela RNA. No caso de MLPs, utiliza-se o Erro Quadrático Médio (Equação 3.10). Para cada neurônio da camada de saída (j) e exemplo n, utiliza-se a Equação 3.9 para obter o valor de erro. Em seguida estes são somados e normalizados segundo a quantidade de exemplos no conjunto de dados (N).

$$\xi(n) = \frac{1}{2}\sum_{j} e_j^2(n) \tag{3.9}$$

$$\xi_{medio} = \frac{1}{N}\sum_{n=1}^{N} \xi(n) \tag{3.10}$$

Nota-se que o Erro Quadrático consiste de uma função baseada nos pesos sinápticos (W) e termos *Bias* da MLP. Consequentemente, a sua minimização produz pesos que apresentam taxas de erro baixas. Normalmente, esta minimização é realizada por meio do Gradiente Descendente (GD).

O GD é um método de otimização de primeira ordem iterativo na qual minimiza-se uma função de custo multivariada utilizando o gradiente negativo da função com respeito a variável de interesse (RUDER, 2016). A cada iteração, toma-se um passo em direção ao gradiente negativo de maneira que a variável de interesse passa a se aproximar ao valor desejado.

Sendo a^q o valor da variável na iteração q, $a^q + 1$ o valor da variável na próxima iteração, α o tamanho do passo e $\Delta F(a^q)$ o gradiente da função de custo com respeito a variável a na iteração q, a Equação 3.11 apresenta o cálculo do GD.

$$a^{q+1} = a^q - \alpha\Delta F(a^q) \tag{3.11}$$

No contexto de MLPs, a função de custo, na maioria das vezes, consiste no Erro Quadrático Médio (Equação 3.10) e a variável de interesse corresponde aos pesos sinápticos w (termos *Bias* incluídos). Intuitivamente, ao atualizar os pesos empregando o Gradiente Descendente, a RNA gradualmente modela o valor de seus pesos para que seu erro seja mínimo.

Ainda, os termos α e $\Delta F(a^q)$ são associados para formar o termo $\Delta w_{ji}(n)$. Esse é utilizado

para a definição da Regra Delta que dita a atualização de pesos em MLPs. As Equações 3.12 e 3.13 apresentam o cálculo do termo $\Delta w_{ji}(n))$ e a Regra Delta respectivamente.

$$\Delta w_{ji}(n)) = -\alpha \frac{\partial \xi(n)}{\partial w_{ji}(n)} \tag{3.12}$$

$$w_{novo_{ji}} = w_{antigo_{ji}} - \Delta w_{ji}(n) \tag{3.13}$$

De maneira similar ao Perceptron, o algoritmo BP ajusta os pesos de acordo com o erro obtido. No caso dos neurônios da camada de saída, este é diretamente calculável visto que a saída desejável é provida pelo conjunto de dados, porém o mesmo não se aplica a camada oculta. Sendo assim, são necessárias duas fórmulas para atualizar os pesos. Neste trabalho, optou-se por uma apresentação mais concisa. Caso o leitor esteja interessado na motivação por trás das fórmulas, a obra de Haykin (HAYKIN, 1998) contem definições com embasamento matemático.

Para facilitar o ajuste dos pesos, define-se a Equação 3.14 no qual o termo $\delta_j(n)$, denominado gradiente, varia de acordo com a camada a ser atualizada e o valor de $\gamma_i(n)$ é definido na Equação 3.5.

$$\Delta_{ji}(n) = \alpha \delta_j(n) \gamma_i(n) \tag{3.14}$$

Ao lidar com a camada final, utiliza-se $\delta_j(n)$ obtido por meio da Equação 3.15. Nesta situação, o valor de $e_j(n)$ é calculado diretamente a partir da Equação 3.3, o termo $\varphi\prime$ consiste na derivada da função de ativação (Equação 3.7) e $\upsilon_j(n)$ refere-se ao valor de ativação, Equação 3.1.

$$\delta_j(n) = e_j(n) \varphi\prime_j(\upsilon_j(n)) \tag{3.15}$$

Como mencionado, considerando que não é possível calcular o erro diretamente para a camada intermediária, estima-se o seu valor através dos erros da camada de saída. A Equação 3.16 apresenta o calculo do gradiente para camada intermediárias, neste caso o termo $\varphi\prime_j(\upsilon_j(n))$ é obtido da mesma maneira que na Equação 3.15, porém com respeito a camada em questão e $\sum_h \delta_k(n) w_{kj}(n)$ refere-se ao próprio gradiente proveniente dos neurônios k da camada de saída.

$$\delta_j(n) = \varphi\prime_j(\upsilon_j(n)) \sum_h \delta_k(n) w_{kj}(n) \tag{3.16}$$

Para RNAs em geral, o termo α é denominado como constante de aprendizado e seu valor está definido em $[0, 1]$. No caso de valores baixos, o treinamento tende a ser lento, fato que acarreta em muitas épocas de treino. Na situação oposta, a MLP aprende mais rápido, porém tende a produzir taxas de erro oscilantes.

Ao treinar MLPs, também deve-se considerar quando parar o treinamento. A literatura ainda não apresenta um consenso, entretanto, na maioria das vezes, o treinamento é dado por completo ao atingir um número predeterminado de épocas ou quando o erro Médio 3.10 entre duas épocas é inferior ao determinado limiar.

Definir um critério de parada é essencial para o desempenho de RNAs em geral, pois caso o treinamento seja interrompido precocemente, aspectos importantes do conjunto de dados não são aprendidos pela RNA, obtendo resultados insatisfatórios. Esse problema é denominado *Underfitting*, sua definição formal é dada na Definição 1.

Definição 1. *O problema de Underfitting ocorre quando um classificador não consegue aprender a distribuição dos dados, resultando em taxas de erro altas para ambos conjuntos de teste e treino (TAN; STEINBACH; KUMAR, 2005).*

Em contrapartida, caso o treinamento seja estendido além do necessário, a RNA memoriza o conjunto de dados e não aprende corretamente como classificar novos exemplos. Esse fenômeno recebe o nome de *OverFitting* (Definição 2)

Definição 2. *O Overfitting ocorre quando um classificador se adéqua excessivamente ao conjunto de treino, resultando em taxas de erro baixas no conjunto de treino, porém altas no conjunto de teste.*

Recentemente, devido ao avanço do *Deep Learning*, extensões do algoritmo BP com melhor poder de representação como Adam (KINGMA; BA, 2014a) foi proposto. A Subseção seguinte detalha o seu funcionamento.

3.1.2.2 Adam

O algoritmo (KINGMA; BA, 2014a), assim como o BP, aplica gradientes para otimização de uma função. Sua diferença consiste na utilização de taxas de aprendizado (α) individuais para diferentes parâmetros estimados a partir do primeiro e segundo momento do gradiente. De maneira geral, o algoritmo emprega parâmetros que tornam a atualização dos pesos robustos a gradientes esparsos.

A atualização dos pesos segue uma implementação direta e intuitiva. Com o intuito de facilitar o compreendimento do algoritmo, a lista a seguir sumariza cada termo utilizado:

- m_t = Estimativa do primeiro momento na iteração t, inicialmente 0
- v_t = Estimativa do segundo momento na iteração t, inicialmente 0
- g_t = Gradiente do erro calculado na iteração t
- $\hat{m}_t$ = Estimativa corrigida do primeiro momento na iteração t
- $\hat{v}_t$ = Estimativa corrigida do segundo momento na iteração t
- β_1 = Taxa de decaimento exponencial do primeiro momento
- β_2 = Taxa de decaimento exponencial do segundo momento
- ε = Constante para estabilidade numérica
- t = Contador de iterações, inicialmente 0
- α = Constante de aprendizado

Inicialmente, calcula-se o gradiente da mesma maneira que o algoritmo BP (Equações 3.15 3.16), todavia deve-se calcular um vetor de gradientes (g_t), de maneira que cada posição corresponda a um gradiente. Em seguida, utiliza-se a Equação 3.17 para obter o termo m_t.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \tag{3.17}$$

Da mesma maneira, o segundo momento é estimado usando a Equação 3.18. Nota-se que g_t^2 corresponde ao gradiente elevado a segunda potência elemento a elemento.

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \tag{3.18}$$

Posteriormente ambos os momentos são corrigidos com base nas Equações 3.19 e 3.20. No caso β_1^t e β_2^t correspondem a seus valores elevados a potência t. Esta correção garante que o ajuste seja gradativamente menor.

$$\hat{m}_t = \frac{m_t}{(1-\beta_1^t)} \tag{3.19}$$

$$\hat{v}_t = \frac{v_t}{(1-\beta_2^t)} \tag{3.20}$$

Por fim, seguindo a notação da equação do ajuste de pesos da MLP (Equação 3.13), o termo Δ_{ji} é obtido por meio da Equação 3.21.

$$\Delta_{ji} = \frac{\alpha \hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \tag{3.21}$$

3.2 Deep Learning

Durante muito tempo, arquiteturas rasas, com somente uma camada intermediária, foram utilizadas para solucionar problemas de Aprendizado de Máquina. Apesar do seu sucesso, principalmente através da RNA MLP, em muitos problemas do mundo real, seu poder representativo não é suficiente (DENG; YU et al., 2014). Em virtude disto, arquiteturas profundas, com muitas camadas intermediárias, passaram a ser foco de pesquisas.

Ao analisar o cérebro, percebe-se claramente a existência de várias camadas de neurônios capazes de processar informações captadas pela audição e visão com o intuito de interpretá-las (KRUGER et al., 2013). Intuitivamente, RNAs deveriam explorar o mesmo tipo de arquitetura. Seguindo esta lógica, RNAs com múltiplas camadas intermediárias, também chamadas de redes profundas, passaram a ser estudadas no paradigma *Deep Learning*.

Algoritmos tradicionais de treinamento como o *Back-Progation* não são diretamente aplicáveis em redes profundas. Segundo Bengio (BENGIO et al., 2009), em arquiteturas profundas, o algoritmo BP tende a ficar preso em mínimos locais, obtendo desempenhos inferiores à arquiteturas rasas. Usando pesos inicializados aleatoriamente, as camadas mais próximas da entrada tendem a não ser otimizadas. Em alguns casos, a taxa de erro no conjunto de treino pode ser relativamente pequena, entretanto a rede falha no quesito de generalização, obtendo resultados indesejáveis. Este problema é decorrente do valores inconsistentes do gradiente produzido pelos erros. Em muitos casos, o valor do gradiente é muito grande e, consequentemente, os pesos são super ajustados (*Overfitting* Definição 2). Na situação oposta, quando o valor do gradiente é baixo, os pesos não são ajustados devidamente resultando em *Underfitting* (Definição 1) (SCHMIDHUBER, 2015; GLOROT; BENGIO, 2010). Em ambas situações, o modelo tende a não apresentar desempenho satisfatório.

A subotimização produzida pelos algoritmos de arquiteturas rasas, juntamente com sua alta complexidade computacional, impossibilitou consideravelmente o estudo de RNAs com arqui-

teturas profundas (BENGIO et al., 2007). Entretanto, os avanços produzidos nos trabalhos de (HINTON; OSINDERO; TEH, 2006) e (HINTON; SALAKHUTDINOV, 2006) possibilitaram a criação de RNAs capazes de construir arquiteturas profundas com alta generalização e desempenho. A principal diferença consiste em empilhar várias RNAs treinadas gananciosamente camada a camada. As duas principais RNAs são RBM (Restricted Booltzman Machine) (HINTON; SALAKHUTDINOV, 2006) e AE (Auto-Encoder) (HINTON; OSINDERO; TEH, 2006).

De maneira simples, a RBM consiste em uma rede neural estocástica de duas camadas capaz de aprender uma distribuição de probabilidade com base nos dados de entrada. Por sua vez, AE são redes neurais *Feed-Forward* também de duas camadas cuja camada intermediária é treinada para aprender uma codificação dos dados de entrada. As Subseções 3.2.1 e 3.2.3 apresentam mais informações.

Como proposto nos trabalhos de Hinton, a construção de redes profundas é divida em duas fases, a fase de pré-treino, responsável por gerar os valores iniciais do pesos sinápticos e a fase de ajuste fino usada para finalizar o ajuste dos pesos. Na primeira fase, RBMs ou AE são acopladas diretamente umas as outras (empilhadas), ou seja, a saída de uma rede é diretamente usada para alimentar a próxima. O treinamento não supervisionado destas RNAS é realizado de maneira gananciosa, uma camada por vez. Inicialmente a primeira rede é treinada usando os dados de entrada diretamente, em seguida a segunda rede utiliza os dados processados pela primeira para realizar seu treinamento. Este processo se repete até a última rede. Por fim, obtém uma RNA profunda composta por várias outras RNAs empilhadas usada como inicialização.

Na segunda fase, adiciona-se uma camada de saída referente as possíveis classificações. Para ajustar os pesos desta camada, algoritmos como BP pode ser usado. Novamente o mesmo procedimento é adotado, i.e. os dados são alimentados a primeira RNA e propagados até a camada de saída. Ao utilizar estes algoritmos com a inicialização feita na primeira fase, o tempo de treino é consideravelmente reduzido. Além disso, ao contrário das inicializações aleatórias, é muito improvável que a RNA profunda inicializada com pré-treino não supervisionado fique presa em mínimos locais (BENGIO et al., 2007).

3.2.1 Restricted Booltzman Machines

Assim como o nome sugere, RBMs são adaptações da BM (Booltzman Machine) tradicional proposta em (HINTON; SEJNOWSKI, 1986). Por sua vez, BMs são modelos baseados em energia e consistem em uma versão modificada das redes de Hopfield (HOPFIELD, 1982) e foi umas das primeiras RNAs a buscar aprender uma distribuição de probabilidade. Para cada configuração da rede há um valor de energia associado (LECUN et al., 2006). Sendo assim, o

aprendizado da rede consiste em modelar o valor da energia de acordo com as propriedades desejadas (BENGIO et al., 2009). Nesse caso, a função de probabilidade utilizada é apresentada na Equação 3.22 na qual x representa a configuração de entrada e h a configuração da camada oculta.

$$P(x) = \frac{e^{-Energy(x,h)}}{Z} \tag{3.22}$$

Como mostrado na Equação 3.22, a probabilidade de uma determinada configuração é baseada em sua energia, em que valores altos de energia representam probabilidades baixas, consequentemente configurações desejáveis possem valores baixos de energia.

O fator normalizante Z garante que o valor produzido representa uma probabilidade. O valor de Z é obtido através da soma da energia de todos os possíveis estados do modelo. A Equação 3.23 representa como o valor de Z é calculado.

$$Z = \sum_{x} e^{-Energy(x)} \tag{3.23}$$

A arquitetura das BM consiste de duas camadas, uma de entrada e uma intermediária (oculta), cujos neurônios são associados completamente, utilizando pesos sinápticos forçando dependências entre neurônios. Esta RNA inicial apresentou muitas deficiências, pois o tempo de treino crescia exponencialmente com os dados e o ruído afetava consideravelmente seu treinamento, inviabilizando sua aplicação em problemas reais. A Figura 3.4 demonstra a arquiteturas de BM.

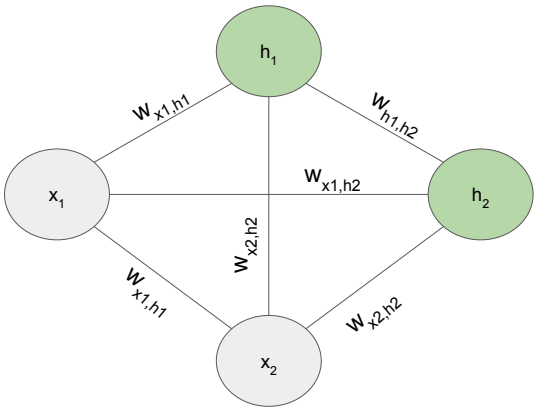


Figura 3.4: Arquitetura da Boltzman Machine tradicional.

Diferentemente da versão original, como pode ser observado na Figura 3.5, RBMs não possuem conexões entre neurônios da mesma camada, resultando em neurônios independentes entre si. É válido notar que os pesos sinápticos, representados por W, são assimétricos de maneira que os valores se mantêm quando propagados em ambas direções, entretanto os termos *Bias* são diferentes. Quando a rede propaga sinapses em direção a camada oculta utiliza-se o valor b, na direção oposta c (HINTON; OSINDERO; TEH, 2006).

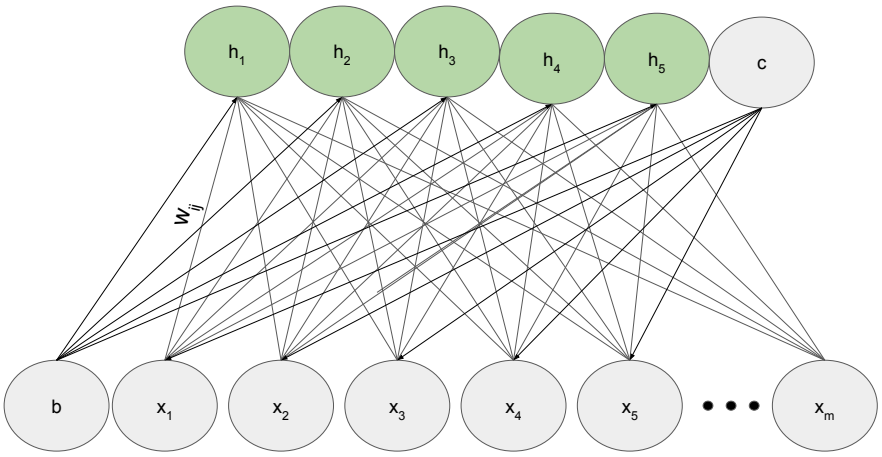


Figura 3.5: Arquitetura da Restricted Boltzmann Machine.

Esta arquitetura pode parecer limitadora quando comparada a BM original, entretanto segundo Freund (FREUND; HAUSSLER, 1994) RBMs podem representar qualquer distribuição discreta com uma quantidade suficiente de neurônios ocultos. Ainda, de acordo com Roux (ROUX; BENGIO, 2008), ao adicionar neurônios ocultos e considerando que seja possível, RBMs tendem a aperfeiçoar sua modelagem da distribuição.

Originalmente, a RBM foi proposta no trabalho de Smolensky (SMOLENSKY, 1986) com o nome de *Harmonioum*, entretanto se manteve inviável devido ao alto custo de treinamento do algoritmo proposto. Essa dificuldade foi superada no trabalho de Hinton (HINTON; OSINDERO; TEH, 2006) no qual um algoritmo de treino eficiente foi criado. Anteriormente a explicação do algoritmo, conceitos utilizados devem ser explicados.

Considerando que o valor de x (dados de entrada) sempre está disponível, utiliza-se a função de energia com base na probabilidade marginal (Equação 3.24).

$$P(x) = \sum_{h} \frac{e^{-Energia(x,h)}}{Z} \tag{3.24}$$

Ao assumir a independências entre os neurônios, a equação de energia passa a ser relativamente mais simples. Sua definição é apresentada na Equação 3.25, sendo b e c os valores de *Bias* e W os pesos assimétricos.

$$Energia(x,h) = -b - ch - hWx \tag{3.25}$$

Em seguida, o conceito Energia Livre (*Free Energy*) também é importado. Este é dado como a probabilidade logarítmica não normalizada da energia, e sua utilização favorece o cálculo do gradiente. A Equação 3.26 apresenta a energia livre, por sua vez a Equação 3.27 apresentam a função de probabilidade.

$$EnergiaLivre(x) = -bx - \sum_i log \sum_{h_i} e^{h_i W_i x} \tag{3.26}$$

$$P(x) = \frac{e^{-EnergiaLivre(x)}}{\sum_x e^{-EnergiaLivre(x)}} \tag{3.27}$$

Por fim, para obter o gradiente da função de probabilidade e consequentemente possibilitar o ajuste dos pesos, deriva-se o logaritmo da função de probabilidade (Equação 3.27). Sendo assim, adiciona-se o termo θ que representa os parâmetros da RNA, pesos sinápticos W e ambos *Bias* b e c. Em seguida, deriva-se o gradiente do logaritmo da função probabilidade com respeito a θ. Esta derivação é apresentada na Equação 3.28.

$$\begin{aligned} \frac{\partial logP(x)}{\partial \theta} &= -\frac{\partial EnergiaLivre(x)}{\partial \theta} + \frac{1}{Z}\sum_{\tilde{x}} e^{-EnergiaLivre(\tilde{x})} \frac{\partial EnergiaLivre(\tilde{x})}{\partial \theta} \\ &= -\frac{\partial EnergiaLivre(x)}{\partial \theta} + \sum_{\tilde{x}} P(\tilde{x}) \frac{\partial EnergiaLivre(\tilde{x})}{\partial \theta} \end{aligned} \tag{3.28}$$

Nota-se que a Equação 3.28 possui dois termos $-\frac{\partial EnergiaLivre(x)}{\partial \theta}$ e $\sum_{\tilde{x}} P(\tilde{x}) \frac{\partial EnergiaLivre(\tilde{x})}{\partial \theta}$. No trabalho de Hinton (HINTON; OSINDERO; TEH, 2006), os termos recebem as nomenclaturas de fase positiva e fase negativa. Estas não se referem a seus sinais. A fase positiva refere-se ao primeiro termo, e corresponde a uma amostragem proveniente do conjunto de dados (x). Por sua vez, a fase negativa, segundo termo, consiste em amostragens obtidas do modelo.

Para realizar a amostragem do modelo, necessária para o segundo termo, utiliza-se as Equações 3.29 e 3.30 juntamente com o conceito de amostragem de Gibbs (BENGIO et al., 2009). Esta é feita através da subamostragem sequencial de um conjunto de variáveis aleatórias x.

$$P(h_i = 1|x) = \frac{e^{c_i + W_i x}}{1 + e^{c_i + W_i x}} = sigmoide(c_i + W_i x) \tag{3.29}$$

$$P(x_i = 1|h) = \frac{e^{b_i + W_i h}}{1 + e^{b_i + W_i h}} = sigmoide(b_i + W_i h) \tag{3.30}$$

Esta subamostragem é realizada através da propagação em ambas direções na RNA. Inicialmente, utilizam-se dados de entrada (x) para alimentar a rede, em seguida as saídas são propagadas em direção a camada oculta, e de volta para a camada inicial utilizando as Equações 3.29 e 3.30. Esse processo é denominado como uma Cadeia ou Passo de Gibbs e, se repetida infinitas vezes, converge para o valor real da probabilidade $P(x)$. A Figura 3.6 demonstra o funcionamento da amostragem de Gibbs.

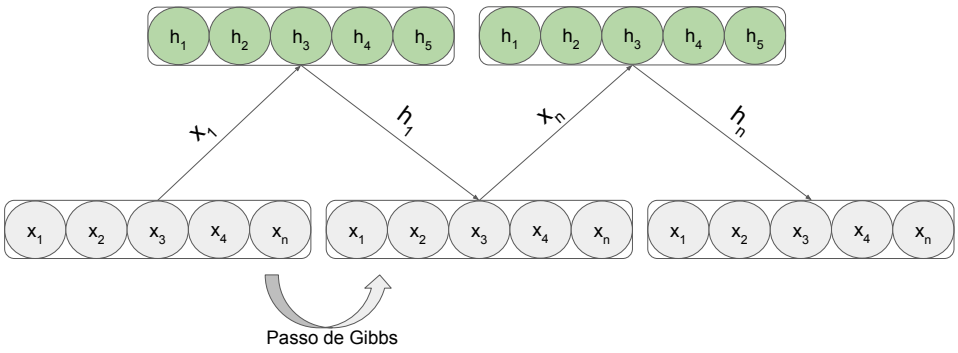


Figura 3.6: Amostragem de Gibbs. Cada passo corresponde a propagar as sinapses para "cima" e para "baixo".

Utilizando a amostragem de Gibbs e duas aproximações, Hinton propôs o algoritmo *Constractive Divergence*. A primeira aproximação consiste considerar a utilização somente de um exemplo por vez para o ajuste dos pesos. Essa resulta na alteração do segundo termo da Equação 3.28, produzindo a Equação 3.31.

$$\Delta\theta = -\frac{\partial EnergiaLivre(x)}{\partial\theta} + \frac{\partial EnergiaLivre(\tilde{x})}{\partial\theta} \tag{3.31}$$

Tomando como base o gradiente do logaritmo da função de probabilidade, o algoritmo emprega a Equação 3.28. O primeiro termo (fase positiva) é calculado diretamente através da Equação 3.29 com base nos dados de entrada x. O segundo termo (fase negativa) representa a amostragem obtido do modelo. Seu cálculo é viabilizado com base na segunda aproximação. Esta aplica k-Cadeias de Gibbs para obtenção da representação da RNA ($\tilde{x}$), sendo k a quantidade de amostragens realizados sequencialmente e $k >= 1$. Quanto maior o valor de k, mais

precisa será a aproximação, entretanto testes empíricos mostram que $k = 1$ fornece resultados satisfatórios.

O intuito do algoritmo consiste em treinar localmente ao redor dos dados de entrada x, reduzindo a energia livre no ponto exato x e alocando energia livre em pontos próximos à h, i.e., dados semelhantes à x produzirão h semelhantes. Ainda, o *Contrastive Divergence* também busca delinear o contraste entre exemplos do conjunto de dados x e representações produzidas pelo modelo h, ou seja, busca-se reduzir a energia nos dados de entrada (aumentar probabilidade) e aumentar-la para representações do modelo (reduzir probabilidade) (BENGIO et al., 2009).

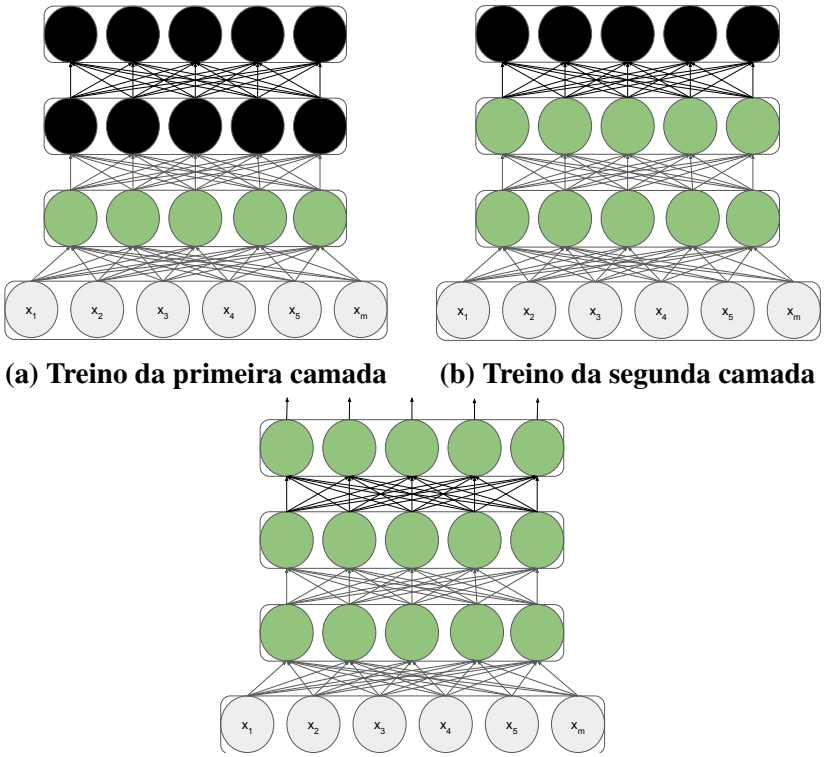
3.2.2 Deep Belief Networks

Considerando que treinar arquiteturas profundas utilizando somente algoritmos como o BP apresentam resultados sub-ótimos e inferiores a estruturas com poucas camadas, é necessário utilizar outros algoritmos de treinamento. Numa tentativa de construir redes profundas, Hinton (HINTON; OSINDERO; TEH, 2006) propõe a utilização da DBN (*Deep Belief Networks*).

DBNs são RNAs profundas utilizadas como inicialização de outras redes. Sua estrutura consiste em várias camadas empilhadas, em grande parte RBMs, treinadas de maneira gananciosa e não supervisionada.

O treinamento das DBNs é realizada em duas etapas. Na primeira, treina-se uma RBM utilizando diretamente os dados de entrada x. Em seguida, treina-se uma segunda RBM, porém, desta vez, utiliza-se a representação obtida ao alimentar a primeira RBM com x, para treinar a segunda RBM. Caso uma terceira RBM seja empilhada, esta seria treinada utilizando a representação obtida a partir da segunda RBM, que por sua vez utiliza a primeira RBM. Este processo se repete até que um número satisfatório de RBMs seja atingido.

Após empilhar várias RBMs, pode-se utilizar a DBN como inicialização dos pesos sinápticos de uma rede profunda. Para tanto, adiciona-se uma camada final referente as classes do problema. O treinamento desta última camada é realizado utilizando algoritmos como o BP, e proporciona um ajuste fino nos pesos da DBN. A Figura 3.7 mostra ambas etapas do treinamento da DBN.



(a) Treino da primeira camada **(b) Treino da segunda camada**

(c) Treino da terceira camada

Figura 3.7: Treinamento não supervisionado camada a camada de DBNs.

3.2.3 Auto-encoder

Inicialmente proposto no trabalho de Freund (FREUND; HAUSSLER, 1994), Auto-encoders (AE) são RNAs do tipo *Feed-Forward* compostas três camadas: entrada, oculta e saída inteiramente conectadas, e treinadas de maneira não supervisionada. Da mesma maneira que RBMs buscam obter uma representação do conjunto de dados, AEs são treinadas para obter uma codificação dos dados com maior acurácia possível (LAROCHELLE et al., 2009). Essa codificação consiste em empregar a camada oculta para obter uma representação abstrata, e em seguida converte-la de volta para os dados originais. Esse processo de transformação dos dados possibilita o aprendizado de características úteis e a filtragem de informação inútil (LIU et al., 2016).

Considerando que busca-se obter uma representação codificada dos dados, sua arquitetura necessariamente requer que a camada de saída possua a mesma dimensão (quantidade de neurônios) que o conjunto de dados. Por outro lado, a quantidade de neurônios da camada oculta não é fixa. Em caso de uma quantidade inferior, os neurônios da camada oculta tentam combinar atributos para obter uma representação de dimensão menor, no caso oposto atributos extras serão gerados (LIU et al., 2016). A Figura 3.8 apresenta a arquitetura de AEs. Nota-se

que os neurônios são conectados inteiramente utilizando pesos W. Além disso, cada neurônio também possui um termo *Bias* (b).

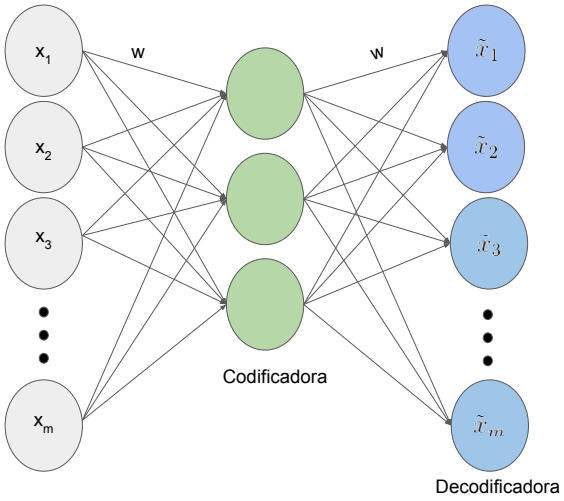


Figura 3.8: Arquitetura de um Autoencoder.

Uma notação formal é apresentada no trabalho de Vincent (VINCENT et al., 2008). Desta vez, a formulação objetiva a descoberta de pesos W e *Bias* b que possuam taxas de erro médio mínimas. Sendo x os dados originais, $\hat{x}$ os dados originais reconstruídos, L uma função de custo, e θ uma configuração da RNA, a Equação 3.32 apresenta esta formulação.

$$\theta = argmin\frac{1}{N}\sum_{i=1}^{n} L(x_n, \hat{x}_n) \tag{3.32}$$

Como função de custo (L), pode-se utilizar o Erro Quadrático Médio (Equação 3.10). Ainda, outras funções como *Cross-Entropy* e Erro Exponencial também podem ser aplicadas. Sendo y a saída obtida ($\upsilon(n)$), d_n a saída desejada (no caso d_n corresponde ao próprio x) e N a quantidade de exemplos do conjunto de dados, as Equações 3.33 e 3.34 apresentam ambas funções de custo. No caso do Erro Exponencial, utiliza-se um parâmetro normalizador τ.

$$CrossEntropy(n) = -\sum_{n}^{N}(y_n ln(d_n)) + (1 - d_n ln(1 - y_n)) \tag{3.33}$$

$$ErroExponencial(n) = \tau.exp(\frac{1}{\tau}\sum_{n}^{N}(y_n - d_n)^2) \tag{3.34}$$

Para o treinamento de AEs, inicialmente algoritmos como o *Back-Propagation* (Subseção 3.1.2) são utilizados. Entretanto, esta estratégia pode limitar a capacidade representativa da RNA, resultando em um modelo trivial que aprende a função identidade e generaliza de maneira

ineficiente.

Uma solução denominada *Denoising Auto-Encoder* (dAE) é proposta nos trabalhos (VINCENT et al., 2008, 2010). Esta consiste em corromper aleatoriamente os dados, de maneira que alguns exemplos passem a apresentar atributos com valores zerados. Esta alteração visa produzir um AE que aprende uma representação robusta a presença de ruído.

O treinamento do dAE utiliza os dados corrompidos ($\tilde{x}$) como entrada e compara os dados originais (x) com a reconstrução da rede y. É válido ressaltar que a corrupção parcial dos dados é realizada somente no treinamento, na fase de teste, os dados são fornecidos para a rede sem alteração. Desta maneira, A Figura 3.9 mostra a arquitetura do dAE.

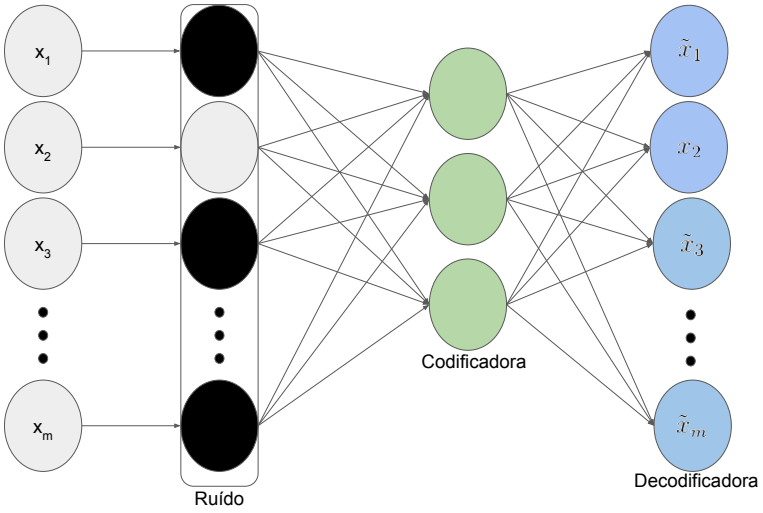


Figura 3.9: Arquitetura de um *Denoising Auto-Encoder*. Uma camada extra de ruído é adicionada.

3.2.4 Stacked Auto-encoders

De maneira similar a DBN, *Stacked Auto-Encoders* (SAEs) são construídos utilizando AE como blocos de construção. Suas fases de treino e teste também seguem o mesmo procedimento dos AEs. Inicialmente, camadas de AEs são treinadas de maneira gananciosa e utilizadas para o treinamento de AEs nos níveis seguintes, em seguida, utiliza-se a nova representação (codificação) obtida para o treinamento do segundo AE. Este procedimento se repete para todas as camadas.

SAEs são utilizadas como inicialização de uma rede profunda. Inicialmente, empilha-se AEs para obter uma representação mais discriminativa, e, por fim, para realizar a classificação, adiciona-se uma camada de saída treinada utilizando algoritmos como BP (RUMELHART; HINTON; WILLIAMS, 1986). A Figura 3.7 também se aplica a SAEs, no caso utiliza-se AEs no lugar

de RBMs.

Trabalhos com resultados empíricos como (LAROCHELLE et al., 2007) apontam que DBNs apresentam resultados levemente superiores, por outro também é válido ressaltar que SAEs apresentam erros de reconstrução com menor variância e não utilizam aproximações como a amostragem de Gibbs (BENGIO et al., 2009). Igualmente, *Denoising Auto-Encoders* também podem ser empilhados para a criação de *Stacked Denoising Auto-Encoders*. Segundo Vincent (VINCENT et al., 2010), utilizar DAEs ao contrário de AEs tradicionais possibilita o aprendizado de uma distribuição mais representativa, obtendo resultados superiores ou competitivos a DBNs.

Capítulo 4

CLASSIFICAÇÃO HIERÁRQUICA

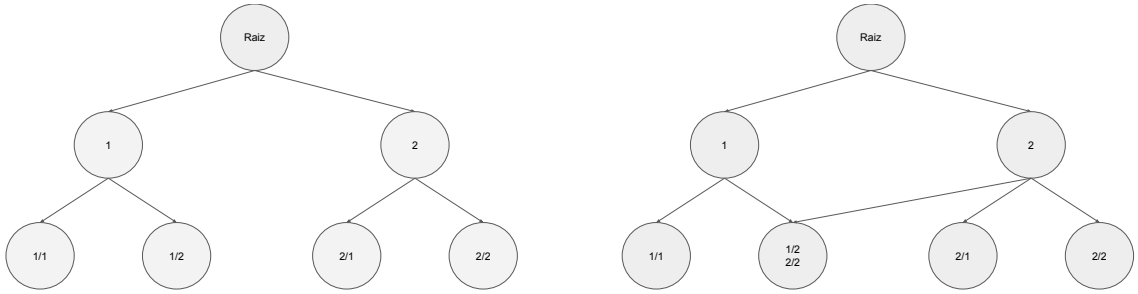
Este Capítulo fornece uma revisão sobre Classificação Hierárquica, disponibilizando uma descrição sobre cada abordagem presente na literatura assim como diferentes campos de aplicações e trabalhos recentes.

No contexto de Aprendizado de Máquina, dado um conjunto de dados **X**, a tarefa de classificação tradicional consiste em construir um classificador capaz de atribuir uma classe a uma instância. De maneira formal, a Definição 3 descreve a classificação.

Definição 3. *Classificação é a tarefa de aprender uma função f que mapeia cada vetor de valores $\boldsymbol{x}$ do conjunto de dados $\boldsymbol{X}$ para uma classe c pré-definida. A função f também é referida como modelo de classificação ou simplesmente classificador (TAN; STEINBACH; KUMAR, 2005).*

Na classificação tradicional, todas as classes $c \in C$ são mutuamente exclusivas, ou seja, para cada instância x é atribuída somente uma classe. Desta maneira, assume-se que não existem dependências entre as possíveis classes, tampouco uma estrutura que define relações entre classes similares. Entretanto, em problemas mais complexos, as classes são definidas por meio de relacionamentos entre superclasses e subclasses estabelecidas por uma hierarquia predefinida. Nestas situações, têm-se um problema de Classificação Hierárquica (CH). A Definição 4 formaliza o conceito.

Definição 4. *Sendo $\boldsymbol{X}$ o conjunto de dados, um problema de CH consiste em aprender uma função f capaz de classificar um instância $\boldsymbol{x} \in \boldsymbol{X}$ em um conjunto de classes $c \in C$, sendo C o conjunto de todas classes do problema. Ainda, a função f deve respeitar as restrições estabelecidas pela hierarquia do problema de maneira que, ao predizer um determinada classe, todas suas superclasses também são preditas (CERRI et al., 2016).*



(a) Hierarquia do tipo árvore. Os nós possuem somente um pai

(b) Hierarquia do tipo DAG. Os nós podem possuir mais de um pai (nó 1/1 2/1)

Figura 4.1: Exemplos de Hierarquias de Classes

Adicionalmente, problemas de CH são separados de acordo com suas peculiaridades. A Definição 5 apresenta os critérios de separação.

Definição 5. *Dado um problema de CH, a organização de suas classes é definida segundo três critérios* (γ, ψ, ϕ)*, cujos valores variam de acordo com* $\gamma \in \{Árvore, DAG\}$*,* $\psi \in \{HSC, HMC\}$*, e* $\phi \in \{Obrigatória\ em\ nó\ folha, Não\ obrigatória\}$ *(SILLA; FREITAS, 2010).*

Como especificado na Definição 5, em problemas hierárquicos, os nós de uma estrutura são utilizados para definir suas possíveis classes. Essa estrutura pode ser do tipo árvore ou DAG (Grafo Acíclico Direcionado). Uma árvore define uma estrutura cujas classes possuem somente uma superclasse (nó pai), enquanto estruturas DAG permitem que um nó possua múltiplas superclasses. A Figura 4.1 exemplifica ambas as possibilidades. Essa variação pode afetar consideravelmente o desempenho de alguns métodos, pois, ao permitir múltiplas superclasses, a estrutura DAG requer classificadores e estratégias mais complexas.

Além de definir as relações interclasses, as estruturas também definem a relações do tipo *Is-a*(restrição hierárquica). Tal relação estabelece que caso um exemplo pertença a um determinado nó, todos os seus nós pais também são inclusos. Tomando a Figura 4.1a como exemplo , supondo que o classificador prediga a classe 2.1, automaticamente a classe 2 faz parte da resposta. Intuitivamente, a mesma propriedade é válida para instâncias utilizadas na fase de treino. Desta maneira, uma classificação sempre consiste de um ou vários caminhos partindo da raiz até nós da hierarquia.

Problemas hierárquicos também são divididos de acordo com a quantidade classes que suas instâncias possuem (critério ψ da Definição 5). Em casos em que somente um caminho é permitido, tem-se um problema HSC (*Hierarchical Single-label Classification*), em caso de múltiplos caminhos, HMC (*Hierarchical Multi-label Classification*). Problemas do tipo HMC tendem a ser mais complexos visto que um classificador deve considerar vários caminhos. Por

sua vez, problemas HSC são mais simples, pois seus classificadores predizem somente um caminho.

O último critério da Definição 5, γ, estabelece quão profundo uma classificação deve ser. Caso $\gamma = \textit{Não obrigatória}$ nós internos podem ser a classificação mais profunda, caso $\gamma = \{\textit{Obrigatória em nó}$ somente nós folhas são classificações.

As abordagens usadas para CH são separadas em três grupos: *Flat*, Global e Local. As Seções 4.1, 4.2 e 4.2 descrevem o funcionamento das abordagens, com suas vantagens e desvantagens, assim como trabalhos que as utilizaram.

4.1 Abordagem *Flat*

Dentre as abordagens presentes na literatura, a *Flat* consiste na solução mais simples para se lidar com problemas hierárquicos. Ela opta por ignorar os aspectos hierárquicos do problema, transformando-o em um problema de classificação tradicional. A literatura diverge na definição da estratégia *Flat*. Alguns trabalhos como (SILLA; FREITAS, 2009b) a apresentam como vários classificadores binários um para cada possível classe da hierarquia, outros (ZIMEK et al., 2010; GHAZI; INKPEN; SZPAKOWICZ, 2010; BABBAR et al., 2013) utilizam somente um classificador para toda a hierarquia. Utilizar somente um classificador resulta em complexidade menor, entretanto este é responsável por distinguir mais classes, podendo resultar em resultados inferiores. Por outro lado, múltiplos classificadores binários são mais custosos, entretanto os classificadores precisam distinguir entre somente duas classes. De maneira geral, uma abordagem é dada como *Flat* se não leva em consideração nenhuma característica hierárquica.

Em ambas situações, classificadores tradicionais da literatura podem ser empregados diretamente. As Figuras 4.2 e 4.3 exemplificam o funcionamento da abordagem *Flat*. Na Figura 4.2 nós verdes circundados representam classificadores binários. Por sua vez, na Figura 4.3 um único classificador é treinado para distinguir entre os nós azul claro (classes).

Devido a sua simplicidade, a abordagem *Flat* é computacionalmente acessível. Entretanto, apresenta deficiências, pois características relacionadas as dependências entre as classes não são utilizadas. Além disso, devido a relação *Is-a* entre as classes, ao classificar uma instância, todas as classes referentes aos nós pais são adicionados a classificação final, podendo resultar em classificações totalmente errôneas. Em grande parte dos trabalhos, esta abordagem é empregada como comparador *baseline*.

Quanto às especificidades definidas anteriormente, esta abordagem consegue lidar com am-

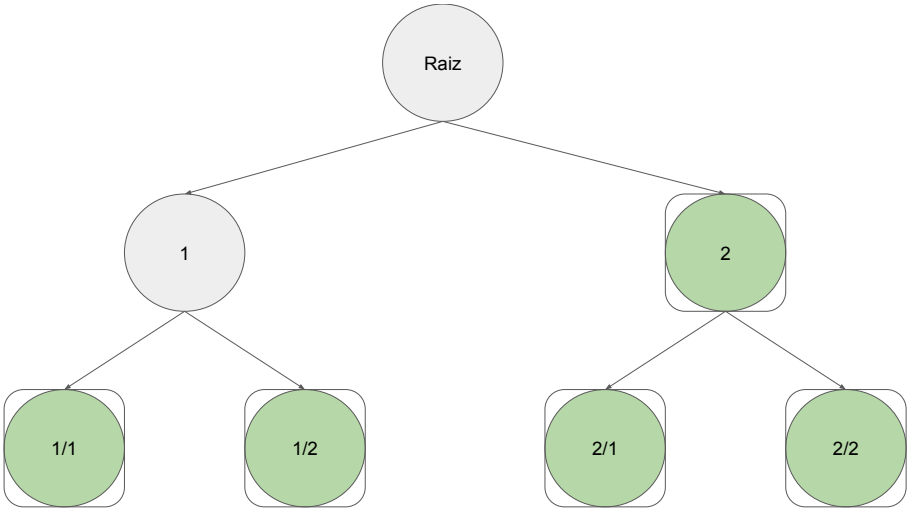


Figura 4.2: Funcionamento da abordagem Flat binária.

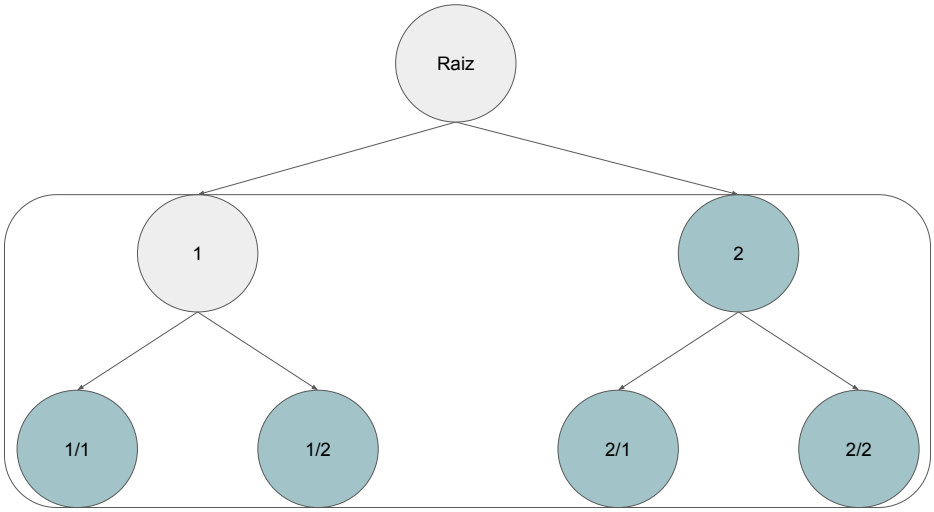


Figura 4.3: Funcionamento da abordagem Flat multi-classe

bas hierarquias, árvore e DAG, pois suas estruturas são ignoradas. Também consegue predizer nós internos, em razão de um nó interno ser tratado normalmente como uma classe. Por fim, no caso de problemas HMC, classificadores *multi-label* também podem ser aplicados.

Segundo Zimek (ZIMEK et al., 2010), essa abordagem também pode ser utilizada para verificar a qualidade dos atributos e as relações entre as classes. Essa análise é feita comparando os resultados da abordagem *Flat* com abordagens hierárquicas Local (Seção 4.2) e Global (Seção 4.3). Caso os atributos não reflitam a hierarquia, é muito provável que a abordagem *Flat* obtenha resultados competitivos ou melhores. Caso a hierarquia seja imprópria é muito provável que abordagens locais deteriorem os resultados. De maneira complementar, (BABBAR et al., 2013) afirma que a abordagem *Flat* é aconselhável quando o conjunto de dados é balanceado. A Seção 4.1.1 apresenta alguns trabalhos que empregaram a abordagem *Flat*.

4.1.1 Trabalhos Relacionados

O trabalho de Phachongkitphiphat (PHACHONGKITPHIPHAT; VATEEKUL, 2014) propôs uma alteração para melhorar resultados em problemas HMC cujas hierarquias são muito extensas. No primeiro nível da hierarquia, a abordagem local LCN (Subseção 4.2.1) é utilizada, caso um determinado nó não seja predito, todo os nós do seu ramo não são incluídos na resposta. Em seguida, completa o processo utilizando um classificador *Flat*.

No trabalho de Babbar (BABBAR et al., 2013), o comportamento da abordagem *Flat* é avaliado também em hierarquias extensas. Nesse caso, a grande variedade dos dados e o tamanho da hierarquia dificultam demasiadamente o processo de classificação. Em virtude desse fato, os autores propuseram um método de poda (eliminação de nós) na hierarquia com base em *Meta-Learning*. Cada meta classificador leva em consideração as relações entre nós irmãos e filhos para determinar se a remoção de um nó pode levar ou não a resultados superiores.

No contexto de classificações de emoções a partir de textos, o trabalho de Ghazi (GHAZI; INKPEN; SZPAKOWICZ, 2010) investiga a aplicação das abordagens *Flat* e Local. Ao propor e reestruturar seu problema usando uma hierarquia, os autores mostram que ao explorar as relações entre superclasses e subclasses, os resultados são favorecidos.

4.2 Abordagem Local

Diferentemente, a abordagem Local utiliza múltiplos classificadores capazes de distinguir uma quantidade limitada de classes. De maneira geral, classificadores locais podem ser vistos

como casos de reduções de AM, pois reduzem o problema de classificação para vários problemas menores cujas soluções são combinadas para solucionar o problema maior (CERRI et al., 2016).

Apesar de todas estratégias da abordagem Local reduzirem o problema de classificação, suas fases de treino e teste diferem consideravelmente. Segundo a revisão de Silla (SILLA; FREITAS, 2010), a literatura oferece três variações

- LCN - Classificador Local por Nó (DUMAIS; CHEN, 2000);
- LCPN - Classificador Local por nó Pai (KIRITCHENKO; MATWIN; FAMILI, 2005);
- LCL - Classificador Local por Nível (CERRI; BARROS; CARVALHO, 2015).

Como vantagem, por explorar diferentes técnicas para o treino dos seus classificadores e as relações entre as classes, as três estratégias tendem a obter resultados superiores a abordagem *Flat*. Ainda, os benefícios de qualquer classificador da literatura podem ser incorporadas ao modelo. Entretanto, devido à quantidade de classificadores, sua complexidade computacional é mais elevada.

As Subseções 4.2.1, 4.2.2 e 4.2.3 apresentam uma descrição mais detalhada sobre cada estratégia da abordagem Local, assim como trabalhos que as utilizaram.

4.2.1 Classificador Local por Nó

A estratégia Classificador Local por Nó (LPN) propõe a utilização de um classificador binário por nó da hierarquia. Essa redução tem como objetivo reduzir a complexidade dos classificadores de maneira que somente aspectos relacionados a um nó sejam considerados. A Figura 4.4 exemplifica o comportamento dessa estratégia.

Em sua fase de treino, um classificador binário é associado e treinado para cada nó da hierarquia, com exceção do nó raiz. Na fase de teste, todos os classificadores são utilizados para obtenção da classificação final. Desta maneira, ambas etapas podem ser paralelizadas.

Por utilizar classificadores binários, cada nó da hierarquia recebe uma classificação, verdadeira ou falsa (0 ou 1). Sendo assim, inconsistências são possíveis. Supondo a hierarquia da Figura 4.4, um teste pode predizer os nós $\{1, 2.1\}$ como classificação, pode-se notar que esta não é válida, pois os nós não são conexos na hierarquia. Desta maneira, é necessário realizar correções de inconsistências.

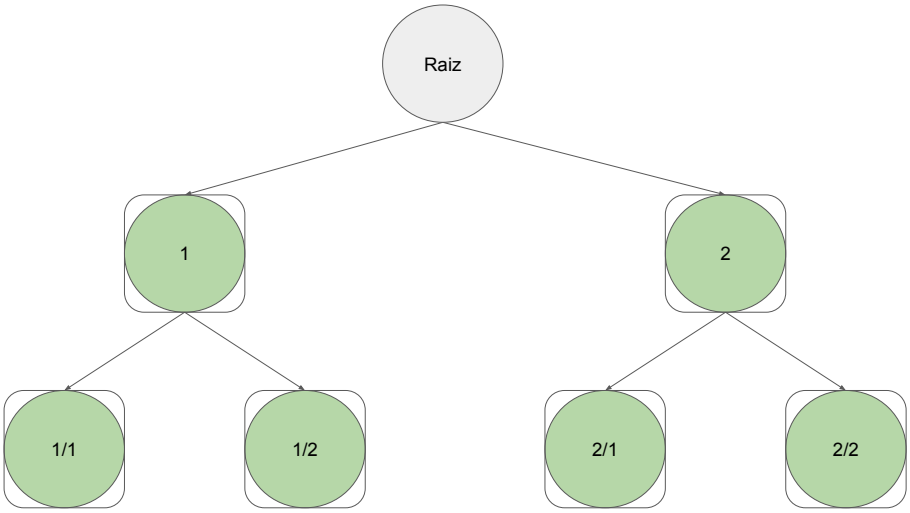


Figura 4.4: Funcionamento da estratégia LCN. Um classificador binário é treinado para cada nó circundado (Nós 1,2,1.1,1.2,2.1 e 2.2)

Uma vantagem desta estratégia reside na sua indiferença quanto aos tipos de hierarquias. Entretanto, como desvantagem, muitos classificadores são necessários, fato que afeta sua complexidade computacional. A Seção 4.2.1.1 apresenta alguns trabalhos que utilizaram esta estratégia.

4.2.1.1 Trabalhos Relacionados

O trabalho de Dumais (DUMAIS; CHEN, 2000) faz um comparativo entre as abordagens *Flat* e LCN na classificação de páginas *Web*. Neste caso, os autores utilizaram SVMs e a estratégia LCN apresentou resultados superiores a *Flat*.

O trabalho de Cesa-Bianchi (CESA-BIANCHI; RE; VALENTINI, 2011) explora a sinergia entre *emsembles*, métodos de custo e *data-fusion* em problemas HMC. Seus resultados são interessantes, pois reforçam a necessidade de vários tipos de técnicas. Ao utilizar diversos classificadores (*Ensemble*), treinados com diversos conjunto de dados (*data-fusion*) e métodos de custo para decidir qual nó expandir, a precisão em diferentes níveis da hierarquia tende a não decair.

No trabalho de Ying (YING; YING, 2011), é proposta uma estratégia utilizando um nó de retorno que representa possíveis erros em níveis superiores. Usando LCN, esta abordagem adiciona um classificação extra (retorno) para cada classificador que não está no primeiro nível. Na fase de teste, caso uma instância seja classificada na classe de retorno, assume-se que o classificador do nível superior errou, e a classificação é realizada novamente no nível superior, porém a classe predita anteriormente (errada) é removida. Essa abordagem apresentou resultados satisfatórios, porém os autores não deixaram claro o comportamento da estratégia quando

todos os caminhos possíveis atingem um nó de retorno.

No trabalho de Zhang (ZHANG et al., 2016), uma hierarquia de classificadores é proposta para a classificação de emoções. O conjunto de dados utilizado é composto por postagens oriundas de micro-*blogs* chineses cujo conteúdo é esparso.

4.2.2 Classificador Local por Nó Pai

Assim como o nome sugere, essa estratégia reduz o problema de classificação original para vários problemas locais no qual, para cada nó pai, um classificador é construído. Usando a hierarquia da Figura 4.5, em sua fase de treino, um classificador é treinado para os nós "Raiz, 1 e 2". Nota-se que um nó extra é adicionado como "Raiz" da hierarquia. Os classificadores treinados são capazes de distinguir apenas entre seus nós filhos. Desta maneira, cada classificador é responsável por diferenciar menos classes, e consequentemente, tende a ser mais eficiente.

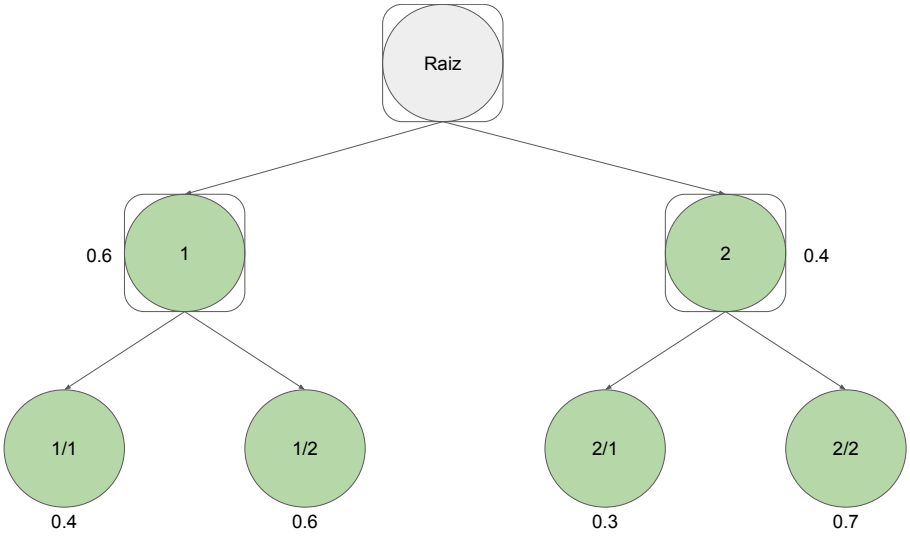


Figura 4.5: Funcionamento da estratégia LCPN. Neste caso, um classificador é treinado para os nós internos (Raiz, 1 e 2)

Na fase de teste, uma abordagem *Top-Down* é utilizada. A partir do classificador do nó "Raiz", cada instância é classificada entre seus nós filhos. A classe obtida para o nó em questão, é expandida, de maneira que a instância é testada novamente usando o classificador associado ao nó predito. Esse processo se repete até que um nó folha seja atingido. Consequentemente, a classificação final é composta da concatenação das classificações obtidas da raiz até uma folha.

Tomando a Figura 4.5 como exemplo e as probabilidades preditas por seus classificadores, o exemplo testado seria classificado como 1/2, pois, o classificador do nó "Raiz" obteve a classificação 1 (60% de probabilidade) e em seguida, o classificador do nó "1" classificou a instância como 2 (60% de probabilidade).

Esta estratégia de teste apresenta vantagens e desvantagens. Como benefício, inconsistências de classificação não são possíveis, pois a classificação de uma instância sempre corresponde a um caminho. Como inconveniência, erros provenientes de camadas superiores são propagados para níveis mais profundos (NAKANO et al., 2017a).

Também é válido notar que a estratégia de treino permite a paralelização do treino dos classificadores. Entretanto, na fase de teste, esta paralelização não é possível, visto que o classificador a ser utilizado depende diretamente da classificação oriunda do nível anterior. Ainda, classificações *Não obrigatória* não são possíveis, tornando necessário o uso de estratégias para reduzir a profundidade das classificações.

A Subseção 4.2.2.1 apresenta trabalhos que aplicam a estratégia LCPN, juntamente com técnicas de decisão para definir qual nó expandir, visando diminuir a propagação de erros.

4.2.2.1 Trabalhos Relacionados

Uma técnica denominada *Selective Top-Down* é proposta no trabalho de Secker (SECKER et al., 2007). Para o treinamento de cada classificador, vários tipos de classificadores são testados, aquele que apresentar maior desempenho em um conjunto de validação é selecionado. Apesar de obter melhores resultados que a estratégia *Top-Down* clássica, erros ainda são propagados.

Uma extensão do trabalho de Secker (SECKER et al., 2007) é proposta em (SECKER et al., 2010). Desta vez, além dos classificadores, atributos também são selecionados. Ao contrários das técnicas de seleção de atributos tradicionais que são aplicadas somente uma vez no conjunto de dados, a estratégia proposta aplica separadamente a seleção de atributos para cada classificador da hierarquia, resultando em diferentes atributos para cada classificador. Um trabalho muito semelhante a este, porém em outra aplicação, é apresentado por Silla (JR; FREITAS, 2011).

O trabalho de Wang (WANG; ZHAO; LU, 2014) também investiga a aplicação de técnicas de *Meta-Learning* para minimização do erro. Essa técnica treina um meta-classificador para definir qual nó deve ser expandido em cada nível da hierarquia. O meta-classificador utiliza os atributos do conjunto de dados, juntamente com vários meta-atributos, como probabilidade média e mínima de um nó, para sua construção.

O trabalho de Zhu (ZHU; WEI; NGO, 2014) emprega as probabilidades dos nós filhos e netos para definir qual o nó a ser expandido. Esta técnica foi desenvolvida exclusivamente para problemas HSC, e visa descobrir uma matriz de pesos responsável por auxiliar na decisão do nó a ser expandido utilizando probabilidades de predições.

O trabalho de Hernandez (HERNANDEZ; SUCAR; MORALES, 2013) propõe três técnicas si-

milares que usam todos os classificadores. A primeira consiste em ordenar as probabilidades de todos os nós e selecionar a classificação consistente com maior probabilidade. Esta técnica é simples, porém é afetada por nós com poucos filhos, pois suas probabilidades são altas. A segunda emprega a multiplicação das probabilidades segundo um ramo da hierarquia, podendo sofrer com erros numéricos devido à multiplicação de valores pequenos. A terceira consiste na média das probabilidades, mostrando-se a melhor das três, devido a simplicidade e eficiência. Todas estas técnicas são específicas para problemas cuja hierarquia é uma árvore e permitem somente uma classe por nível.

Hernandez (HERNáNDEZ; SUCAR; MORALES, 2014) também propõem a utilização de todos nós classificadores para obtenção da classificação final. A escolha da classificação final reside na multiplicação direta das probabilidades, o ramo com maior probabilidade é escolhido como classificação final. Esta técnica é específica para problemas hierárquicos HSC e consegue lidar com ambos tipos de hierarquia. Ainda, baseia-se no teorema de Bayes, pois busca-se obter a maior probabilidade conjunta, entretanto ao reduzir a decisão para uma simples multiplicação, assume-se independência condicional entre superclasses e subclasses.

O trabalho de Naik (NAIK; RANGWALA, 2016) emprega uma redução na hierarquia por meio de eliminação de nós. Com base em um limiar calculado automaticamente, os autores implementam duas técnicas, uma local e uma global, ambas utilizando a média das probabilidades das classes multiplicado pelo seu desvio padrão, e um parâmetro de normalização. Entretanto, a local considera somente os nós de um nível, enquanto a global utiliza todos os nós da hierarquia. Consequentemente, a local achata níveis inteiros da hierarquia, e a global, por sua vez, elimina nós específicos. Seus resultados mostram que a estratégia global obteve melhor desempenho.

4.2.3 Classificador Local por Nível

Com base nos níveis (alturas) da hierarquia, esta estratégia reduz o problema para várias classificações menores nos quais um classificador é treinado para cada nível. Sendo assim, quando comparada as outras estratégias locais, menos classificadores são necessários.

Em relação aos tipos de hierarquia, esta estratégia apresenta um viés que pode ser problemático caso o problema possua um DAG como hierarquia, pois um nó pode possuir mais um de uma superclasse, e, consequentemente, mais de uma altura. No trabalho de (CERRI et al., 2016), os autores assumem a maior altura como definitiva. Tal problema não ocorre quando a estrutura é uma árvore.

A fase de treino desta estratégia constrói um classificador local para cada nível da árvore.

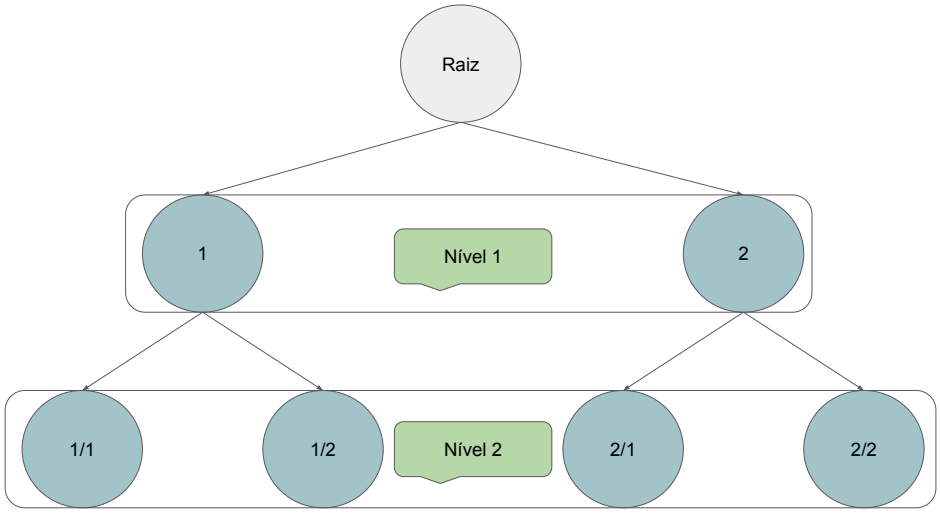


Figura 4.6: Funcionamento da estratégia LCL. Neste caso, um classificador para o nível 1 e outro para o nível 2

Como pode ser observado na hierarquia da Figura 4.6, dois classificadores são necessários. No nível 1, o classificador distingue entre as classes 1 e 2. No nível 2, o classificador distingue entre as classes 1/1, 1/2, 2/1 e 2/2.

Na fase de teste, a instância é classificada por todos os classificadores, consequentemente, para cada nível, uma classificação é obtida. Sendo assim, existe a possibilidade paralelismo, tanto na fase de treino como teste.

Após a fase de teste, classificações inconsistentes são possíveis. Tomando a Figura 4.7 como exemplo, nota-se que os nós obtidos como resposta não são conexos (não formam um caminho), portanto nessas situações uma fase de correção pós-testes é necessária.

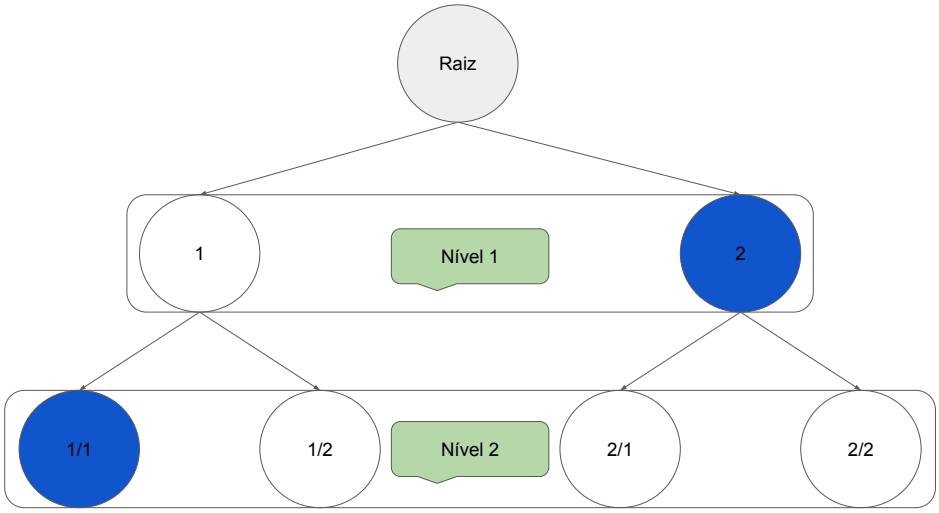


Figura 4.7: Predições da estratégia LCL com inconsistência

Na subseção a seguir, métodos de correção de inconsistência, assim como trabalhos relaci-

onados são apresentados.

4.2.4 Trabalhos Relacionados

Paes et. al. (PAES; PLASTINO; FREITAS, 2012) propuseram uma correção com base nas probabilidades preditas pelos classificadores. Essa técnica é denominada como SWV (*Sum of Weighted Votes*) e utiliza a soma das probabilidades de todos nós classificados da hierarquia. O ramo que contém a maior soma é mantido como resposta. Por fim, caso uma classificação *Obrigatória em nó folha* seja necessária, classificadores por nó pai são utilizados para completar a classificação até um nó folha. Apesar de ser mais complexa, esta técnica busca corrigir erros em níveis superiores.

Cerri (CERRI; BARROS; CARVALHO, 2013) aplicaram um método de correção similar à abordagem por nó pai. No primeiro nível, esta estratégia busca qual classe obteve maior probabilidade e expande o nó correspondente. Nos níveis subsequentes, os nós com maior probabilidade e que são conexos com os escolhidos no nível diretamente superior são explorados. Este processo se repete até que um nó folha seja atingido. Da mesma maneira que a LCPN, esta técnica de correção tende a propagar erros nas camadas superiores. Por questões de referência, este método é denominado Correção LCPN.

No trabalho de Cerri (CERRI; BARROS; CARVALHO, 2015) utilizou-se um correção simples denominada *Simple-Prune*. Seguindo um método *Top-Down*, esta técnica elimina todas as classificações cujos nós pais não foram preditos. Seus resultados são competitivos, entretanto tendem a propagar erros de níveis superiores.

Cerri (CERRI; BARROS; CARVALHO, 2014) utiliza redes neurais *Multi-Layer Perceptron* para classificar funções de proteínas. Esse problema é do tipo HMC e possui uma hierarquia no formato de árvore. Nesse trabalho, os autores propõem a utilização das probabilidades preditas pelas redes neurais de níveis superiores como atributos extras para as redes do nível seguinte. Esse método traz resultados positivos, entretanto não permite a paralelização em ambas fases de teste e treino. Seu método de correção de inconsistência é o *Simple-Prune*.

No trabalho de Cerri (CERRI; BARROS; CARVALHO, 2013), redes neurais *Multi-Layer Perceptron* são utilizadas na classificação de GPCRs. Este problema é do tipo HSC, possui uma árvore como hierarquia e o método de correção utilizado foi o LCPN. Seus resultados não foram competitivos com as abordagens comparadas. Os autores relatam que o motivo dos resultados inferiores reside no conjunto de dados muito desbalanceado, principalmente nos níveis mais profundos da hierarquia.

O trabalho de Cerri (CERRI; BARROS; CARVALHO, 2015) estuda a aplicação de redes neurais *Multi-Layer Perceptron* juntamente com a hierarquia proveniente da GO (*Gene Ontology* [1]) para a classificação de funções de proteínas. O problema em questão possui várias classes por nível (HMC), um DAG como hierarquia e o método de correção *Simple-Prune*. Os autores exploram a utilização dos rótulos originais das instâncias de treino como características extras para as redes de camadas mais profundas. Na fase de teste, as predições são adicionadas aos vetores de características. Através dos resultados obtidos, os autores mostram que, ao forçar a dependência entre rótulos, as redes tendem a produzir resultados superiores. Também é válido mencionar que, por ser um DAG, um nó pode ter diferentes níveis, consequentemente ajustes são necessários. Nesse trabalho, os autores optaram por atribuir o nível mais profundo.

O problema de classificação de funções de genes do fungo levedo (Saccharomyces cerevisia) é novamente pesquisado em (CERRI et al., 2016). Este trabalho estende as pesquisas de Cerri (CERRI; BARROS; CARVALHO, 2014, 2013), além de usar as probabilidades e os rótulos como atributos extras, somente as probabilidades como atributos também são avaliadas. Seus resultados mostram que somente probabilidades fornecem resultados inferiores, enquanto, forçar as dependências entre classes favorece os resultados. Este trabalho lidou com um problema HMC, estrutura árvore e correção do tipo *Simple-Prune*.

4.3 Abordagem Global

De maneira semelhante a abordagem *Flat*, a abordagem Global visa a construção de somente um classificador, todavia aspectos relacionados à hierarquia são levados em conta na fase de treino do modelo. Sendo assim, os classificadores globais tendem a ser mais complexos no aspecto computacional.

Segundo a literatura, vários algoritmos são usados como base para as abordagens globais. De maneira geral, pode-se sumarizá-los em dois grupos: Adaptação de Algoritmos ou Extração de Regras. Ambos geram modelos, com respeito as limitações dos problema capazes de classificar instâncias em qualquer nó da hierarquia. Assim como seu nome sugere, trabalhos que utilizaram Adaptação de Algoritmo buscam utilizar algoritmos da literatura e adaptá-los para lidar com problemas de Classificação Hierárquica. Por sua vez, a Extração de Regras engloba trabalhos que buscam realizar a classificação através de regras do tipo Se-Então.

Na próxima subseção 4.3.1 trabalhos recentes que utilizaram a abordagem global são apresentados. Inicialmente, abordagens de Adaptação de Algoritmos são apresentado. Em seguida,

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

abordagens de Extração de Regras são sumarizadas.

4.3.1 Trabalhos Relacionados

Incorporando o conceito de PCT (*Predictive Clustering Trees*) ao contexto de CH, o trabalho de Vens (VENS et al., 2008) propõe três técnicas diferentes. PCTs utilizam árvores de decisões como uma hierarquia de *clusters* cujos nós são separados recursivamente em *clusters* menores, a construção de PCTs visa minimizar a variância intra-*cluster*. Sendo assim, cada técnica proposta neste trabalho constrói árvores de decisão com base em PCTs. A primeira técnica, Clus-HMC, constrói uma árvore capaz de predizer qualquer classe da hierarquia. A segunda, Clus-SC, transforma o problema em várias classificações binárias, ou seja, uma árvore é construída para cada classe. A terceira, Clus-HSC, utiliza uma estratégia semelhante à LCN para a construção de uma árvore por aresta da hierarquia. No geral, a técnica Clus-HMC obteve resultados superiores, enquanto as outras duas se mantiveram competitivas.

Alguns trabalhos adaptam o classificador tradicional NB (*Naive Bayes*) para CH. A primeira adaptação foi proposta no trabalho de Silla (JR; FREITAS, 2009). Este constrói um NB global, denominado GMNB (*Global Model Naive Bayes*), capaz de predizer qualquer nó da hierarquia. Para a fase de teste, o GMNB usa o processo de classificação do NB tradicional, porém leva em consideração a hierarquia do problema, i.e. caso uma instância possua classe 2.1, ela contribui para o cálculo de probabilidade das classe 2 e 2.1. Este trabalho é estendido no trabalho de Merschmann (MERSCHMANN; FREITAS, 2013), desta vez a estratégia LCN é utilizada como base para a construção do classificador global com nome ELHNB (*Extended Local Hierarchical Naive Bayes*). Na fase de treino, para cada nó da hierarquia, um classificador NB binário é utilizado. Na fase de teste, a nova instância é testada por todos os classificadores e para cada caminho possível na hierarquia a média geométrica é calculada. O caminho até um nó com a maior média geométrica é dado como classificação. Seus resultados se mostraram superiores ao GMNB. Uma adaptação do ELHNB é apresentada por Fabris (FABRIS; FREITAS, 2014) com complexidade computacional consideravelmente menor. Seus resultados se mantiveram competitivos com o ELHNB tradicional.

No trabalho de Borges (BORGES; NIEVOLA, 2012), uma rede neural competitiva com duas camadas é utilizada como classificador global para problemas HMC. Na primeira camada, os atributos de entrada são processados e passados para segunda camada. Esta, construída a partir da hierarquia, é responsável por selecionar um conjunto de neurônios vencedores (classificações). Os neurônios pais dos neurônio vencedores são excitados e seus pesos ajustados.

No trabalho de Barros (BARROS et al., 2013), uma técnica com base em Classificação por

Centróides é desenvolvida. Este tipo de classificação toma como base algoritmos de *Clustering* (Agrupamento) e os utiliza como referência, a classe do centroide mais próximo de uma nova instância é dado como sua classificação. Neste trabalho, para cada nó da hierarquia, o algoritmo de agrupamento EM (*Expectation Maximization*) é utilizado. Para o cálculo das coordenadas do centroide, duas técnicas são propostas. A primeira toma a média de suas instâncias. A segunda utiliza a média das instâncias cuja probabilidade de pertencer ao centroide supera um determinado limiar.

No trabalho de Ferrandin (FERRANDIN et al., 2015), novamente uma técnica com base em Classificação por Centroides é utilizada. Para cada nó da hierarquia, um centroide é construído. Para sua construção, todos as instâncias são transformadas em representações *tf-idfs* (*frequency–inverse document frequency*). Em seguida, a média de todas as instâncias pertencentes a um nó, levando em conta as relações estabelecidas pela hierarquia, é tomada como coordenadas do centroide. Uma segunda técnica também é proposta, esta incorpora a frequência de atributos de uma mesma classe como peso extra na representação *tf-idf*.

Uma grande quantidade de trabalhos propõem abordagens globais que geram regras de classificação do tipo ”Se-Então” segundo algum critério de otimização. Nestes trabalhos, tais regras são criadas sequencialmente até que uma quantidade satisfatória de exemplos sejam cobertas. Até o momento, a literatura oferece uma variedade de abordagens, por exemplo, os trabalhos de Cerri utilizam Algoritmos Genéticos (CERRI; BARROS; CARVALHO, 2012, 2013) e Evolução Gramatical (CERRI; BARROS; CARVALHO, 2015). De maneira similar, Otero (OTERO; FREITAS; JOHNSON, 2010) e Alves (ALVES; DELGADO; FREITAS, 2010) empregam Otimização por Colônia de Formigas e Sistemas Imunológicos Artificiais respectivamente. Usando uma abordagem diferente, Ferrandin (FERRANDIN; ROMAO, 2016) utilizou o algoritmo *FP-Growth* como base para extrair regras de acordo com padrões emergentes.

Capítulo 5

MÉTODOS PROPOSTOS

Este capítulo contém a motivação por traz das novas estratégias locais propostas, ressaltando sua importância na classificação de TEs e problemas similares. Da mesma maneira, também apontamos o motivo da utilização de Deep Learning neste contexto.

Como ressaltado no Capítulo anterior, algumas estratégias da abordagem Local não são capazes de predizerem classificações *Não obrigatória* automaticamente. Esta particularidade é uma limitação nesta pesquisa, pois segundo a Definição 4, a classificação de TEs consiste em um problema do tipo {Árvore,HSC,*Não obrigatória*}.

De maneira geral, os dois primeiros critérios não são muito desafiadores, uma vez que problemas do tipo Árvore e HSC são relativamente mais simples que suas versões alternativas, DAG e HMC respectivamente. A dificuldade reside no último critério, *Não obrigatória*, no qual a estratégia LCPN não possui método automático para este cenário em específico.

Considerando os métodos de correção mencionadas na Seção 4.2.4, ambos *Simple-Prune* e *Sum of Weighted Votes* são capazes de predizer classificações *Não obrigatória*. O primeiro simplesmente elimina classes cuja superclasses não foram preditas, possivelmente reduzindo a classificação final para um nó interno. Por sua vez, o método SWV pode selecionar um caminho até um nó interno caso a sua probabilidade média seja superior a de outros caminhos.

Todavia este não é o caso da estratégia LCN, pois normalmente seleciona-se recursivamente o nó cujo classificador prediz com a maior probabilidade até que um nó folha seja atingido, de maneira idêntica LCPN. Intuitivamente, o mesmo aplica-se a estratégia LCPN, em ambas situações uma classificação *Obrigatória em nó* folha é sempre obtida.

Uma maneira de possibilitar classificações *Não obrigatória* consiste em aplicar *thresholds* (limiares) para cada nó (CECI; MALERBA, 2007). Assim, toda classificação cuja probabilidade

é menor que o limiar estipulado é eliminada da classificação final, todavia este tipo de ajuste pode levar ao problema do bloqueio. Este ocorre quando o limiar estipulado elimina todas as classificações a partir de um determinado nó, resultando em classificações precoces que não atingem níveis satisfatórios na hierarquia (SUN et al., 2004).

Utilizando a estratégia LCN, três maneiras de lidar com o problema do bloqueio são discutidas no trabalho de (SUN et al., 2004). O primeiro método, já mencionado no parágrafo anterior, *Threshold Reduction Method* (Método de Redução por Limiar) consiste em simplesmente eliminar toda classificação cuja probabilidade é menor que um determinado *Threshold*. Intuitivamente, a eliminação tem início a partir dos nós folhas, e finaliza quando alguma probabilidade é maior que o limiar. A limitação deste método, consiste na necessidade de determinar um limiar manualmente. O segundo método, *Restricted Voting Method* (Método de Voto Restrito), utiliza um segundo classificador para cada nó, conectando um nó com seu nó "neto". Caso a probabilidade do classificador original bloqueie a classificação, a probabilidade do classificador extra é utilizada para comparação. Uma classificação é eliminada somente se ambas probabilidades não atinjam o limiar especificado, novamente é necessário um valor manual. O terceiro método, *Extended Multiplicative Method* (Método de Multiplicação Estendida), leva em conta as probabilidades dos nós pais e do nó em questão. Ambas são multiplicadas, e caso o valor resultante supere um determinado limiar, a classificação é mantida, caso contrário é eliminada e o processo continua para nós superiores. Neste caso, a sua maior limitação consiste em estabilidade numérica, uma vez que os valores comparados tendem a ser muito próximos a 0.

Além disso, caso um especialista estabelecesse um limiar para esta pesquisa, esta ainda seria problemática, pois não seria possível usar a estratégia LCPN. Em algumas hierarquias, como a mostrada na Figura 5.1 (nó "2"), e também as do foco desta pesquisa (Figuras 6.1 e 6.2 (nó "2.1")), não existe um limiar possível, caso a classe "2" seja de interesse no problema, uma vez que existe somente uma classe para o classificador, e consequentemente sua probabilidade é 100%, todavia um limiar neste valor eliminaria todas as outras predições. Desta forma, esses três métodos não foram investigados nesta pesquisa.

Em duas pesquisas mais recentes que utilizaram a estratégia LCPN, dois métodos foram propostos. No trabalho de Hernandez (HERNáNDEZ; SUCAR; MORALES, 2014), uma redução com base em entropia é aplicada. Assumindo um método *bottom-up*, para cada nó a entropia é calculada. Caso seu valor seja menor que zero a classificação é desconsiderada, e o nó do nível superior é analisado. Caso o primeiro nível da hierarquia seja atingido ou uma entropia maior que zero seja encontrada, o processo se encerra e a classificação é reduzida ao nó encontrado. Em um trabalho subsequente, Ramizes-Corona (RAMíREZ-CORONA; SUCAR; MORALES, 2016)

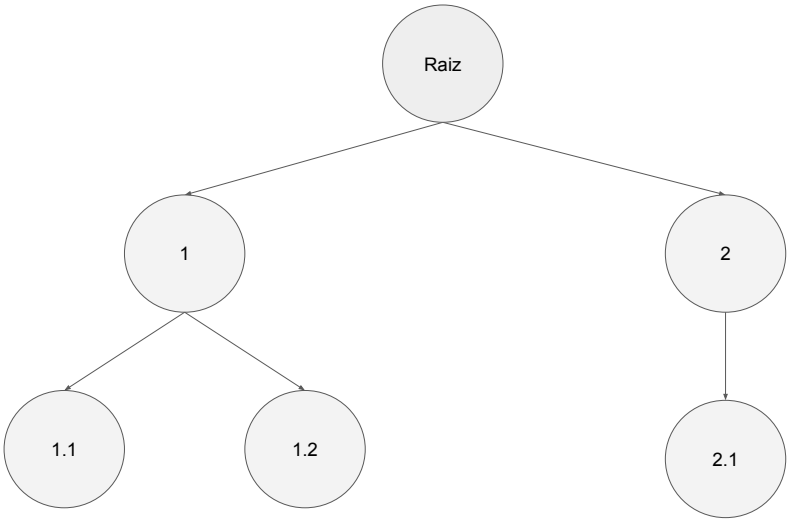


Figura 5.1: Hierarquia com nós interno com somente um nó filho

propôs o emprego de uma poda *top-down* com base nas probabilidades dos seus nós filhos. Antes de realizar a poda, as probabilidades de todos nós são obtidas. Nesse caso, a probabilidade de um nó filho corresponde a multiplicação de seus nós ancestrais, i.e. todos os seus nós superiores (nós pais) até a raiz. Em seguida, verifica-se se a probabilidade do nó mais provável é maior que a probabilidade dos seus nós irmãos. Caso sim, o método continua buscando nós mais profundos para poda. Do contrário, a poda é realizada.

Apesar de não dependerem de um limiar pré-definido, estes métodos apresentaram uma deficiência relevante para esta pesquisa. Em ambas situação, os autores concluíram que seus métodos são adequados para hierarquias profundas, com dez ou mais níveis. No caso de hierarquias pequenas, três ou quatro níveis, esses métodos não se mostraram satisfatórios, chegando ao ponto em que a abordagem *Flat* se mostrou superior. Similarmente aos conjuntos de dados utilizados nesses trabalhos , Elementos Transponíveis também possuem quatro níveis. Sendo assim, a utilização desses métodos foi desmotivada.

Levando em consideração as limitações acima apresentadas, foram propostas duas novas estratégias locais chamadas nLLCPN (Classificador Local por Nó Pai *Não obrigatória*) e LCPNB (Classificador Local por Nó Pai e Ramo), que predizem classificações *Não obrigatória* automaticamente. Estes métodos são apresentados detalhadamente nas Seções 5.1 e 5.2

5.1 Classificador Local por Nó Pai não Folha

Naturalmente, a estratégia Classificadores por nó Pai não consegue predizer classificações *Não obrigatória*. Considerando essa limitação, a estratégia denominada Classificação Local

por Nó Pai não Folha é proposta. Esta estende o conceito original de Classificador Local por nó Pai e habilita classificações *Não obrigatória* automaticamente.

Para possibilitar classificações *Não obrigatória*, a estratégia Classificador Local por Nó Pai não Folha (nLLCPN, **n**on-**L**eaf **L**ocal **C**lassifier per **P**arent **N**ode) modifica a hierarquia do problema, adicionando uma classe extra para cada classificador cujo nó é uma classificação *Não obrigatória*. A Figura 5.2 representa a mesma hierarquia da Figura 4.1a, porém adaptada com base no conjunto de dados apresentado na Tabela 5.1. Como pode-se notar, o classificador associado ao nó "2 possui uma classe extra, representando seu próprio nó (classificações *Não obrigatória*), pois existe uma instância que possui a classe "2 como mais específica. Naturalmente, como instâncias positivas, somente o Exemplo 1 será usado. Diferentemente, o mesmo não acontece para a classe "1, uma vez que esta não possui instâncias cujas classes mais profundas estão associadas a ela, i.e. não é uma classificação *Não obrigatória*.

Tabela 5.1: Conjunto de Dados de Exemplo

	Atributo - 1	Atributo - 2	Classe
Exemplo 1	1.0	A	2
Exemplo 2	2.0	B	1.2
Exemplo 3	3.0	B	1.1
Exemplo 4	2.0	A	2.1
Exemplo 5	1.0	C	2.2

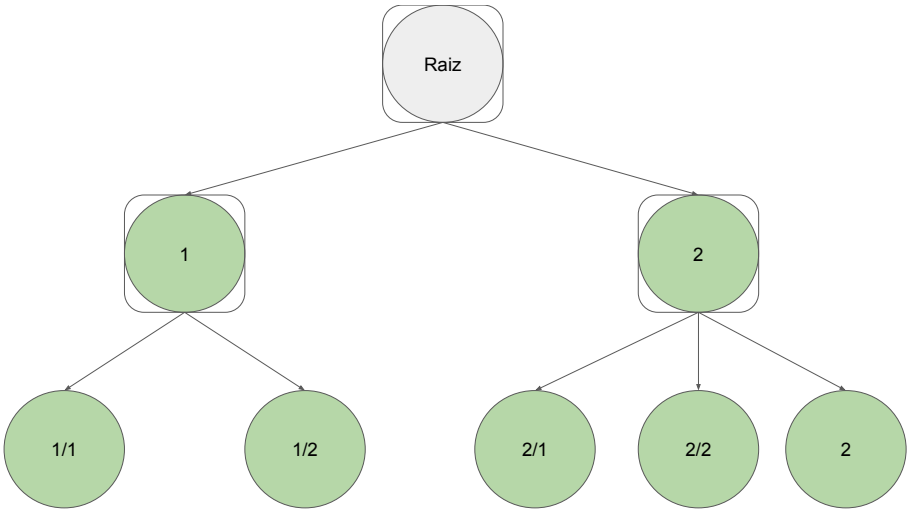


Figura 5.2: Modificação na hierarquia realizada pela estratégia nlLCPN.

Sua fase de treino é similar ao LCPN, entretanto, devido à modificação na hierarquia, os classificadores dos nós pais e que são classificações *Não obrigatória* aprendem a distinguir entre seus nós filhos e seu próprio nó. Como instâncias positivas para as classes *Não obrigatória*,

são utilizadas somente as instâncias cuja classe corresponde ao nó em questão. Novamente tomando o conjunto de dados apresentado na Tabela 5.1, somente o Exemplo 1 seria utilizado como exemplo positiva para a classe “2. Em relação ao paralelismo, os mesmos padrões da abordagem LCPN são mantidos, ou seja, possível na fase de treino, porém não na fase de teste.

Esta estratégia consegue obter classificações *Não obrigatória* de maneira eficiente, entretanto as deficiências do LCPN são mantidas. Sendo assim, erros provenientes de níveis superiores ainda são propagadas. Considerando esta deficiência, a estratégia LCPNB, cujo intuito consiste em minimizar a propagação de erros, é proposta.

5.2 Classificador Local por Nó Pai e Ramo

Utilizando a mesma modificação de hierarquia proposta na estratégia nLLCPN, esta estratégia denominada Classificador Local por Nó Pai e Ramo (LCPNB, **L**ocal **C**lassifier per **P**arent **N**ode and **B**ranch) expande o conceito da estratégia nLLCPN visando eliminar a propagação de erros.

A hipótese por trás desta estratégia sugere que a propagação de erros surge quando o classificador, possivelmente em níveis rasos, não consegue distinguir a classe da instância com precisão, resultando em probabilidades muitos próximas e conflitantes. Entretanto, classificadores de nós mais fundos são mais específicos e portanto aprendem a diferenciar características inerentes a nós mais fundos. Portanto, uma instância que é classificada erroneamente em níveis superiores pode ser ajustada utilizando classificadores de níveis mais fundos. Sendo assim, esta estratégia utiliza todos os classificadores da hierarquia para obtenção da classificação final.

Combinando os aspectos positivos dos trabalhos relacionados, mais especificamente a utilização de todos os classificadores, juntamente com a técnica de decisão baseado na média, a estratégia LCPNB em sua fase de treino aplica o mesmo processo que a nLLPCN, porém sua fase de teste é diferente. Esta estrategia expande todos os nós da hierarquia, somando as probabilidades dos ramos. Com todas as probabilidades somadas, ou seja, toda classe possível possui uma soma de probabilidades associada, a média das mesmas são calculadas. O ramo que apresentou maior média é selecionado como classificação final. Optou-se pela média, pois segundo Hernandez (HERNáNDEZ; SUCAR; MORALES, 2014), esta é mais eficiente e menos suscetível a erros numéricos. Tomando a Figura 5.3 como exemplo e as probabilidades especificadas, nota-se que, caso a estratégia LCPN fosse utilizada, a classificação obtida seria 1/2. Por outro lado, a estratégia LCPNB prediz uma classificação diferente. Ao tomar a média das probabilidade, a instância seria classificada em 2 com 62.5% de chance, pois sua probabilidade passa ser a

maior. As probabilidades de todas as classificações são listadas abaixo.

1. 1/1 = (0.55 + 0.4)/2 = 47.5 %
2. 1/2 = (0.55 + 0.6)/2 = 57.5 %
3. 2 = (0.45 + 0.80)/2 = 62.5 %
4. 2/1 = (0.45 + 0.1)/2 = 27.5 %
5. 2/2 = (0.45 + 0.1)/2 = 27.5 %

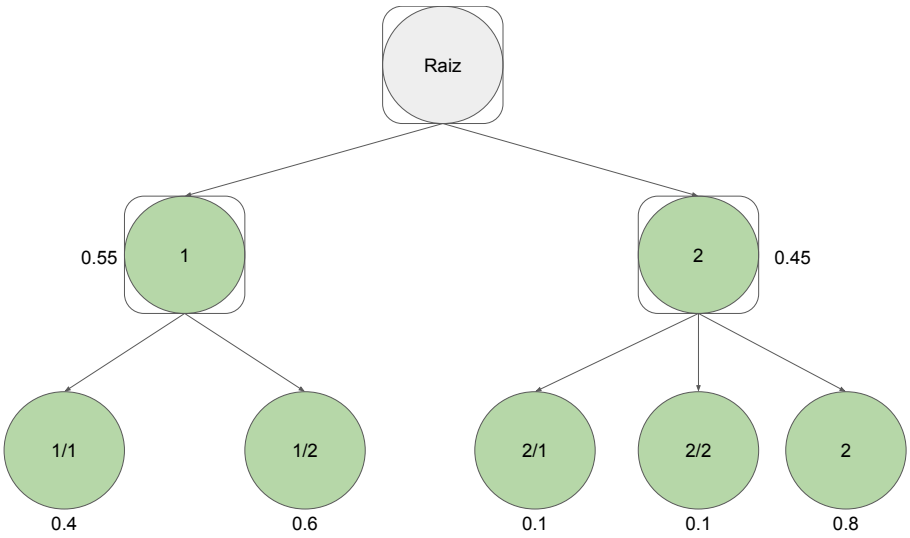


Figura 5.3: Funcionamento da estratégia LCPBN.

Em ambas fases de treino e teste, a estratégia LCPNB é paralelizável. Além disso, até o momento, esta estratégia é a única que naturalmente reduz a propagação de erros e permite classificações *Não obrigatória*.

5.3 Usando Deep Learning

De maneira similar as outras, as estratégias propostas nesta pesquisa podem ser usadas com flexibilidade em relação a seus classificadores locais. Como a primeira, nLLCPN, não depende de probabilidades para fazer sua classificação, qualquer classificador da literatura funcionaria normalmente, todavia este não é caso para a estratégia LCPNB.

Naturalmente, ao calcular a média de várias probabilidades, seria inviável utilizar classificador como Árvore de Decisão e *Naive-Bayes* (NB), considerando que Árvores normalmente

predizem uma classe com 100% de probabilidade, enquanto o algoritmo NB produz probabilidades muito próximas a 0 que podem afetar negativamente o cálculo da média. Desta maneira, é preferível utilizar classificadores que predizem classes com respeito a uma probabilidade, como, por exemplo, RNAs. Sendo assim, neste trabalho RNAs do paradigma *Deep Learning* foram investigadas.

Mais precisamente, motivado pelo seu desempenho superior, as redes *Deep* MLPs, *Denoising Auto-Encoder* e *Restricted Boltzmann Machine* são usadas como classificadores locais. Assim, variando de acordo com a estratégia utilizada, uma RNA é treinada para cada nó, nó pai, ou nível da hierarquia.

De maneira complementar, espera-se que as capacidades representativas destas redes se destaquem no contexto de CH, assumindo que este é o primeiro trabalho a explorar *Deep Learning* em problemas HSC (Hierarchical Single-Label Classification). Ao adicionar mais camadas ocultas, atributos mais representativos serão aprendidos incrementalmente.

Também é válido ressaltar que RNAs são classificadores altamente parametrizáveis. Neste sentido, várias combinações destes podem ser utilizadas para otimizar o seu desempenho, todavia a literatura ainda não oferece conteúdo neste contexto em específico, fato que possibilita uma nova área de pesquisa.

Capítulo 6

METODOLOGIA

Este capítulo contém todos os passos para obtenção das sequências de DNA utilizadas, assim como detalhes sobre a implementação dos métodos de Classificação Hierárquica e Deep Learning,

6.1 Visão Geral

Considerando que este trabalho é o primeiro a abordar a classificação de TEs como um problema de CH, a literatura ainda não apresenta conjuntos de dados. Sendo assim, sequências de DNA previamente anotadas com TEs foram extraídas de bancos de dados públicos, e mapeadas de forma a constituir um problema de CH. Em seguida, vetores de características foram extraídos e pre-processados. Por fim, duas etapas de experimentos foram realizados. A primeira etapa consiste em verificar se métodos de Classificação Hierárquica são superiores à homologia. Para tanto, métodos da abordagem local da literatura, assim como os propostos nesta pesquisa, foram comparados com os métodos de homologia BLASTn e RepeatMasker.

A segunda, e última etapa, consiste em comprovar se a utilização de *Deep Learning* melhora o desempenho dos método em problemas de CH. Para tanto, compara-se se a adição de camadas ocultas favorece ou não os resultados. Para enriquecer os experimentos nesse sentido, conjuntos de dados de proteínas do tipo GPCR (*G protein–coupled receptor*) e EZ ((*Enzyme Commission*) foram adicionados aos experimentos.

Em ambas etapas de experimentos, testes estatísticos são utilizados para validar seus resultados. A lista a seguir sumariza o desenvolvimento deste capítulo.

1. Obtenção do Conjunto de Dados (Sequências de DNA);

2. Extração de Características;
3. Pre-processamentos dos dados;
4. Avaliação dos resultados;
5. Experimentos 1 : Homologia vs Aprendizado de Máquina;
6. Experimentos 2 : Experimentos com *Deep Learning*.

Todas as etapas de desenvolvimento são explicadas com mais detalhes a seguir, enquanto os experimentos estão no próximo Capítulo.

6.2 Obtenção do Conjunto de Dados

De acordo com a nomenclatura de Silla (SILLA; FREITAS, 2010), todos os conjuntos de dados usados nesta pesquisa são definidos pela tupla (Árvore, HSC, *Não obrigatória*).

6.2.1 Elementos Transponíveis

Nesta pesquisa, foram montados três conjuntos de dados distintos a partir de repositórios públicos com TEs previamente anotados:

- PGSB: compilação de várias sequências repetitivas de plantas oriundas de bancos de dados públicos (TREP, TIGR repeats, PlantSat e Genbank) (NUSSBAUMER et al., 2013);
- REPBASE: referência mundial com sequências repetitivas provenientes de várias espécies eucarionte (JURKA et al., 2005);
- PGSB + REPBASE: concatenação dos dois conjuntos de dados.

Todas as sequências coletadas foram organizadas de maneira hierárquica de acordo com a taxonomia de Wicker (Figura 2.2). Vale ressaltar que estes repositórios não contêm todas as classes da taxonomia de Wicker. Isto é esperado, pois a taxonomia de Wicker é dada como uma tentativa de unificar a classificação de TEs, e ainda se encontra em desenvolvimento e adaptação.

Para a transcrição da classe de um TE aplicou-se um esquema numérico no qual cada sequência recebeu um nó da taxonomia de Wicker (Figura 2.2) como classe. Para exemplificar, supondo que uma sequência tenha o rótulo Gypsy, seu novo rótulo será 1/1/2, pois, de

acordo com a Figura 2.2, Gypsy (2) é uma subclasse da ordem LTR (1) que é uma subclasse da Classe 1 Retrotransposons. Desta maneira, cada instância recebe uma nova classe correspondente ao nó da hierarquia associado ao TEs. A Tabela 6.2 apresenta mais detalhes sobre cada repositório.

Detalhes estatísticos sobre estes conjuntos de dados são apresentados na Tabela 6.1. Adicionalmente, suas hierarquias são apresentadas nas Figuras 6.1 e 6.2. Vale ressaltar que a hierarquia da Figura 6.2 também é valida para o conjunto de dados PGSB + REPBASE.

Tabela 6.1: Estatísticas dos conjuntos de dados de TEs

Conjunto de dados	# Exemplos	# Atributos	# Classes por nível
PGSB	18678	336	2 / 4 / 3 / 5
REPBASE	34559	336	2 / 5 / 12 / 9
PGSB+REPBASE	53049	336	2 / 5 / 12 / 9

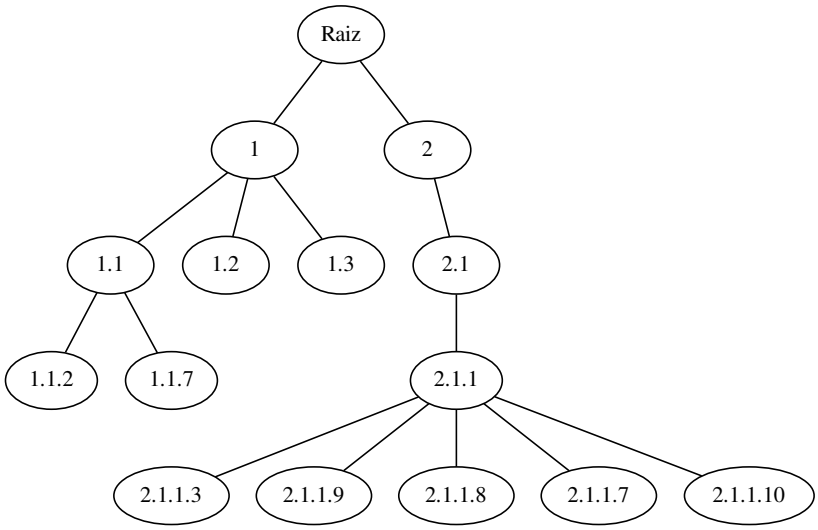


Figura 6.1: Hierarquia do conjunto de dados PGSB

6.2.1.1 Extração de Características

Atualmente existem diversas maneira para transformar sequências de DNA de comprimento desiguais em um vetor de características de tamanho fixo. Nesta pesquisa, optou-se por explorar o conjunto K-mers devido a sua alta capacidade discriminativa e popularidade em pesquisas similares.

Tabela 6.2: Distribuição de exemplos por classe

	Retrotransposons(1) [15998]	DNA Transposons(2) [2680]
PGSB	LTR(1.1) [15336] Copia(1.1.1) [4279] Gypsy(1.1.2) [7625] LINE (1.4) [471] SINE (1.5) [191]	Subclasse1 (2.1) [2680] TIR (2.1.1) [1600] Tc1-Mariner (2.1.1.1) [356] hAT (2.1.1.2) [63] Mutator (2.1.1.3) [320] PIF-Harbinger (2.1.1.8) [141] CACTA (2.1.1.9) [720]
	Retrotransposons(1) [22414]	**DNA Transposons(2) [12145]**
REPBASE	LTR(1.1) [18768] Copia(1.1.1) [6313] Gypsy(1.1.2) [10068] Bel-Pao(1.1.3) [1827] DIRS(1.2) [374] LINE (1.4) [2516] R2(1.4.1) [78] RTE(1.4.2) [439] Jockey(1.4.3) [242] L1(1.4.4) [1566] I(1.4.5) [194] SINE(1.5) [753] tRNA(1.5.1) [505] 7SL(1.5.2) [95] 5S(1.5.3) [29]	Subclasse1 (2.1) [12145] TIR (2.1.1) [7903] Tc1-Mariner (2.1.1.1) [2351] hAT (2.1.1.2) [2437] Mutator (2.1.1.3) [735] Merlin (2.1.1.4) [73] Transib (2.1.1.5) [123] P (2.1.1.6) [376] Piggybac (2.1.1.7) [353] PIF-Harbinger (2.1.1.8) [874] CACTA (2.1.1.9) [581]
	Retrotransposons(1) [38412]	**DNA Transposons(2) [14637]**
PGSB + REPBASE	LTR(1.1) [34104] Copia(1.1.1) [10592] Gypsy(1.1.2) [17693] Bel-Pao(1.1.3) [1827] DIRS(1.2) [374] LINE (1.4) [2990] R2(1.4.1) [78] RTE(1.4.2) [439] Jockey(1.4.3) [242] L1(1.4.4) [1566] I(1.4.5) [194] SINE(1.5) [944] tRNA(1.5.1) [505] 7SL(1.5.2) [95] 5S(1.5.3) [29]	Subclasse1 (2.1) [14637] TIR (2.1.1) [9315] Tc1-Mariner (2.1.1.1) [2707] hAT (2.1.1.2) [2500] Mutator (2.1.1.3) [1055] Merlin (2.1.1.4) [73] Transib (2.1.1.5) [123] P (2.1.1.6) [188] Piggybac (2.1.1.7) [353] PIF-Harbinger (2.1.1.8) [1015] CACTA (2.1.1.9) [1301]

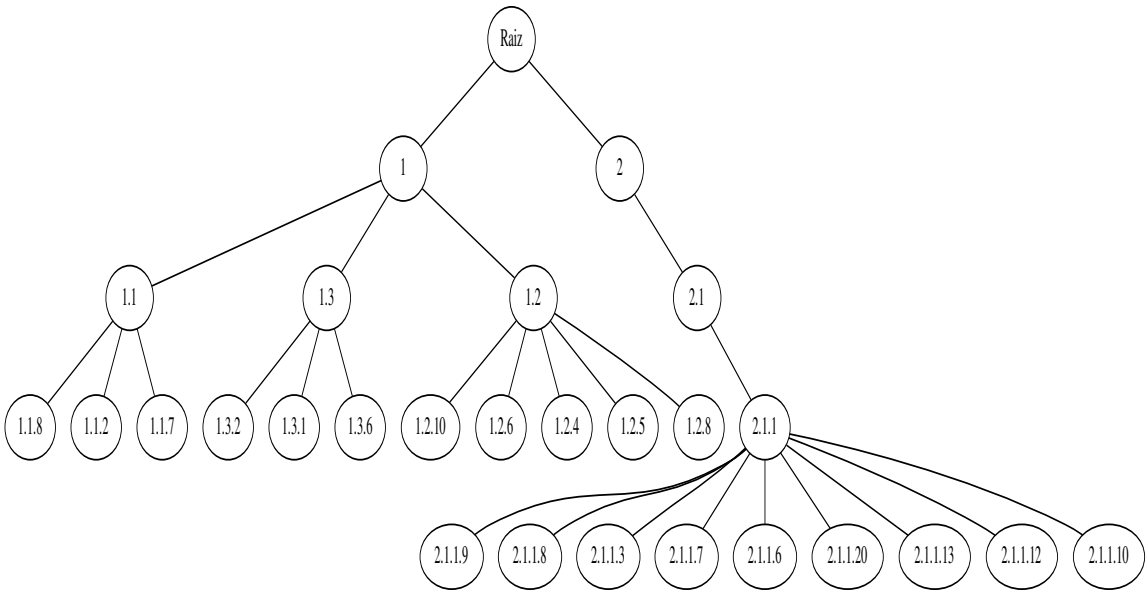


Figura 6.2: Hierarquia do conjunto de dados REPBASE

K-mers são descritos como uma maneira simples e eficiente de representar sequências de DNA com base em sua composição ácido nucleica. De maneira mais formal, dado uma sequência de DNA, K-mers são vetores do tipo $\{\alpha_1, \alpha_2, \alpha_3, ...\alpha_n\}$ no qual cada valor representa a quantidade de ocorrências da subsequência α_i, sendo α_i composto por elementos do conjunto $\{A, T, G, C\}$ e $|\alpha_i| = k$.

Para exemplificar, supondo $K = 2$ e a sequência $ATCGA$, as subsequências a serem contadas são $\{AA, AT, AC, AG, TA, TT, TC, TG, CA, CT, CC, CG, GA, GT, GC, GG\}$, resultando em $\{0,1,0,0,0,1,0,0,0,0,0,1,0,1,0,0\}$.

6.2.2 Proteínas

Como mencionado anteriormente, conjuntos de dados referentes a proteínas também foram utilizados para complementar a avaliação de métodos *Deep Learning*. Quando comparado à TEs, a classificação de proteínas, também chamada de PFP (*Protein Function Prediction*), já foi amplamente estudada em diversos trabalhos da literatura (SILLA; FREITAS, 2009b; SECKER et al., 2007, 2010; COSTA et al., 2008). Em particular, nesta pesquisa escolhemos utilizar os conjuntos de dados GPCR (*G protein–coupled receptor*) e EC (*Enzyme Commission*), devido às características da hierarquia serem idênticas as dos TEs (Árvore,SCH,*Não obrigatória*).

O primeiro, GPCR, consiste em proteínas capazes de reagir a moléculas que estão fora da

célula. Desta maneira, GPCR são capazes de ativar sinais físicos que correspondem à resposta adequada ao estímulo anterior. Também já é conhecido que GPCR estão envolvidas em várias doenças e uma grande quantidade de drogas as têm como alvo.

Por sua vez, EC correspondem a enzimas que são moléculas com funções catalizadoras. Isto é, são capazes de acelerar reações químicas, possibilitando o metabolismo dos seres vivos. Em grande parte, muitas reações não aconteceriam naturalmente sem elas. As características dos conjuntos de dados de proteínas são apresentadas na Tabela 6.3.

Tabela 6.3: Estatísticas dos conjuntos de dados de Proteínas

Conjunto de Dados	# Exemplos	# Atributos	# Classes por Nível
ECInterpro	13902	1216	6 / 41 / 96 / 187
ECProsite	13877	585	6 / 42 / 89 / 187
ECPrints	13828	382	6 / 45 / 92 / 208
ECPfam	13871	708	6 / 41 / 96 / 191
GPCRInterpro	7353	450	12 / 54 / 82 / 50
GPCRProsite	6167	129	9 / 50 / 79 / 49
GPCRPrints	5324	283	8 / 46 / 76 / 49
GPCRPfam	6967	75	12 / 52 / 79 / 49

6.2.2.1 Extração de Características

Para ambos tipos de proteínas, foram utilizados a presença de assinaturas (*motifs*), peso molecular e comprimento de sequência. Para cada sequência, verifica-se se uma determinada assinatura, de acordo com um dos quatro repositórios, caso esteja presente seu valor é 1, caso não, 0. Por sua vez, o peso molecular corresponde a soma acumulada do peso atômico de cada nucleotídeo. Por fim, o comprimento molecular é dado pela quantidade de nucleotídeos presentes na sequência.

Variando de acordo com o repositório, assinaturas correspondem a certos padrões presentes na sequência, em grande parte estes são subsequências associadas a uma determinada propriedade. O repositório Prosite utiliza assinaturas baseadas em expressões regulares; Prints utiliza vários grupos de assinaturas não sobrepostas; o restante deles, com exceção do Interpro, emprega Cadeias Ocultas de Markov. Por sua vez, o repositório Interpro consiste na concatenação de várias assinaturas providas por múltiplos repositórios [1] (COSTA et al., 2007).

[1] https://www.ebi.ac.uk/interpro/

6.3 Pre-processamento dos Dados

Com o intuito de ajustar a distribuição dos atributos, de maneira que todos possuam a mesma relevância, na etapa de pre-processamento, aplica-se a normalização. Esta consiste em ajustar os atributos utilizando a Equação 6.1 no qual $min(x)$ corresponde ao valor mínimo de um determinado atributo x, $max(x)$ ao valor máximo e x' ao novo valor de x normalizado.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{6.1}$$

Ao final, o atributo com maior valor passa a ser 1, enquanto o menor valor torna-se 0. Todos os demais atributos possuem valores entre os valores 0 e 1.

6.4 Avaliação dos Resultados

Para problemas de classificação tradicional, medidas como: *precision*, *recall* e *f-measure*, são frequentemente utilizadas, entretanto estas não apropriadas para esta pesquisa, pois desconsideram aspectos relacionados à hierarquia de classes. Sendo assim, medidas específicas para CH foram adotadas.

6.4.1 Medidas de Classificação Hierárquica

Devido a sua popularidade e simplicidade, serão utilizadas as medidas *hierarchical precision (hP)* e *hierarchical recall (hR)*. Inicialmente propostas por (KIRITCHENKO; MATWIN; FAMILI, 2005) e recomendadas por (SILLA; FREITAS, 2009b), elas estendem as medidas homônimas e consideram que um exemplo pertence não apenas a uma classe, mas também a todas as suas classes ancestrais na estrutura hierárquica. As medidas são definidas formalmente nas Equações 6.2 e 6.3, sendo C_i o conjunto de todas as classes esperadas de um exemplo i, e Z_i o conjunto de todas as classes preditas para um exemplo i.

$$hP = \frac{\sum_i |Z_i \cap C_i|}{\sum_i |Z_i|} \tag{6.2}$$

$$hR = \frac{\sum_i |Z_i \cap C_i|}{\sum_i |C_i|} \tag{6.3}$$

Como pode-se notar, essas medidas são facilmente computáveis e não dependem de parâmetros extras. Todavia, utilizadas isoladamente elas são inadequadas para a avaliação de classificadores (SEBASTIANI, 2002). Sendo assim, elas devem ser combinadas na medida *hierarchical f-measure (hF)* (Equação 6.4).

$$hF = \frac{2 * hP * hR}{hP + hR} \tag{6.4}$$

Afim de exemplificar o cálculo destas medidas, exemplos são apresentados na Tabela 6.4. O Exemplo 1 contém uma classificação no qual espera-se a classe 1/1/2, porém o modelo obteve a classe 1/1/1. Neste caso, obteve-se uma resposta errada somente no último nível, desta maneira as medidas apresentam valores parciais, 2/3. No segundo exemplo, exemplifica-se uma situação no qual não houve nenhum acerto, ou seja, uma classificação totalmente errônea, naturalmente todas as medidas apresentam valor 0. Por outro lado, o terceiro exemplo apresenta uma classificação inteiramente correta, obtendo o valor 1 para todas as medidas. Por fim, o último exemplo mostra uma classificação parcialmente correta, no qual espera-se a classe 2/1/1/1, todavia obteve-se a classe 2/1, portanto o valor da medida *hP* corresponde à 1, enquanto *hR* se limita a 0.5, resultando em 2/3 para a hF. Vale ressaltar que a notação as classes 1/2 e 2/2 representam classes diferentes, pois esta notação representa a subclasse 2 filha da classe 1, enquanto a subclasse 2 está associada a classe 2 respectivamente.

Tabela 6.4: Exemplo de predições e medidas de avaliação por Nível

	Esperado	Predito	$\|Z_i\|$	$\|C_i\|$	$\|Z_i \cap C_i\|$
Exemplo 1	1/1/2	1/1/1	3	3	2
Exemplo 2	2/1/1/1	1/1/1	3	4	0
Exemplo 3	1/1/1	1/1/1	3	3	3
Exemplo 4	2/1/1/2	2/1	2	4	2
Total	-	-	11	14	7
-	*hP*	*hR*	*hF*	-	-
-	7/11	7/14	0.56	-	-

Para melhor compreender os resultados em trabalhos de Classificação Hierárquica, também avalia-se os resultados por níveis da hierarquia. Neste caso, são consideradas somente classes até um determinando nível, eliminando classes que estão além do nível em questão com respeito as classes esperadas. Também com o intuito de exemplificar, a Tabela 6.5, no qual $|LZ_i|$ corresponde à medida $|Z_i|$ por nível, por sua vez $|LC_i|$ corresponde à medida $|Zi|$ e naturalmente $|LZ_i \cap LC_i|$ corresponde a intersecção, exemplifica como ajustar o vetor de classes para calcular as medidas com respeito a cada nível.

Semelhantemente as versões já apresentadas, na versão por nível as mesmas Equações são utilizadas. Para cada exemplo, obtém-se um vetor no qual cada posição possui um valor associado a medida do nível em questão. No caso da medida $|LZ_i|$, o vetor possui a quantidade de classes preditas para cada nível da hierarquia, por exemplo, para o Exemplo 1, as classes 1/1/1

Tabela 6.5: Exemplo de predições e medidas de avaliação por Nível

	Esperado	Predito	$\|LZ_i\|$	$\|LC_i\|$	$\|LZ_i \cap LC_i\|$
Exemplo 1	1/1/2	1/1/1	[1,2,3,0]	[1,2,3,0]	[1,2,2,0]
Exemplo 2	2/1/1/1	1/1/1	[1,2,3,0]	[1,2,3,4]	[0,0,0,0]
Exemplo 3	1/1/1	1/1/1	[1,2,3,0]	[1,2,3,0]	[1,2,3,0]
Exemplo 4	2/1/1/2	2/1	[1,2,0,0]	[1,2,3,4]	[1,2,0,0]
Total	-	-	[4,8,9,0]	[4,8,12,8]	[3,6,5,0]

foram preditas, desta maneira o vetor deve conter os valores [1,2,3], pois foram preditas 1,2 e 3 classes respectivamente. Para a medida $|LC_i|$, considera-se as classes esperadas para cada nível, no Exemplo 2, utiliza-se o vetor [1,2,3,4] uma vez que sua classe é 2/1/1/1.

Por fim, no caso da intersecção, $|LZ_i \cap LC_i|$, conta-se a quantidade de classes que estão presentes em ambos Esperado e Predito com prioridade ao Esperado. Isto é, caso o comprimento do vetor de classes do predito seja maior que o esperado, classes preditas de níveis além do esperado são ignoradas. De maneira informal, itera-se ambos vetores de classes, retirando classes além do nível em questão, até que todas os níveis do Esperado sejam avaliados. Por exemplo, considerando o Exemplo 1, primeiramente compara-se as classes 1 com 1, em seguida compara-se 1/1 com 1/1, e, finalmente, 1/1/2 com 1/1/1, resultando no vetor [1,2,2], pois no último nível somente duas classes são iguais. A Tabela 6.6 apresenta as medidas calculas por nível.

Tabela 6.6: Medidas hP,hR e hF por Nível

hP	*hR*	*hF*
[3/4,6/8,5/12,0]	[3/4,6/8,5/9,0]	[3/4,6/8,0.47,0]

6.4.2 Testes Estatísticos

Para garantir a relevância estatísticas dos resultados, utilizou-se a avaliação proposta por Densar (DEMSAR, 2006). Nesta, primeiramente verifica-se se existe um método que se diferencia dos demais usando o teste de Friedman, e em caso positivo, aplica-se o teste de Nemenyi para ranquear os métodos. Intuitivamente, métodos com desempenhos superiores estão nas posições mais altas do ranking. Por sua vez, métodos conectados por uma barra horizontal não apresentam diferença estatisticamente significativa.

6.5 Abordagens e Algoritmos

Neste trabalho foram implementadas algumas das abordagens de CH detalhadas no Capítulo 4, juntamente com as RNAs *Denoising Auto-Encoder* e *Restricted Boltzmann Machine* descritas anteriormente.

6.5.1 Abordagens de Classificação Hierárquica

Para o desenvolvimento de abordagens de CH, preferiu-se utilizar as estratégias locais, devido a possibilidade de utilizações de RNAs como classificadores, mais precisamente as estratégias locais LCPN e LCL.

Optou-se por não empregar a estratégia LCN pois, como já detalhado na Subseção 4.2.1, a estratégia LCN apresenta maior complexidade computacional e pode levar a múltiplas inconsistências. Além disso, vários trabalhos recentes optam por utilizar LCPN (Subseção 4.2.2.1) e LCL (Subseção 4.2.4).

Naturalmente, as estratégias propostas nesta pesquisa, nlLCPN e LCPNB, também foram investigadas. Como ressaltado anteriormente, estas estratégias adicionam um nó fictício para cada nó pai que também é uma classificação $\{obrigatória\}$. Para exemplificar sua aplicação nos conjuntos de dados, as Figuras 6.3 e 6.4 apresentam as hierarquias modificadas para os conjuntos de dados PSGB e REPBASE respectivamente.

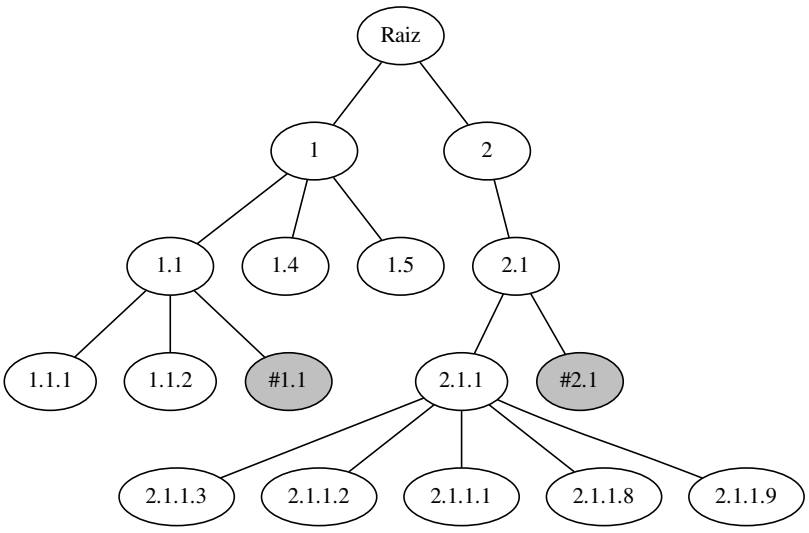


Figura 6.3: Hierarquia modificada do conjunto de dados PGSB

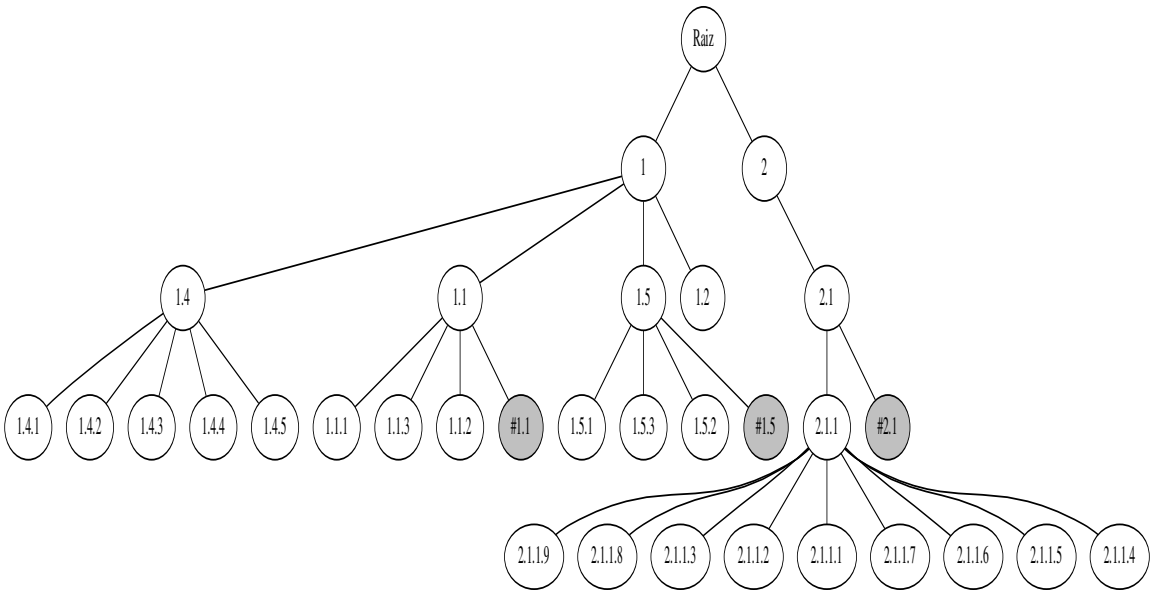


Figura 6.4: Hierarquia modificada do conjunto de dados REPBASE

No caso da estratégia LCL, as correções SWV (PAES; PLASTINO; FREITAS, 2012) e Simple-Prune (CERRI et al., 2016) foram utilizadas, devido ao fato que seus métodos de correção são consideravelmente diferentes. O método SWV utiliza as probabilidades preditas pelos classificadores para ajustar sua classificação. Por sua vez, o método Simple-Prune simplesmente elimina qualquer classificação cujo nó pai não foi predito. No caso da estratégia LCPN, inconsistências não são possíveis.

As estratégias testadas nesta pesquisa são listadas a seguir.

- LCPN (Subseção 4.2.2);
- LCL com correção SWV (Subseção 4.2.3);
- LCL com correção SimplePrune (Subseção 4.2.3);
- nlLCPN (Subseção 5.1);
- LCPNB (Subseção 5.2).

Como classificadores locais, optou-se por utilizar Deep MLPs (Subseção 3.1.2), RBMs e Denoising Auto-Encoders. Desta maneira, cada classificador local consiste em uma RNA. Neste trabalho, preferiu-se fixar somente um tipo de classificador, juntamente com seus parâmetros, uma vez que a otimização destes resulta em um número exponencial de combinações

A abordagem *Flat* não foi utilizada, pois apesar de ser uma abordagem de CH, ela não considera aspectos hierárquicos e consequentemente foge do escopo do trabalho.

Também é válido ressaltar que, até o momento, não existe nenhuma biblioteca ou *Framework* específico para CH. Sendo assim, as abordagens desta pesquisa foram implementadas utilizando a biblioteca de AM *Scikit-learn* (PEDREGOSA et al., 2011) em *Python*.

6.5.2 Deep Learning

Para o desenvolvimento de métodos de *Deep Learning*, mais precisamente as RNAs Auto-Encoder e Stacked Auto-Encoder, foi utilizada a API *Keras*. Esta é uma biblioteca em *Python* que possibilita a construção de diversos tipos de RNAs de maneira intuitiva e modular. Além disso, possibilita a utilização de diversos parâmetros recém propostos como a função de ativação ReLu (NAIR; HINTON, 2010) e otimizadores como o Adam (KINGMA; BA, 2014b). Como sua implementação requer a utilização de um *backend*, optou-se por utilizar o Theano (Theano Development Team, 2016). Trata-se de uma biblioteca Python programável cujo intuito é acelerar, através de paralelismo massivo, a geração de código em baixo nível e utilização de GPU (*Graphics Processing Unit*).

Como parâmetros para as RNAs, foram utilizadas os seguintes valores: 200 neurônios para cada camada oculta, 200 épocas, 1 a 3 camadas ocultas e inicialização uniforme de pesos entre -0.05 a 0.05. Cada camada oculta é seguida de uma camada de *Dropout*, com 50%, para as MLPs e uma camada de ruído, também de 50%, é atribuída para a camada de entrada dos Auto-Encoders.

No caso da MLPs, o algoritmo Adam foi utilizado com seus parâmetros padrões (KINGMA; BA, 2014a). Por sua vez, para os Auto-Encoders o algoritmo BackPropagation foi utilizado para o pré-treino, seguido novamente pelo Adam para o ajuste fino. Em relação as funções de ativação, foi utilizada função de ativação Relu nas camadas ocultas, enquanto nos Auto-Encoders foi utilizada a *softmax* na camada de saída. A Equação 6.5 descreve a função *softmax*, no caso γ_j corresponde ao valor de ativação proveniente do neurônio j e m refere-se a quantidade total de neurônios.

$$\frac{e^{\gamma_j}}{\sum_j^m e^{\gamma_j}} \tag{6.5}$$

Surpreendentemente, o Keras não oferece um módulo de RBMs. Portanto, RBMs e DBNs foram implementadas diretamente utilizando a biblioteca *Theano*. Como algoritmo de treina-

mento, foi utilizado o *Contrastive Divergence* (Subseção 3.2.1) com $k = 1$. Em ambas camadas ocultas e de saída, a função logística foi utilizada.

6.6 Homologia

Normalmente, os métodos de homologia utilizam um vasto banco de dados no qual a sequência buscada é comparada com inúmeras outras sequências. Entretanto, devido à aplicação de técnicas de AM, optou-se por separar as sequências de DNA em conjuntos de teste e treino.

Desta maneira, os métodos BLASTn e RepeatMasker serão executados seguindo o mesmo procedimento que as abordagens de CH. No caso, foram utilizados 10-*fold cross-validation*. Os outros métodos discutidos no Capítulo 2 não foram avaliados, pois apresentam as seguintes deficiências: limitados a somente certos tipos de TEs, disponibilizados somente com classificadores treinados ou não acessíveis publicamente.

6.6.1 BLASTn

Como já especificado na Subseção 2.1.1, o BLAST requer arquivos do tipo FASTA como entrada. Para preservar os dados correspondentes à sequência, seu identificador (id) e sua classe são armazenados.

Em seguida, é necessária usar a ferramenta FORMATDB. Esta converte ambos os arquivos FASTA de treino e teste para o formato de uso interno do BLAST. Para isso é utilizada a linha de comando: $formatdb - pF - ifilepath - nfileoutput$.

O parâmetro $-pF$ indica que as sequências em questão correspondem a nucleotídeos. Por sua vez, os parâmetros $-i$ e $-n$ indicam os caminhos dos arquivos de entrada e saída respectivamente. Intuitivamente, $filepath$ e $fileoutput$ correspondem ao caminho dos arquivos e fasta e caminho dos arquivos de saída respectivamente.

Em seguida, executa-se o BLASTn. É válido ressaltar que o BLAST não é desenvolvido especificamente para TEs. Neste âmbito, o trabalho de (WICKER et al., 2007) sugere a utilização dos parâmetros *identity* e *cover* com valor acima de 80.

O parâmetro *identity*, referenciado por $-perc\_identity$, corresponde à extensão no qual as sequências possuem o mesmo nucleotídeo. Por sua vez, o parâmetro *cover*, $-qcov\_hsp\_perc$, corresponde à porcentagem de cobertura do alinhamento em relação ao comprimento total da sequência.

Também decidiu-se selecionar somente a sequência alinhada com maior confiabilidade. Desta maneira, o parâmetro $-max\_target\_seqs$ é setado para 1. Os outros parâmetros são mantidos em *default*. Como formato de saída, optou-se por usar o XML devido a sua interpretabilidade.

Para processar os resultados do BLASTn, utilizou-se a biblioteca Biopython[2]. Esta oferece o *parsing* de maneira eficiente dos XML.

Desta maneira, ao obter a sequência homóloga, recupera-se sua classe e esta é dada como classificação. Para algumas sequências, não obtêm-se sequência homóloga, ou seja, o BLASTn não foi capaz de obter um alinhamento. configurando um caso de *No Hit*. Nestas situações, a classe majoritária é utilizada para a classificação.

Nesta pesquisa, utilizou-se o BLASTn na versão 2.2.31+, associado com o FORMATDB versão 2.2.26.

6.6.2 Repeat Masker

Da mesma maneira que o método BLASTn, o RepeatMasker também requer arquivos no formato FASTA. Como já apresentado na Subseção 2.1.2, diferentes mecanismos de busca podem ser utilizados, entretanto preferiu-se utilizar o Cross_Match devido a sua popularidade.

Também de maneira semelhante ao BLASTn, o RepeatMasker será utilizado como um classificador no qual a classe associada à sequência homóloga será dada como classificação.

Nesta pesquisa, utilizou-se o Repeat Masker versão 4.0.7 e Cross_Match versão 0.990319.

[2] http://biopython.org/

Capítulo 7

EXPERIMENTOS

Este Capítulo traz os resultados e a discussão de ambas as etapas de experimentos. Inicialmente, discute-se sobre os resultados da homologia quando comparada ao Aprendizado de Máquina. Em seguida, experimentos utilizando métodos de Deep Learning são investigados

Para melhor sumarizar e representar os resultados, estes são discutidos utilizando a métrica *hierarchical f-measure* Equação 6.4. No caso, seus valores são representativos, pois contêm as outras métricas em sua formulação.

Igualmente para facilitar o compreendimento dos experimentos, nos experimentos de Aprendizado de Máquina com Homologia, foi analisado o cenário geral e cada nível individualmente. No caso dos experimentos comparando *Deep Learning*, optou-se por apresentar somente a Tabela com os resultados considerando todos os níveis e os testes de Nemenyi de cada nível individualmente. As Tabelas referentes aos resultados por níveis estão disponíveis separadamente no Apêndice A, devido ao grande volume de resultados.

7.1 Aprendizado de Máquina vs Homologia

Primeiramente métodos de Aprendizado de Máquina (AM) tradicionais são comparados com a homologia. Para tanto, considerou-se somente a utilização de MLPs com uma camada oculta. A Tabela 7.1 apresenta os resultados dos métodos de homologia BLASTn e RepeatMasker (RM). Por sua vez, resultados de estratégias locais de Classificação Hierárquica considerando todos os níveis estão contidos na Tabela 7.2.

Ao comparar os métodos de homologia entre si, percebe-se que, com exceção do Nível 3 no conjunto de dados PGSB, em todas as situações o método RM apresentou resultados superiores. De certa maneira, este comportamento é esperado, considerando que o RM foi desenvol-

Tabela 7.1: Resultados dos métodos de Homologia

	BLASTn	Blastn - No Hits	RepeatMasker	RepeatMasker - No Hits
			PGSB	
Todos	0.80 ±0.01	0.55 ±0.011	**0.81 ± 0.01**	0.58 ± 0.08
Nível 1	0.90±0.01		**0.91**±0.07	
Nível 2	0.88±0.01		**0.91**±0.08	
Nível 3	**0.73**±0.017		0.72±0.01	
Nível 4	0.34±0.05		**0.47**±0.06	
			REPBASE	
Todos	0.54 ± 0.06	0.91 ± 0.02	**0.68 ± 0.02**	0.60 ± 0.005
Nível 1	0.68±0.08		**0.68**±0.01	
Nível 2	0.68±0.08		**0.78**±0.01	
Nível 3	0.56±0.06		**0.70**±0.02	
Nível 4	0.21±0.15		**0.62**±0.08	
			PGSB +REPBASE	
Todos	0.63 ± 0.04	0.79 ±0.113	**0.72 ± 0.020**	0.59 ± 0.070
Nível 1	0.76±0.03		**0.78**±0.01	
Nível 2	0.76±0.03		**0.78**±0.01	
Nível 3	0.62±0.03		**0.70**±0.01	
Nível 4	0.24±0.08		**0.62**±0.05	

vido especialmente para sequências com repetições, enquanto o BLAST é um método geral de homologia.

Apesar de estarem sendo utilizados em um contexto de Aprendizado de Máquina, nota-se que os métodos apresentam resultados satisfatórios em algumas situações. Em particular, no conjunto de dados PGSB, ambos apresentaram resultados próximos à 0.8 quando analisando todos os níveis e mantiveram resultados altos nos primeiros dois níveis. Todavia, no caso do REPBASE e PGSB+REPBASE, os resultados são desencorajadores quando comparados às técnicas de AM.

Adicionalmente, a quantidade de *No Hits* (sem classificação) é surpreendentemente alta em todos os conjunto de dados. Na melhor situação, o método BLASTn obteve 55% de sequências sem classificação. Desta maneira, a inviabilidade desses métodos segundo esta metodologia é ressaltada. Ainda, pode-se afirmar que os resultados foram acrescidos artificialmente devido a classificação padrão atribuída para a classe majoritária no caso de *No Hits*.

Porém, deve-se levar em consideração que normalmente há uma vasta biblioteca de sequências pré-anotadas, consideravelmente maior que os *folds* utilizados. Assim, não deve-se desconsiderar a homologia por inteira. Esta é um dos métodos mais consolidado na literatura de bioinformática e sua utilização em seu contexto ainda é altamente explorada.

Ao comparar as estrategias locais de HC com homologia, percebe-se que métodos de AM

Tabela 7.2: Resultados dos métodos de Machine Learning

	PGSB				
	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
Todos	0.88±0.02	0.88±0.02	0.88±0.02	0.90±0.03	**0.91**±0.03
Nível 1	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02
Nível 2	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02
Nível 3	0.80±0.02	0.80±0.02	0.80±0.02	**0.85**±0.03	**0.85**±0.03
Nível 4	0.59±0.07	0.58±0.08	0.59±0.07	0.63±0.08	**0.64**±0.08
	REPBASE				
Todos	0.85±0.01	0.85±0.01	0.85±0.01	**0.86**±0.01	**0.86**±0.01
Nível 1	0.95±0.01	0.95±0.01	0.95±0.01	**0.95**±0.01	**0.95**±0.01
Nível 2	0.95±0.01	0.95±0.01	0.95±0.01	**0.95**±0.01	**0.95**±0.01
Nível 3	0.84±0.01	0.84±0.01	0.84±0.01	**0.85**±0.01	**0.85**±0.01
Nível 4	0.69±0.02	0.69±0.02	0.69±0.02	0.69±0.05	**0.70**±0.05
	PGSB +REPBASE				
Todos	0.85±0.02	0.85±0.02	0.85±0.02	**0.86**±0.02	**0.86**±0.02
Nível 1	**0.95**±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01
Nível 2	**0.95**±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01
Nível 3	0.82±0.02	0.81±0.02	0.82±0.01	**0.83**±0.02	**0.83**±0.02
Nível 4	0.66±0.02	0.65±0.02	**0.66**±0.03	0.65±0.07	0.66±0.07

foram superiores. Segundo o teste estatístico (Figura 7.1), a estratégia proposta nesta pesquisa, LCPNB, foi ranqueada em primeiro lugar com relevância estatística sobre as outras estratégias ao analisar todos os níveis. Por sua vez, métodos de homologia foram designados como estatisticamente inferiores aos métodos de AM, porém igualmente entre si.

No primeiro e segundo níveis da hierarquia (Figuras 7.2 e 7.3), desempenhos similares foram obtidos usando AM. No geral, todas as estrategias locais apresentaram um valor de aproximadamente 95% nos três conjuntos de dados com desvio padrão mínimo. Estes valores motivam a aplicação de AM, principalmente em casos cujo interesse reside nestes níveis. Como esperado, não existe diferença estatística entre as estratégias de HC, todavia os métodos de homologia, novamente, são ranqueadas em posições inferiores.

No terceiro nível, percebe-se resultados similares quando comparados a todos níveis (Figura 7.4). A estratégia LCPN e suas variantes, nlLCPN e LCPNB, juntamente com a SWV, apresentaram resultados estatisticamente superiores.

Inesperadamente, no último nível (Figura 7.5), o método RepeatMasker foi estatisticamente equivalente às estrategias SimplePrune, LCPN e SWV, enquanto as estratégias LCPNB, nlLCPN e SWV são superiores as demais, porém iguais entre si. Este comportamento é ex-

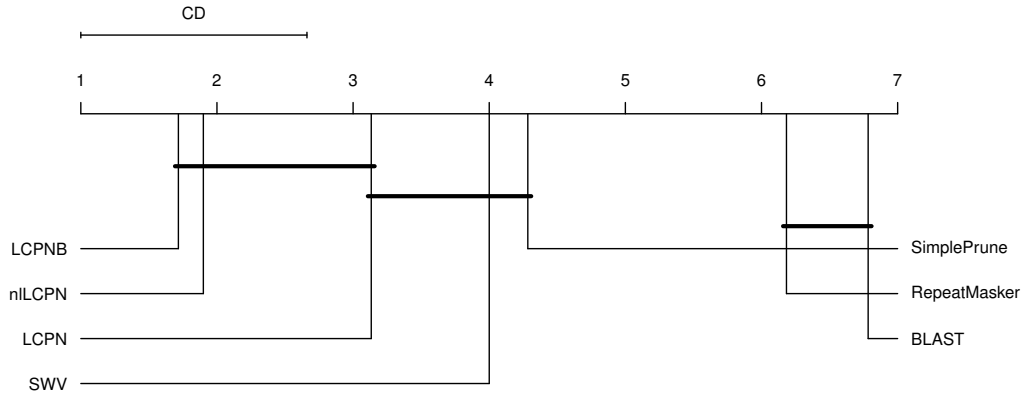


Figura 7.1: Teste de Nemenyi - Comparando Homologia com Machine Learning - Todos Níveis

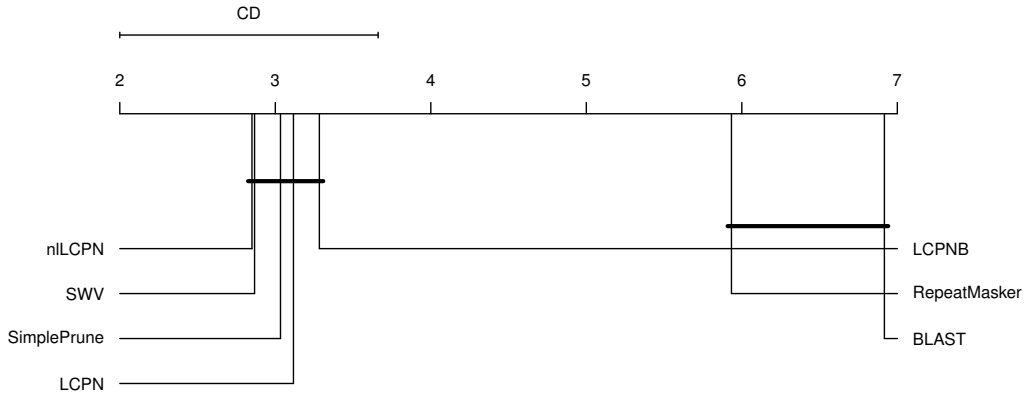


Figura 7.2: Teste de Nemenyi - Comparando Homologia com Machine Learning - Nível 1

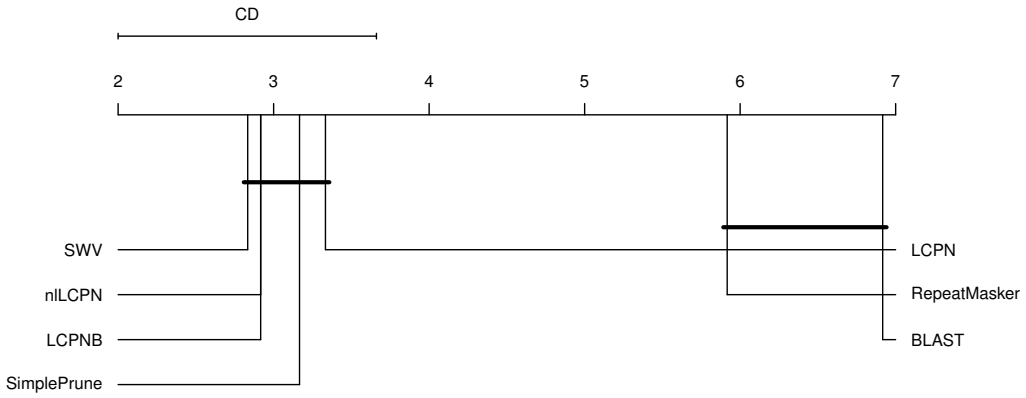


Figura 7.3: Teste de Nemenyi - Comparando Homologia com Machine Learning - Nível 2

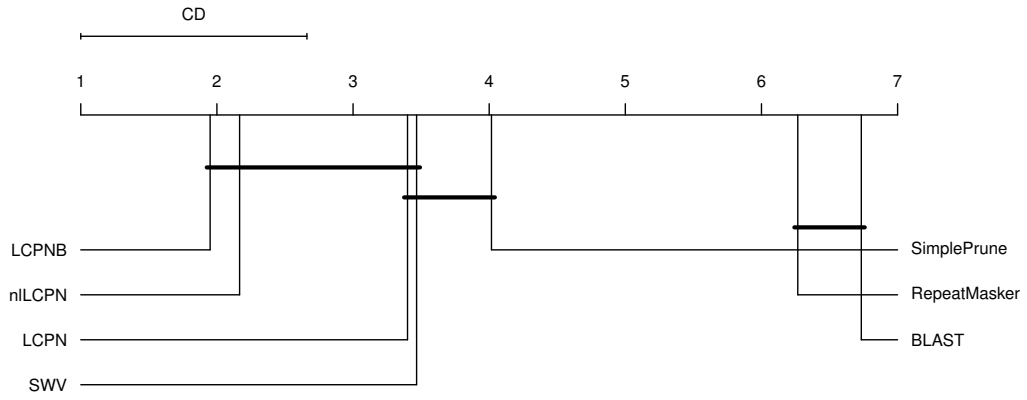


Figura 7.4: Teste de Nemenyi - Comparando Homologia com Machine Learning - Nível 3

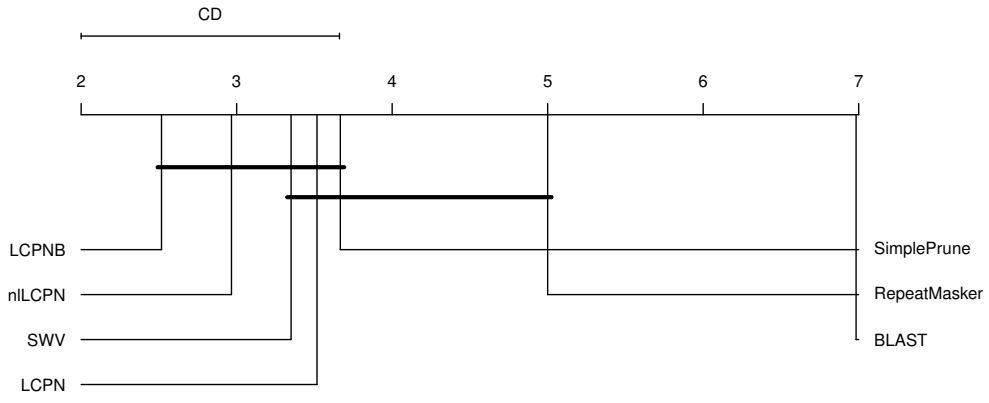


Figura 7.5: Teste de Nemenyi - Comparando Homologia com Machine Learning - Nível 4

plicado pelo desbalanceamento dos conjuntos de dados no qual uma quantidade mínima de instâncias pertencem ao último nível da hierarquia. Em concordância com as outras situações, o método BLASTn apresentou o pior desempenho.

7.2 Experimentos com Deep Learning

Nesta etapa, foi estudado como as redes neurais do paradigma *Deep Learning* se comportam em problemas de Classificação Hierárquica, destacando aspectos como a adição de camadas ocultas, assim como sua capacidade de generalização especialmente em níveis inferiores. Estes experimentos buscam estudar como a adição de camadas ocultas afeta o desempenho das redes neurais. Para tanto redes neurais com 1 à 3 camadas ocultas foram comparadas entre si independente da abordagem de Classificação Hierárquica. Para comprovar os resultados, testes estatísticos foram realizados levando em consideração a média de cada abordagem em todos os conjuntos de dados por *fold*.

Em todas as figuras e tabelas nesta seção, adotou-se uma notação na qual a quantidade de camadas ocultas é descrita logo após a sigla da rede neural utilizada, por exemplo, Stacked Denoising Auto-Encoder com três camadas ocultas (SDA3). Como também destacado no Capítulo 6, com o intuito de complementar a análise, oito conjuntos de dados da literatura, referentes a predição de funções de proteínas, foram adicionados.

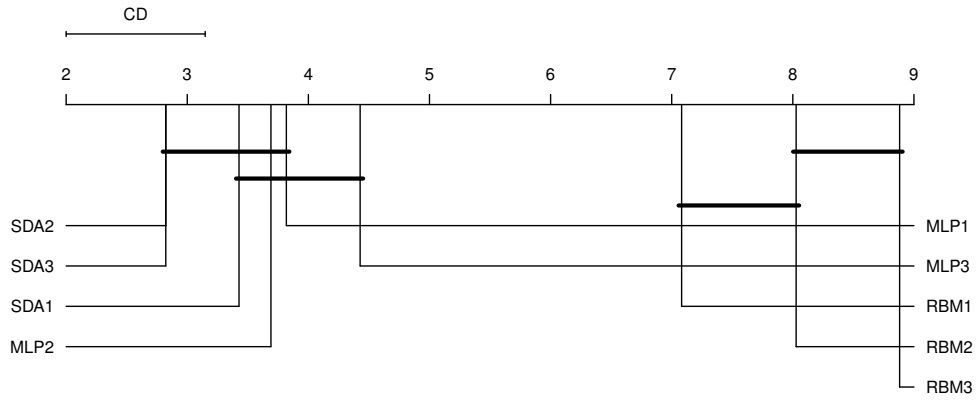


Figura 7.6: Teste de Nemenyi - Comparando Deep Learning com Machine Learning - Todos Níveis

Como observado nas Tabelas 7.3, 7.4, 7.5, e na Figura 7.6, a adição de camadas ocultas influenciou consideravelmente o resultado. No caso das Restricted Boltzmann Machines (RBMs) e MLPs, uma quantidade menor de camadas levou a resultados superiores. Este desempenho é desmotivador neste contexto, visto que quanto mais profunda a rede, espera-se resultados superiores.

Tabela 7.3: Resultados nos conjunto de dados de TEs

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
PGSB					
MLP1	0.88±0.02	0.88±0.02	0.88±0.02	0.90±0.03	0.91±0.03
MLP2	0.89±0.02	0.88±0.02	0.88±0.02	0.90±0.03	0.90±0.03
MLP3	0.88±0.03	0.88±0.02	0.88±0.02	0.90±0.03	0.90±0.03
SDAE1	0.88±0.02	0.88±0.02	0.88±0.02	0.90±0.02	0.90±0.02
SDAE2	0.89±0.02	0.88±0.02	0.88±0.03	0.90±0.02	**0.91**±0.02
SDAE3	0.89±0.02	0.89±0.02	0.89±0.02	0.91±0.03	0.91±0.03
RBM1	0.87±0.03	0.87±0.03	0.83±0.08	0.86±0.04	0.86±0.04
RBM2	0.85±0.07	0.84±0.07	0.81±0.1	0.79±0.11	0.78±0.11
RBM3	0.56±0.14	0.56±0.14	0.59±0.12	0.5±0.15	0.5±0.15
REPBASE					
MLP1	0.85±0.01	0.85±0.01	0.85±0.01	0.86±0.01	0.86±0.01
MLP2	0.86±0.01	0.85±0.01	0.86±0.01	0.86±0.01	0.86±0.01
MLP3	0.85±0.01	0.84±0.01	0.85±0.01	0.86±0.01	0.86±0.01
SDAE1	0.86±0.01	0.86±0.01	0.86±0.01	**0.87**±0.01	**0.87**±0.01
SDAE2	0.86±0.01	0.86±0.01	0.86±0.01	0.86±0.01	**0.87**±0.01
SDAE3	0.86±0.01	0.86±0.02	0.85±0.02	0.86±0.01	0.86±0.01
RBM1	0.74±0.03	0.73±0.03	0.71±0.08	0.74±0.04	0.74±0.04
RBM2	0.71±0.09	0.70±0.09	0.63±0.09	0.60±0.11	0.59±0.10
RBM3	0.39±0.10	0.39±0.10	0.39±0.10	0.34±0.09	0.34±0.09
PGSB + REPBASE					
MLP1	0.85±0.02	0.85±0.02	0.85±0.02	0.86±0.02	0.86±0.02
MLP2	0.85±0.02	0.85±0.01	0.85±0.01	0.86±0.02	0.86±0.02
MLP3	0.85±0.02	0.84±0.01	0.84±0.01	0.85±0.02	0.86±0.02
SDAE1	0.86±0.02	0.85±0.02	0.85±0.02	**0.87**±0.02	**0.87**±0.02
SDAE2	0.86±0.02	0.85±0.02	0.86±0.02	**0.87**±0.02	0.86±0.02
SDAE3	0.86±0.02	0.86±0.02	0.86±0.02	**0.87**±0.02	**0.87**±0.02
RBM1	0.75±0.02	0.75±0.02	0.72±0.07	0.75±0.03	0.75±0.03
RBM2	0.62±0.06	0.60±0.04	0.54±0.11	0.59±0.02	0.59±0.02
RBM3	0.48±0.13	0.46±0.14	0.51±0.13	0.47±0.15	0.47±0.15

Todavia, a sua inferioridade estatística não é muito agravada em valores, pois em grande parte as redes MLP2 e MLP3 estão atrás por 1% ou 2%. O mesmo não se aplica as RBMs, uma vez que estas atingem valores mínimos de desempenho em algumas situações como 39% no conjunto de dados REPBASE. Esta deterioração de resultados é surpreendente visto que os Stacked Denoising Auto-Encoders (SDA) partem de um pressuposto similar, e, mesmo assim, obtiveram resultados estatisticamente superiores.

A hipótese a respeito da relativa falha das RBMs reside em dois aspectos: aproximações e *Overfitting*. Como descrito com mais detalhes na Seção 3.2.1, o algoritmo *Constrastive Divergence* utiliza a amostragem de Gibbs, uma aproximação da distribuição da rede, para atualizar os pesos. Esta propriedade não está presente nas outras redes, uma vez que MLPs e SDAs utilizam o gradiente descendente. Além disso, existe um decaimento notável quando a rede se torna mais profunda. Ao contrário do esperado, a adição de camadas ocultas, neste caso, não proporcionou mais generalização, mas sim *Overfitting*. O mesmo não acontece com outras redes devido ao *Dropout* e a camada de ruído extra da MLP e SDA respectivamente. Apesar do fraco desempenho, não deve-se desconsiderar completamente RBMs. Assim como nas SDAs, a arquitetura e algoritmos usados nesta pesquisa são suas primeiras versões e trabalhos recentes disponibilizam redes mais modernas neste contexto.

A superioridade estatística dos SDAs profundos quando comparados a MLPs, apesar de pequena, ressalta o potencial da inicialização dos pesos. No caso das MLPs, uma inicialização padrão foi utilizada. SDAs, por outro lado, usam o pré-treino não supervisionado. Este, segundo os resultados, favoreceu o desempenho das SDA, uma vez que MLPs utilizam praticamente a mesma configuração e o seu *Dropout* simula a camada de ruído.

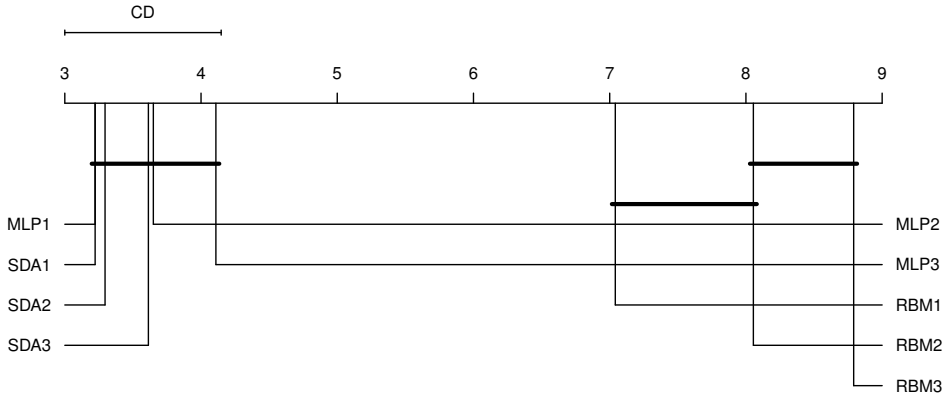


Figura 7.7: Teste de Nemenyi - Comparando Deep Learning com Machine Learning - Nível 2

Apesar dos valores ligeiramente superiores, nos níveis 1 e 2, (Tabelas A.1, A.2, A.3, e Figuras A.1, 7.7), não existe superioridade estatística. Todas as versões de MLPs e SDAs, quando comparadas entre si, apresentam a mesma relevância estatística, porém, quando RBMs são comparadas, estas estão constantemente ranqueadas em posições inferiores.

Devido a grande quantidade de instâncias disponíveis, os resultados no primeiro nível foram consideravelmente altos, alcançando valores superiores a 90% em quase todos os casos, com as SDAs e MLPs. As RBMs entretanto, especialmente com 2 e 3 camadas, podem obter valores indesejáveis.

Da mesma maneira, no segundo nível, foram obtidos resultados satisfatórios em muitas situações, principalmente nos conjuntos de dados de EC e TEs. Em contrapartida, em alguns casos, como GPCRpfam e GPCRProsite, os melhores desempenhos foram de 70% e 63%.

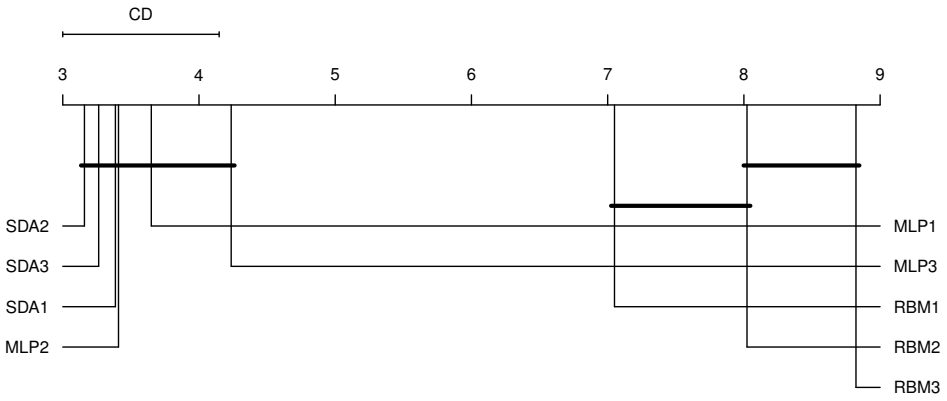


Figura 7.8: Teste de Nemenyi - Comparando Deep Learning com Machine Learning - Nível 3

Nos níveis 3 e 4 das hierarquias de TEs e proteínas, comportamentos idênticos aos níveis anteriores foram obtidos. Como pode ser observado nas Figuras 7.8 e 7.9, e Tabelas A.7, A.8, A.9, A.10, A.11 e A.12, as redes MLP e SDA, independentemente da quantidade de camadas ocultas, foram estatisticamente superiores às RBMs.

No terceiro nível, em especial nos conjuntos de dados de TEs e EC, as redes ainda mantiveram resultados relativamente altos. Nos TEs, as redes MLP e SDA apresentaram valores acima de 80% em todas as estratégias. Por sua vez, as mesmas redes produziram resultados na faixa de 90% para as proteínas EC. Por outro lado, nos conjuntos de dados GPCR, houve situações em que o melhor valor foi meramente superior a aleatoriedade, no caso dos conjuntos GPCRPfam e GPCRProsite, os melhores valores foram de 55% e 50% respectivamente. Este desempenho ressalta o aumento da dificuldade em níveis mais profundos da hierarquia, uma vez que a quantidade de dados diminui consideravelmente. Este é um problema intrínseco em Classificação Hierárquica, e já foi documentado em outros estudos (CERRI et al., 2016).

Mais uma vez, circunstancias similares são encontradas no quarto nível. No geral, as redes MLP e SDA mostraram resultados superiores, porém, em alguns poucos casos, RBMs com uma camada oculta também se mostraram viáveis. De maneira inesperada, os resultados nos conjuntos de dados de TEs e EC se preservaram altos, por volta de 80% e 90%. Por sua vez, os resultados dos conjuntos de GPCRPfam e GPCRProsite foram consideravelmente baixos. Este comportamento pode ser explicado pela pequena quantidade de atributos e instâncias quando comparados aos TEs e proteínas.

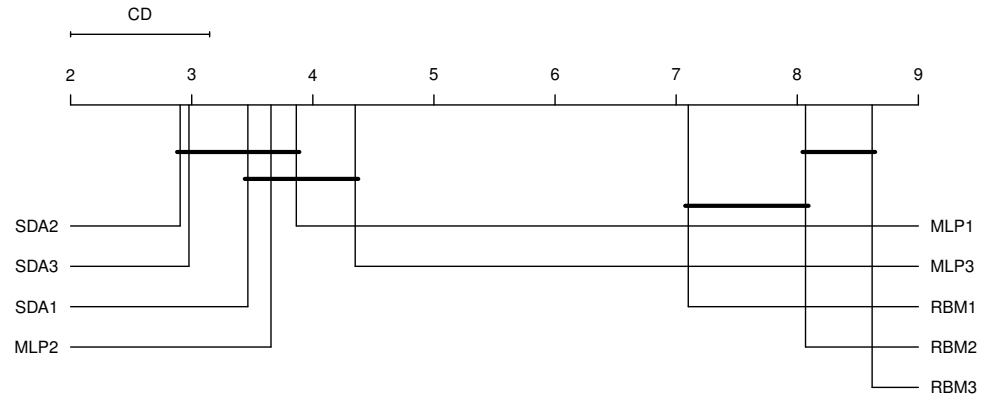


Figura 7.9: Teste de Nemenyi - Comparando Deep Learning com Machine Learning - Nível 4

Tabela 7.4: Resultados nos conjuntos de dados GPCR

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			GPCRPfam		
MLP1	0.57±0.01	0.57±0.01	0.57±0.01	0.56±0.01	0.53±0.01
MLP2	0.57±0.01	0.57±0.01	0.57±0.01	0.58±0.01	0.44±0.03
MLP3	0.57±0.01	0.57±0.01	0.57±0.01	0.58±0.01	0.42±0.02
SDA1	0.57±0.01	0.57±0.01	0.57±0.01	0.56±0.01	0.53±0.01
SDA2	0.58±0.01	0.58±0.01	0.58±0.01	**0.59**±0.01	0.43±0.03
SDA3	0.58±0.01	0.58±0.01	0.58±0.01	**0.59**±0.01	0.41±0.04
RBM1	0.51±0.01	0.51±0.01	0.51±0.01	0.56±0.01	0.53±0.01
RBM2	0.49±0.04	0.49±0.04	0.49±0.04	0.51±0.01	0.42±0.03
RBM3	0.49±0.05	0.49±0.05	0.49±0.05	0.48±0.04	0.45±0.05
			GPCRPrints		
MLP1	0.76±0.01	0.76±0.01	0.76±0.01	0.77±0.01	0.78±0.01
MLP2	0.76±0.01	0.76±0.01	0.76±0.01	0.78±0.01	0.78±0.01
MLP3	0.76±0.01	0.76±0.01	0.76±0.01	0.78±0.01	0.75±0.06
SDA1	0.76±0.01	0.76±0.01	0.76±0.01	0.77±0.01	0.78±0.01
SDA2	0.77±0.01	0.77±0.01	0.77±0.01	0.78±0.01	**0.79**±0.01
SDA3	0.76±0.01	0.76±0.01	0.76±0.01	0.79±0.01	0.77±0.04
RBM1	0.72±0.01	0.72±0.01	0.72±0.01	0.77±0.01	0.78±0.01
RBM2	0.68±0.02	0.68±0.02	0.68±0.02	0.71±0.02	0.71±0.06
RBM3	0.69±0.02	0.69±0.02	0.69±0.02	0.63±0.03	0.63±0.04
			GPCRProsite		
MLP1	0.57±0.01	0.57±0.01	0.57±0.01	0.57±0.01	0.48±0.03
MLP2	0.57±0.01	0.57±0.01	0.57±0.01	0.58±0.01	0.46±0.04
MLP3	0.57±0.01	0.57±0.01	0.57±0.01	0.57±0.01	0.47±0.06
SDA1	0.58±0.01	0.58±0.01	0.58±0.01	0.57±0.01	0.48±0.03
SDA2	0.57±0.01	0.57±0.01	0.57±0.01	0.58±0.01	0.46±0.04
SDA3	0.57±0.02	0.57±0.02	0.57±0.02	**0.59**±0.02	0.44±0.03
RBM1	0.49±0.01	0.49±0.01	0.49±0.01	0.57±0.01	0.48±0.03
RBM2	0.48±0.04	0.48±0.04	0.48±0.04	0.50±0.01	0.34±0.05
RBM3	0.26±0.14	0.26±0.14	0.26±0.14	0.28±0.17	0.28±0.17
			GPCRInterpro		
MLP1	0.79±0.01	0.79±0.01	0.79±0.01	0.80±0.01	0.78±0.02
MLP2	0.79±0.01	0.79±0.01	0.79±0.01	**0.81**±0.01	0.78±0.02
MLP3	0.79±0.00	0.79±0.00	0.79±0.00	**0.81**±0.01	0.75±0.05
SDA1	0.79±0.01	0.79±0.01	0.79±0.01	0.80±0.01	0.78±0.02
SDA2	0.79±0.01	0.79±0.01	0.79±0.01	**0.81**±0.01	0.79±0.02
SDA3	0.79±0.01	0.79±0.01	0.79±0.01	**0.81**±0.01	0.77±0.04
RBM1	0.75±0.02	0.75±0.02	0.75±0.02	0.80±0.01	0.78±0.02
RBM2	0.67±0.05	0.67±0.05	0.67±0.05	0.66±0.02	0.65±0.03
RBM3	0.35±0.02	0.35±0.02	0.35±0.02	0.52±0.07	0.51±0.07

Tabela 7.5: Resultados nos conjuntos de dados EC

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			ECPfam		
MLP1	0.93±0.00	0.93±0.00	0.93±0.00	**0.97**±0.00	**0.97**±0.00
MLP2	0.93±0.00	0.93±0.00	0.93±0.00	**0.97**±0.00	**0.97**±0.00
MLP3	0.93±0.00	0.93±0.00	0.93±0.00	**0.97**±0.00	**0.97**±0.00
SDA1	0.93±0.00	0.93±0.00	0.93±0.00	**0.97**±0.00	**0.97**±0.00
SDA2	0.93±0.00	0.93±0.00	0.93±0.00	**0.97**±0.00	**0.97**±0.00
SDA3	0.93±0.00	0.93±0.00	0.93±0.00	**0.97**±0.00	**0.97**±0.00
RBM1	0.91±0.00	0.91±0.00	0.91±0.00	**0.97**±0.00	**0.97**±0.00
RBM2	0.72±0.19	0.72±0.19	0.72±0.19	0.30±0.13	0.44±0.13
RBM3	0.71±0.22	0.71±0.22	0.71±0.22	0.18±0.09	0.17±0.09
			ECPrints		
MLP1	0.92±0.00	0.92±0.00	0.92±0.00	**0.95**±0.00	0.94±0.00
MLP2	0.92±0.00	0.92±0.00	0.92±0.00	**0.95**±0.00	**0.95**±0.00
MLP3	0.91±0.00	0.91±0.00	0.91±0.00	**0.95**±0.00	**0.95**±0.00
SDA1	0.92±0.00	0.92±0.00	0.92±0.00	**0.95**±0.00	0.94±0.00
SDA2	0.92±0.00	0.92±0.00	0.92±0.00	**0.95**±0.00	**0.95**±0.00
SDA3	0.92±0.00	0.92±0.00	0.92±0.00	**0.95**±0.00	**0.95**±0.00
RBM1	0.88±0.01	0.88±0.01	0.88±0.01	**0.95**±0.00	0.94±0.00
RBM2	0.14±0.03	0.14±0.03	0.14±0.03	0.28±0.00	0.63±0.04
RBM3	0.14±0.03	0.14±0.03	0.14±0.03	0.21±0.02	0.37±0.05
			ECProsite		
MLP1	0.93±0.00	0.93±0.00	0.93±0.00	**0.96**±0.00	**0.96**±0.00
MLP2	0.93±0.00	0.93±0.00	0.93±0.00	**0.96**±0.00	0.96±0.01
MLP3	0.92±0.00	0.92±0.00	0.92±0.00	**0.96**±0.00	**0.96**±0.00
SDA1	0.93±0.00	0.93±0.00	0.93±0.00	**0.96**±0.00	**0.96**±0.00
SDA2	0.93±0.01	0.93±0.01	0.93±0.01	**0.96**±0.00	**0.96**±0.00
SDA3	0.93±0.00	0.93±0.00	0.93±0.00	**0.96**±0.00	**0.96**±0.00
RBM1	0.92±0.00	0.92±0.00	0.92±0.00	**0.96**±0.00	**0.96**±0.00
RBM2	0.19±0.00	0.19±0.00	0.19±0.00	0.29±0.01	0.62±0.02
RBM3	0.19±0.00	0.19±0.00	0.19±0.00	0.16±0.01	0.16±0.01
			ECInterpro		
MLP1	0.94±0.00	0.94±0.00	0.94±0.00	**0.98**±0.00	**0.98**±0.00
MLP2	0.94±0.00	0.94±0.00	0.94±0.00	**0.98**±0.00	**0.98**±0.00
MLP3	0.94±0.00	0.94±0.00	0.94±0.00	**0.98**±0.00	**0.98**±0.00
SDA1	0.94±0.00	0.94±0.00	0.94±0.00	**0.98**±0.00	**0.98**±0.00
SDA2	0.93±0.00	0.93±0.00	0.93±0.00	**0.98**±0.00	**0.98**±0.00
SDA3	0.93±0.00	0.93±0.00	0.93±0.00	**0.98**±0.00	**0.98**±0.00
RBM1	0.93±0.00	0.93±0.00	0.93±0.00	**0.98**±0.00	**0.98**±0.00
RBM2	0.92±0.01	0.92±0.01	0.92±0.01	0.36±0.09	0.36±0.09
RBM3	0.65±0.27	0.65±0.27	0.65±0.27	0.22±0.11	0.22±0.11

Capítulo 8

CONCLUSÃO

Este capítulo contém as considerações finais sobre esta pesquisa, apresentando detalhes sobre o que pode ser concluído, contribuições acadêmicas e, por fim, ideias a serem exploradas em trabalhos futuros.

Neste mestrado, foi proposta a classificação de Elementos Transponíveis (TEs) como um problema de Classificação Hierárquica (CH), mais precisamente, a abordagem Local foi investigada utilizando Redes Neurais Artificiais (RNAs) do paradigma *Deep Learning*.

Considerando as especifidades dos TEs, duas novas estratégias baseadas na estratégia Classificador Local por Nó Pai (LCPN) foram propostas: Classificador Local por Nó Pai não Folha (nlLCPN) e Classificador Local Por Nó Pai e Ramo (LCPNB). Estas permitem classificações *Não obrigatória* automaticamente, e, no caso da LCPNB, a minimização da propagação de erros,

Para tanto, ambas são idênticas em suas fases de treino, adicionando um nó fictício para cada classificador associado a um nó pai que também é uma classificação *Não obrigatória*. Desta maneira, estes classificadores aprendem a distinguir entre seis nós associados e seus respectivos nós filho. A diferença está na fase de teste. Na primeira estratégia, nlLCPN, uma nova instância é classificada da mesma maneira que a estratégia LCPN. Assim, recursivamente, a nova instância é testada por nós pais até que um nó folha seja atingido, porém caso um classificador prediga a classe associada a si mesmo, o teste se encerra e uma classificação *Não obrigatória* é obtida. Todavia, teoricamente, esta estratégia propaga erros, pois classificações incorretas em níveis superiores podem levar a predições totalmente incorretas.

Por sua vez, a segunda estratégia utiliza todos os classificadores da hierarquia. Nesta estratégia, a instância é testada por todos os classificadores para obter predições de probabilidade. Em seguida, calcula-se a média das probabilidades até todas as predições possíveis, e a

classificação com maior média é dada como final. Desta maneira, mesmo se os classificadores errarem, existe a possibilidade de correção.

Ao realizar experimentos utilizando as RNAs *Denoising Auto-Encoder* ,*Restricted Boltzmann Machine* e Deep MLP como classificadores locais, pode-se notar que ambas as estratégias são capazes de produzir resultados competitivos ou superiores aos métodos da literatura.

No aspecto geral, o desempenho, medido através da *Hierarchical F-Measure*, mostra que as redes profundas apresentam resultados superiores a suas versões rasas. Ao estender essa análise para uma comparação nível a nível, um comportamento diferente é perceptível. Nos dois primeiros níveis da hierarquia, as RNAs apresentam desempenho muito próximo ao perfeito, porém o resultado se deteriorou significativamente no terceiro e quarto níveis. Isto é um fenômeno recorrente em problemas de CH, visto que a quantidade de exemplos para treino é menor em níveis profundos. Desta maneira, neste trabalho, a classificação de TEs é consolidada como um problema de CH. Concomitantemente uma análise sobre como redes profundas se comportam em problemas de classificação hierárquica também é fornecida.

8.1 Contribuições

As principais contribuições desta pesquisa são sumarizadas em três tópicos: Classificação Hierárquica de TEs, *Deep Learning* em problemas de Classificação Hierárquica, e Projetos Paralelos.

Como já mencionado, a classificação de TEs é realizada normalmente utilizando homologia, porém os experimentos desta pesquisa mostram que métodos de Classificação Hierárquica apresentam resultados superiores. Durante os experimentos necessários para esta contribuição, redes neurais com apenas uma camada oculta foram comparadas com o método de homologia BLASTn, resultando na seguinte publicação:

- Nakano, Felipe Kenji, et al. “Top-down strategies for hierarchical classification of transposable elements with neural networks”. IEEE International Joint Conference on Neural Networks (IJCNN), 2017.

Como continuidade, métodos de *Deep Learning* foram implementados e comparados com redes neurais rasas. Esta análise mostrou que, ao adicionar camadas, os resultados tendem a ser superiores, resultando na publicação:

- Nakano, Felipe Kenji, et al. “Improving Hierarchical Classification of Transposable Ele-

ments using Deep Neural Networks". IEEE International Joint Conference on Neural Networks (IJCNN), 2018.

Adicionalmente, para auxiliar o desenvolvimento desta pesquisa, uma ferramenta para pre-processamento de dados especifica para classificação hierárquica foi criada. Esta é capaz de extrair características de sequências de DNA e mapeá-las para uma hierarquia. Todas as configurações são totalmente flexíveis, permitindo sua aplicabilidade em vários problemas. O manuscrito desta contribuição está prestes a ser submetido a um periódico.

- Nakano, Felipe Kenji, et al. "Machine Learning Datasets for Hierarchical Classification of Transposable Elements". Ainda em avaliação.

Por fim, a última contribuição apresenta uma versão estendida dos trabalhos anteriores no qual uma análise com mais dados é realizada. Ambos os experimentos e manuscrito estão em desenvolvimento, com previsão de submissão após a finalização do mestrado.

- Nakano, Felipe Kenji, et al. "Hierarchical Classification of Transposable Elements using Deep Neural Networks". Em desenvolvimento.

Além das contribuições anteriores, projetos paralelos também resultaram em contribuições. A lista a seguir apresenta todos os trabalhos publicados que são relacionados indiretamente ao mestrado.

- Nakano, Felipe Kenji, and Ricardo Cerri. "Denoising Auto-Encoders as Feature Extractors in Hierarchical Classification Problems". Encontro Nacional de Inteligência Artificial e Computacional (ENIAC) 2017 (NAKANO; CERRI, 2017);
- Kenji Nakano, Felipe, et al. "Stacking Methods for Hierarchical Classification". IEEE International Conference on Machine Learning and Applications (ICMLA), 2017 (NAKANO et al., 2017b);
- Mastelini, Saulo Martiello, et al. "Multi-Output Tree Chaining: An Interpretative Modelling and Lightweight Multi-Target Approach". Journal of Signal Processing Systems (2018): 1-25 (MASTELINI et al., 2018);
- Santos et. al. "Strategies for Selection of Positive and Negative Instances in the Hierarchical Classification of Transposable Elements". Aceito para publicação no Brazilian Conference on Intelligent Systems (BRACIS) 2018.

Anteriormente à utilização de *Deep Learning* para Classificação Hierárquica, suas capacidades representativas foram testadas, resultando em uma publicação no ENIAC 2017, "Denoising Auto-Encoders as Feature Extractors in Hierarchical Classification Problems". Nesse artigo, as representações aprendidas por *Denoising Auto-Encoders* (DAE) são utilizadas como atributos para o classificador K-Vizinhos Mais Próximos. O resultado mostrou que DAEs são capazes de extrair representações mínimas, porém significativas.

Após realizar a revisão bibliográfica, notou-se a ausência de estratégias de Classificação Hierárquica que explorassem o método *ensemble* Stacking. Utilizando uma ideia similar ao trabalho de Cerri (CERRI et al., 2016), o artigo "Stacking Methods for Hierarchical Classification" propôs utilizar probabilidades de predições como atributos extras. Os resultados mostraram que os atributos extras favorecem os resultados em grande parte dos casos.

No trabalho "Multi-output Tree Chaining: a interpretative modelling and lightweight multi-target approach", uma nova abordagem é proposta, buscando construir uma hierarquia de maneira que as dependências entre os *targets* fossem mapeadas automaticamente. Ao testar esta abordagem em problemas *Multi-Output*, resultados satisfatórios foram atingidos, e, consequentemente, publicados.

No trabalho realizado durante o estágio de pesquisa no exterior (BEPE FAPESP) na Bélgica, foi desenvolvido um módulo de *Active Learning* para Classificação Hierárquica dentro do *framework* Clus (VENS et al., 2008). Esse é um tema inédito, considerando que seu foco consiste em problemas hierárquicos multirrótulo. O manuscrito, assim como o manual desta contribuição, estão em fase final de desenvolvimento.

O último trabalho, "Strategies for Selection of Positive and Negative Instances in the Hierarchical Classification of Transposable Elements", investiga e compara diversos métodos da literatura para seleção de exemplos positivas e negativas para a estratégia Classificador Local por Nó. Os resultados mostram que utilizar o melhor método influencia consideravelmente o desempenho dos classificadores.

8.2 Trabalhos Futuros

Apesar dos resultados satisfatórios, notou-se que, em alguns casos, a estratégia LCPNB não é superior a nlLCPN. Isto é contraditório, pois espera-se que a LCPNB apresente resultados superiores devido ao seu método de redução de propagação de erros. A suspeita está nos nós pais que apresentam poucos nós filhos, pois é muito provável que predições erradas apresentem probabilidades altas. Desta maneira, no futuro, pretende-se explorar outras maneiras de definir

a classificação final.

Adicionalmente, os experimentos mostraram que a quantidade reduzida de dados, assim como seu desbalanceamento, pode prejudicar o desempenho dos classificadores, principalmente em níveis mais profundos. Consequentemente, métodos que amenizem tais problemas são promissores. Ainda, no contexto de classificação hierárquica, poucos trabalhos abordam este problema, possibilitando uma nova área de pesquisa.

Do ponto de vista das RNAs, outras arquiteturas também devem ser testadas. Particularmente, a rede *Long Short Term Memory* chama a atenção devido a sua capacidade de aprender a partir de dados sequenciais (HOCHREITER; SCHMIDHUBER, 1997). No contexto desta pesquisa, sequências de DNA poderiam ser utilizadas diretamente para o seu treino, assim evitando a extração de atributos.

Da mesma maneira, neste trabalho, nenhuma das arquiteturas foi otimizada devido a quantidade exponencial de combinações. No futuro, pesquisas poderiam ter como foco a otimização de parâmetros em problemas de classificação hierárquica.

Apendice A

RESULTADOS POR NÍVEL

Como mencionado no Capítulo 7, os resultados por nível de todos os conjuntos de dados são apresentados a seguir. Novamente, nota-se que existe uma grande variação nos resultados a medida que níveis mais profundos são investigados, os resultados tendem a decair, ressaltando sua maior dificuldade.

Do ponto de vista do Aprendizado de Máquina, a dificuldade está na grande quantidade de classes associada a pequena quantidade de instâncias, fato que, muitas vezes, acarreta na falta de generalização, resultando em classificadores propícios a *Overfitting*. Da mesma maneira, o desbalanceamento de classes também contribui negativamente para o desempenho dos classificadores.

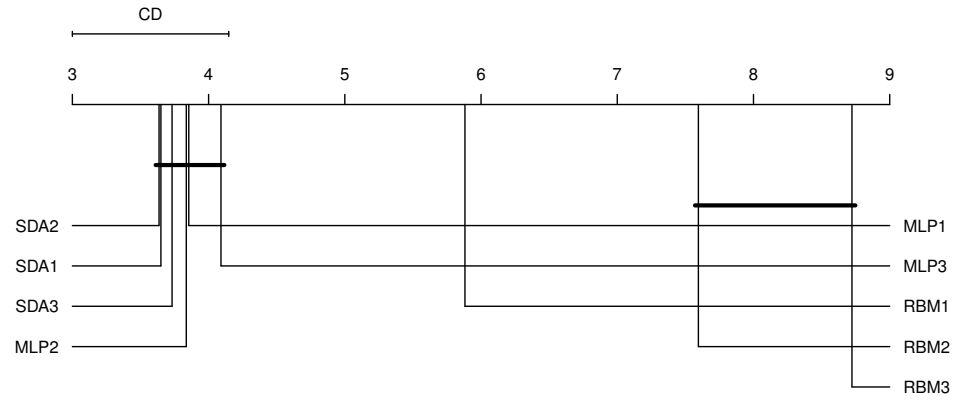


Figura A.1: Teste de Nemenyi - Comparando Deep Learning com Machine Learning - Nível 1

Tabela A.1: Resultados nos conjuntos de dados TEs - Nível 1

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
PGSB					
MLP1	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	0.95±0.02	**0.96**±0.02
MLP2	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	0.95±0.02
MLP3	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	0.95±0.02	0.95±0.02
SDA1	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	0.95±0.02	**0.96**±0.02
SDA2	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02
SDA3	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	0.96±0.02	**0.96**±0.02
RBM1	0.92±0.03	0.92±0.03	0.92±0.03	0.95±0.02	**0.96**±0.02
RBM2	0.86±0.0	0.86±0.0	0.86±0.0	0.86±0.0	0.86±0.0
RBM3	0.64±0.33	0.64±0.33	0.64±0.33	0.72±0.29	0.72±0.29
REPBASE					
MLP1	0.95±0.01	0.95±0.01	0.95±0.01	0.95±0.01	0.95±0.01
MLP2	0.95±0.01	0.95±0.01	0.95±0.01	0.94±0.01	0.95±0.01
MLP3	0.95±0.01	0.95±0.01	0.95±0.01	0.94±0.01	0.94±0.01
SDA1	**0.96**±0.01	**0.96**±0.01	**0.96**±0.01	0.95±0.01	0.95±0.01
SDA2	**0.96**±0.01	**0.96**±0.01	**0.96**±0.01	0.95±0.01	0.95±0.01
SDA3	**0.96**±0.01	**0.96**±0.01	**0.96**±0.01	0.95±0.01	0.95±0.01
RBM1	0.87±0.03	0.87±0.03	0.87±0.03	0.95±0.01	0.95±0.01
RBM2	0.85±0.07	0.85±0.07	0.85±0.07	0.79±0.11	0.78±0.11
RBM3	0.56±0.14	0.56±0.14	0.56±0.14	0.5±0.15	0.5±0.15
PGSB + REPBASE					
MLP1	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01
MLP2	**0.95**±0.01	**0.95**±0.01	0.95±0.01	0.94±0.01	0.94±0.01
MLP3	**0.95**±0.01	**0.95**±0.01	0.95±0.01	0.94±0.01	0.94±0.01
SDA1	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	0.94±0.01	0.94±0.01
SDA2	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	0.94±0.01	0.94±0.01
SDA3	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01
RBM1	0.87±0.01	0.87±0.01	0.87±0.01	0.94±0.01	0.94±0.01
RBM2	0.74±0.05	0.74±0.05	0.74±0.05	0.72±0.0	0.72±0.0
RBM3	0.63±0.18	0.63±0.18	0.63±0.18	0.68±0.13	0.68±0.13

Tabela A.2: Resultados nos conjuntos de dados GPCR - Nível 1

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			GPCRPfam		
MLP1	**0.93**±0.0	**0.93**±0.0	**0.93**±0.0	**0.93**±0.0	0.86±0.01
MLP2	**0.93**±0.0	**0.93**±0.0	**0.93**±0.0	0.93±0.01	0.84±0.02
MLP3	**0.93**±0.0	**0.93**±0.0	**0.93**±0.0	0.93±0.01	0.86±0.05
SDA1	0.93±0.01	0.93±0.01	0.93±0.01	0.93±0.0	0.86±0.01
SDA2	**0.93**±0.0	0.93±0.0	**0.93**±0.0	0.93±0.01	0.82±0.04
SDA3	0.93±0.01	0.93±0.01	0.93±0.01	0.93±0.01	0.75±0.08
RBM1	0.91±0.01	0.91±0.01	0.91±0.01	**0.93**±0.0	0.86±0.01
RBM2	0.9±0.01	0.9±0.01	0.9±0.01	0.9±0.01	0.88±0.03
RBM3	0.9±0.01	0.9±0.01	0.9±0.01	0.91±0.01	0.87±0.02
			GPCRPrints		
MLP1	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	0.88±0.01
MLP2	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	0.89±0.01
MLP3	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	0.89±0.02
SDA1	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	0.88±0.01
SDA2	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	0.91±0.01
SDA3	0.91±0.01	0.91±0.01	0.91±0.01	**0.92**±0.01	0.91±0.01
RBM1	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	**0.92**±0.01	0.88±0.01
RBM2	0.87±0.02	0.87±0.02	0.87±0.02	0.87±0.02	0.86±0.02
RBM3	0.87±0.03	0.87±0.03	0.87±0.03	0.87±0.02	0.87±0.02
			GPCRProsite		
MLP1	0.86±0.01	0.86±0.01	0.86±0.01	0.86±0.01	0.7±0.04
MLP2	0.86±0.01	0.86±0.01	0.86±0.01	**0.87**±0.01	0.74±0.04
MLP3	0.86±0.01	0.86±0.01	0.86±0.01	0.86±0.01	0.75±0.1
SDA1	0.86±0.01	0.86±0.01	0.86±0.01	0.86±0.01	0.7±0.04
SDA2	0.86±0.01	0.86±0.01	0.86±0.01	**0.87**±0.01	0.73±0.06
SDA3	0.86±0.01	0.86±0.01	0.86±0.01	**0.87**±0.01	0.68±0.06
RBM1	0.84±0.0	0.84±0.0	0.84±0.0	0.86±0.01	0.7±0.04
RBM2	0.84±0.01	0.84±0.01	0.84±0.01	0.84±0.01	0.77±0.16
RBM3	0.6±0.35	0.6±0.35	0.6±0.35	0.54±0.34	0.53±0.34
			GPCRInterpro		
MLP1	0.92±0.0	0.92±0.0	0.92±0.0	**0.93**±0.01	0.91±0.03
MLP2	0.92±0.0	0.92±0.0	0.92±0.0	**0.93**±0.01	0.91±0.03
MLP3	0.92±0.01	0.92±0.01	0.92±0.01	**0.93**±0.01	0.9±0.02
SDA1	0.92±0.01	0.92±0.01	0.92±0.01	**0.93**±0.01	0.91±0.03
SDA2	0.92±0.01	0.92±0.01	0.92±0.01	**0.93**±0.01	0.91±0.02
SDA3	0.92±0.01	0.92±0.01	0.92±0.01	**0.93**±0.01	0.91±0.02
RBM1	0.88±0.03	0.88±0.03	0.88±0.03	**0.93**±0.01	0.91±0.03
RBM2	0.84±0.05	0.84±0.05	0.84±0.05	0.88±0.02	0.86±0.03
RBM3	0.82±0.02	0.82±0.02	0.82±0.02	0.84±0.03	0.83±0.04

Tabela A.3: Resultados nos conjuntos de dados EC - Nível 1

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			ECPfam		
MLP1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
MLP2	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
MLP3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA2	0.98±0.01	0.98±0.01	0.98±0.01	**0.99**±0.0	**0.99**±0.0
SDA3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
RBM1	0.98±0.0	0.98±0.0	0.98±0.0	0.99±0.0	0.99±0.0
RBM2	0.83±0.17	0.83±0.17	0.83±0.17	0.4±0.19	0.64±0.14
RBM3	0.84±0.18	0.84±0.18	0.84±0.18	0.39±0.2	0.38±0.2
			ECPrints		
MLP1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	0.98±0.01
MLP2	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	0.98±0.0
MLP3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	0.98±0.01	0.98±0.0
SDA1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	0.98±0.01
SDA2	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
RBM1	0.98±0.0	0.98±0.0	0.98±0.0	0.99±0.0	0.98±0.01
RBM2	0.3±0.0	0.3±0.0	0.3±0.0	0.3±0.0	0.66±0.04
RBM3	0.3±0.0	0.3±0.0	0.3±0.0	0.3±0.0	0.52±0.06
			ECProsite		
MLP1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
MLP2	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
MLP3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA2	0.99±0.01	0.99±0.01	0.99±0.01	**0.99**±0.0	**0.99**±0.0
SDA3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
RBM1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
RBM2	0.35±0.0	0.35±0.0	0.35±0.0	0.35±0.0	0.73±0.03
RBM3	0.35±0.0	0.35±0.0	0.35±0.0	0.35±0.0	0.35±0.0
			ECInterpro		
MLP1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
MLP2	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
MLP3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA2	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
SDA3	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
RBM1	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0	**0.99**±0.0
RBM2	0.98±0.01	0.98±0.01	0.98±0.01	0.84±0.21	0.84±0.21
RBM3	0.76±0.23	0.76±0.23	0.76±0.23	0.58±0.27	0.58±0.27

Tabela A.4: Resultados dos conjuntos de dados de TEs - Nível 2

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
PGSB					
MLP1	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02
MLP2	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02
MLP3	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02
SDA1	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02
SDA2	**0.96**±0.02	**0.96**±0.02	**0.96**±0.02	0.95±0.02	0.95±0.02
SDA3	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02	0.95±0.02
RBM1	0.9±0.04	0.9±0.04	0.9±0.04	0.95±0.02	0.95±0.02
RBM2	0.84±0.0	0.84±0.0	0.84±0.0	0.84±0.0	0.84±0.0
RBM3	0.59±0.32	0.59±0.32	0.59±0.32	0.7±0.28	0.7±0.28
REPBASE					
MLP1	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01	0.94±0.01
MLP2	0.94±0.01	0.94±0.01	0.94±0.01	0.93±0.01	0.94±0.01
MLP3	0.94±0.01	0.94±0.01	0.94±0.01	0.93±0.01	0.93±0.01
SDA1	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	0.94±0.01	0.94±0.01
SDA2	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	0.94±0.01	0.94±0.01
SDA3	**0.95**±0.01	**0.95**±0.01	**0.95**±0.01	0.94±0.01	0.94±0.01
RBM1	0.85±0.03	0.85±0.03	0.85±0.03	0.94±0.01	0.94±0.01
RBM2	0.82±0.08	0.82±0.08	0.82±0.08	0.74±0.12	0.73±0.11
RBM3	0.5±0.12	0.5±0.12	0.5±0.12	0.45±0.12	0.45±0.12
PGSB + REPBASE					
MLP1	0.93±0.01	0.93±0.01	0.93±0.01	0.93±0.01	0.93±0.01
MLP2	**0.94**±0.01	**0.94**±0.01	**0.94**±0.01	0.93±0.01	0.93±0.01
MLP3	**0.94**±0.01	**0.94**±0.01	**0.94**±0.01	0.92±0.01	0.93±0.02
SDA1	**0.94**±0.01	**0.94**±0.01	**0.94**±0.01	0.93±0.01	0.93±0.01
SDA2	0.94±0.02	0.94±0.02	0.94±0.02	0.93±0.01	0.93±0.01
SDA3	**0.94**±0.01	**0.94**±0.01	**0.94**±0.01	**0.94**±0.01	**0.94**±0.01
RBM1	0.85±0.02	0.85±0.02	0.85±0.02	0.93±0.01	0.93±0.01
RBM2	0.71±0.05	0.71±0.05	0.71±0.05	0.68±0.0	0.68±0.0
RBM3	0.6±0.16	0.6±0.16	0.6±0.16	0.58±0.15	0.58±0.15

Tabela A.5: Resultados dos conjuntos de dados de GPCR - Nível 2

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			GPCRPfam		
MLP1	0.77±0.01	0.77±0.01	0.77±0.01	0.77±0.01	0.7±0.01
MLP2	0.77±0.01	0.77±0.01	0.77±0.01	**0.78**±0.01	0.58±0.04
MLP3	0.77±0.01	0.77±0.01	0.77±0.01	0.77±0.01	0.56±0.03
SDA1	0.77±0.01	0.77±0.01	0.77±0.01	0.77±0.01	0.7±0.01
SDA2	0.77±0.01	0.77±0.01	0.77±0.01	**0.78**±0.01	0.56±0.04
SDA3	0.76±0.01	0.76±0.01	0.76±0.01	**0.78**±0.01	0.54±0.06
RBM1	0.69±0.01	0.69±0.01	0.69±0.01	0.77±0.01	0.7±0.01
RBM2	0.68±0.04	0.68±0.04	0.68±0.04	0.69±0.01	0.53±0.02
RBM3	0.7±0.05	0.7±0.05	0.7±0.05	0.66±0.03	0.63±0.04
			GPCRPrints		
MLP1	0.87±0.01	0.87±0.01	0.87±0.01	**0.88**±0.01	0.86±0.01
MLP2	0.87±0.01	0.87±0.01	0.87±0.01	**0.88**±0.01	0.86±0.01
MLP3	0.87±0.01	0.87±0.01	0.87±0.01	**0.88**±0.01	0.83±0.06
SDA1	0.87±0.01	0.87±0.01	0.87±0.01	**0.88**±0.01	0.86±0.01
SDA2	0.87±0.01	0.87±0.01	0.87±0.01	**0.88**±0.01	0.88±0.01
SDA3	0.86±0.01	0.86±0.01	0.86±0.01	**0.88**±0.01	0.86±0.04
RBM1	0.84±0.01	0.84±0.01	0.84±0.01	**0.88**±0.01	0.86±0.01
RBM2	0.8±0.01	0.8±0.01	0.8±0.01	0.82±0.02	0.8±0.06
RBM3	0.8±0.01	0.8±0.01	0.8±0.01	0.81±0.01	0.81±0.01
			GPCRProsite		
MLP1	0.75±0.01	0.75±0.01	0.75±0.01	0.75±0.01	0.62±0.04
MLP2	0.75±0.01	0.75±0.01	0.75±0.01	**0.76**±0.01	0.61±0.05
MLP3	0.75±0.01	0.75±0.01	0.75±0.01	**0.76**±0.01	0.63±0.08
SDA1	**0.76**±0.01	0.76±0.01	0.76±0.01	0.75±0.01	0.62±0.04
SDA2	0.75±0.01	0.75±0.01	0.75±0.01	**0.76**±0.01	0.59±0.05
SDA3	0.74±0.02	0.74±0.02	0.74±0.02	**0.76**±0.01	0.56±0.05
RBM1	0.66±0.01	0.66±0.01	0.66±0.01	0.75±0.01	0.62±0.04
RBM2	0.65±0.04	0.65±0.04	0.65±0.04	0.68±0.02	0.48±0.08
RBM3	0.39±0.22	0.39±0.22	0.39±0.22	0.39±0.25	0.39±0.25
			GPCRInterpro		
MLP1	0.88±0.0	0.88±0.0	0.88±0.0	0.89±0.01	0.87±0.02
MLP2	0.88±0.0	0.88±0.0	0.88±0.0	0.89±0.01	0.87±0.03
MLP3	0.88±0.0	0.88±0.0	0.88±0.0	0.89±0.01	0.84±0.06
SDA1	0.88±0.01	0.88±0.01	0.88±0.01	0.89±0.01	0.87±0.02
SDA2	0.88±0.01	0.88±0.01	0.88±0.01	**0.9**±0.01	0.87±0.02
SDA3	0.88±0.01	0.88±0.01	0.88±0.01	0.89±0.01	0.85±0.04
RBM1	0.83±0.03	0.83±0.03	0.83±0.03	0.89±0.01	0.87±0.02
RBM2	0.77±0.06	0.77±0.06	0.77±0.06	0.81±0.01	0.79±0.02
RBM3	0.52±0.03	0.52±0.03	0.52±0.03	0.71±0.09	0.7±0.09

Tabela A.6: Resultados dos conjuntos de dados de EC - Nível 2

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			ECPfam		
MLP1	0.97±0.0	0.97±0.0	0.97±0.0	0.98±0.0	**0.99**±0.01
MLP2	0.97±0.0	0.97±0.0	0.97±0.0	0.98±0.0	0.98±0.0
MLP3	0.97±0.0	0.97±0.0	0.97±0.0	0.98±0.0	0.98±0.0
SDA1	0.97±0.0	0.97±0.0	0.97±0.0	0.98±0.0	**0.99**±0.01
SDA2	0.97±0.0	0.97±0.0	0.97±0.0	0.98±0.0	0.98±0.0
SDA3	0.97±0.0	0.97±0.0	0.97±0.0	0.98±0.0	0.98±0.0
RBM1	0.95±0.0	0.95±0.0	0.95±0.0	0.98±0.0	**0.99**±0.01
RBM2	0.78±0.19	0.78±0.19	0.78±0.19	0.39±0.18	0.58±0.16
RBM3	0.77±0.19	0.77±0.19	0.77±0.19	0.27±0.14	0.26±0.15
			ECPrints		
MLP1	0.96±0.0	0.96±0.0	0.96±0.0	**0.97**±0.0	0.97±0.01
MLP2	0.97±0.01	0.97±0.01	0.97±0.01	**0.97**±0.0	0.97±0.0
MLP3	0.96±0.01	0.96±0.01	0.96±0.01	**0.97**±0.0	0.97±0.0
SDA1	0.97±0.0	0.97±0.0	**0.97**±0.0	**0.97**±0.0	0.97±0.01
SDA2	0.97±0.01	0.97±0.01	0.97±0.01	**0.97**±0.0	0.97±0.0
SDA3	0.96±0.0	0.96±0.0	0.96±0.0	**0.97**±0.0	**0.97**±0.0
RBM1	0.93±0.01	0.93±0.01	0.93±0.01	**0.97**±0.0	0.97±0.01
RBM2	0.19±0.03	0.19±0.03	0.19±0.03	0.3±0.0	0.66±0.04
RBM3	0.2±0.03	0.2±0.03	0.2±0.03	0.27±0.03	0.51±0.06
			ECProsite		
MLP1	0.97±0.01	0.97±0.01	0.97±0.01	**0.99**±0.0	0.98±0.01
MLP2	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.0
MLP3	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.0
SDA1	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.01
SDA2	0.97±0.01	0.97±0.01	0.97±0.01	**0.99**±0.0	0.98±0.0
SDA3	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.01
RBM1	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.01
RBM2	0.25±0.0	0.25±0.0	0.25±0.0	0.36±0.01	0.73±0.03
RBM3	0.25±0.0	0.25±0.0	0.25±0.0	0.24±0.01	0.24±0.01
			ECInterpro		
MLP1	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	**0.99**±0.0
MLP2	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.0
MLP3	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.0
SDA1	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	**0.99**±0.0
SDA2	0.96±0.0	0.96±0.0	0.96±0.0	**0.99**±0.0	0.98±0.01
SDA3	0.97±0.0	0.97±0.0	0.97±0.0	**0.99**±0.0	0.98±0.0
RBM1	0.96±0.0	0.96±0.0	0.96±0.0	**0.99**±0.0	**0.99**±0.0
RBM2	0.96±0.01	0.96±0.01	0.96±0.01	0.56±0.14	0.55±0.13
RBM3	0.71±0.25	0.71±0.25	0.71±0.25	0.33±0.15	0.33±0.15

Tabela A.7: Resultados nos conjuntos de dados de TEs - Nível 3

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
PGSB					
MLP1	0.8±0.02	0.8±0.02	0.8±0.02	**0.85**±0.03	**0.85**±0.03
MLP2	0.81±0.02	0.81±0.02	0.81±0.02	**0.85**±0.03	**0.85**±0.03
MLP3	0.8±0.02	0.8±0.02	0.8±0.02	0.84±0.04	0.84±0.04
SDA1	0.8±0.02	0.8±0.02	0.8±0.02	0.85±0.03	0.85±0.03
SDA2	0.81±0.02	0.81±0.02	0.81±0.02	0.84±0.02	0.84±0.02
SDA3	0.81±0.02	0.81±0.02	0.81±0.02	0.84±0.03	0.84±0.03
RBM1	0.75±0.04	0.75±0.04	0.75±0.04	**0.85**±0.03	**0.85**±0.03
RBM2	0.66±0.04	0.66±0.04	0.66±0.04	0.65±0.06	0.65±0.06
RBM3	0.47±0.26	0.47±0.26	0.47±0.26	0.51±0.22	0.51±0.22
REPBASE					
MLP1	0.84±0.01	0.84±0.01	0.84±0.01	**0.85**±0.01	**0.85**±0.01
MLP2	0.84±0.01	0.84±0.01	0.84±0.01	0.84±0.01	0.85±0.01
MLP3	0.84±0.01	0.84±0.01	0.84±0.01	0.84±0.02	0.84±0.02
SDA1	0.85±0.01	0.85±0.01	0.85±0.01	**0.85**±0.01	**0.85**±0.01
SDA2	0.85±0.01	0.85±0.01	0.85±0.01	0.85±0.02	0.85±0.02
SDA3	0.85±0.01	0.85±0.01	0.85±0.01	0.85±0.02	0.85±0.02
RBM1	0.74±0.03	0.74±0.03	0.74±0.03	**0.85**±0.01	**0.85**±0.01
RBM2	0.72±0.08	0.72±0.08	0.72±0.08	0.58±0.09	0.58±0.08
RBM3	0.4±0.13	0.4±0.13	0.4±0.13	0.4±0.11	0.4±0.11
PGSB + REPBASE					
MLP1	0.81±0.02	0.81±0.02	0.81±0.02	**0.83**±0.02	**0.83**±0.02
MLP2	0.82±0.02	0.82±0.02	0.82±0.02	**0.83**±0.02	**0.83**±0.02
MLP3	0.82±0.02	0.82±0.02	0.82±0.02	0.82±0.02	**0.83**±0.02
SDA1	0.82±0.01	0.82±0.01	0.82±0.01	**0.83**±0.02	**0.83**±0.02
SDA2	0.82±0.02	0.82±0.02	0.82±0.02	**0.83**±0.02	**0.83**±0.02
SDA3	0.82±0.02	0.82±0.02	0.82±0.02	**0.83**±0.02	**0.83**±0.02
RBM1	0.74±0.02	0.74±0.02	0.74±0.02	**0.83**±0.02	**0.83**±0.02
RBM2	0.62±0.07	0.62±0.07	0.62±0.07	0.59±0.03	0.59±0.03
RBM3	0.47±0.14	0.47±0.14	0.47±0.14	0.49±0.13	0.49±0.13

Tabela A.8: Resultados nos conjuntos de dados de GPCR - Nível 3

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			GPCRPfam		
MLP1	0.59±0.0	0.59±0.0	0.59±0.0	0.58±0.0	0.55±0.01
MLP2	**0.6**±0.01	**0.6**±0.01	**0.6**±0.01	0.59±0.01	0.43±0.04
MLP3	0.59±0.0	0.59±0.0	0.59±0.0	0.59±0.01	0.39±0.03
SDA1	0.59±0.01	0.59±0.01	0.59±0.01	0.58±0.0	0.55±0.01
SDA2	**0.6**±0.01	**0.6**±0.01	**0.6**±0.01	**0.6**±0.01	0.4±0.04
SDA3	0.59±0.01	0.59±0.01	0.59±0.01	0.6±0.01	0.4±0.05
RBM1	0.51±0.01	0.51±0.01	0.51±0.01	0.58±0.0	0.55±0.01
RBM2	0.49±0.04	0.49±0.04	0.49±0.04	0.51±0.01	0.2±0.1
RBM3	0.52±0.05	0.52±0.05	0.52±0.05	0.48±0.03	0.46±0.03
			GPCRPrints		
MLP1	0.78±0.01	0.78±0.01	0.78±0.01	0.79±0.01	**0.8**±0.01
MLP2	0.78±0.01	0.78±0.01	0.78±0.01	0.79±0.01	**0.8**±0.01
MLP3	0.77±0.01	0.77±0.01	0.77±0.01	0.79±0.01	0.77±0.06
SDA1	0.78±0.01	0.78±0.01	0.78±0.01	0.79±0.01	**0.8**±0.01
SDA2	0.78±0.01	0.78±0.01	0.78±0.01	**0.8**±0.01	**0.8**±0.01
SDA3	0.78±0.01	0.78±0.01	0.78±0.01	**0.8**±0.01	0.78±0.04
RBM1	0.74±0.01	0.74±0.01	0.74±0.01	0.79±0.01	**0.8**±0.01
RBM2	0.71±0.02	0.71±0.02	0.71±0.02	0.76±0.01	0.75±0.06
RBM3	0.72±0.01	0.72±0.01	0.72±0.01	0.7±0.03	0.7±0.03
			GPCRProsite		
MLP1	0.61±0.01	0.61±0.01	0.61±0.01	0.61±0.01	0.5±0.03
MLP2	0.61±0.01	0.61±0.01	0.61±0.01	**0.62**±0.01	0.49±0.05
MLP3	0.61±0.01	0.61±0.01	0.61±0.01	0.61±0.01	0.5±0.07
SDA1	**0.62**±0.01	**0.62**±0.01	**0.62**±0.01	0.61±0.01	0.5±0.03
SDA2	0.61±0.01	0.61±0.01	0.61±0.01	**0.62**±0.01	0.48±0.04
SDA3	0.61±0.01	0.61±0.01	0.61±0.01	**0.62**±0.01	0.45±0.04
RBM1	0.52±0.01	0.52±0.01	0.52±0.01	0.61±0.01	0.5±0.03
RBM2	0.52±0.04	0.52±0.04	0.52±0.04	0.54±0.02	0.34±0.07
RBM3	0.29±0.18	0.29±0.18	0.29±0.18	0.29±0.2	0.29±0.2
			GPCRInterpro		
MLP1	0.79±0.01	0.79±0.01	0.79±0.01	0.8±0.01	0.78±0.03
MLP2	0.79±0.01	0.79±0.01	0.79±0.01	0.8±0.01	0.78±0.03
MLP3	0.79±0.01	0.79±0.01	0.79±0.01	0.8±0.01	0.74±0.06
SDA1	0.79±0.01	0.79±0.01	0.79±0.01	0.8±0.01	0.78±0.03
SDA2	0.79±0.01	0.79±0.01	0.79±0.01	**0.81**±0.01	0.78±0.02
SDA3	0.78±0.0	0.78±0.0	0.78±0.0	**0.81**±0.01	0.77±0.04
RBM1	0.76±0.01	0.76±0.01	0.76±0.01	0.8±0.01	0.78±0.03
RBM2	0.69±0.05	0.69±0.05	0.69±0.05	0.7±0.02	0.69±0.03
RBM3	0.36±0.03	0.36±0.03	0.36±0.03	0.55±0.09	0.54±0.08

Tabela A.9: Resultados nos conjuntos de dados de EC - Nível 3

ECPfam					
MLP1	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	**0.97**±0.0
MLP2	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	**0.97**±0.0
MLP3	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	0.97±0.01
SDA1	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	**0.97**±0.0
SDA2	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	0.97±0.01
SDA3	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	0.97±0.01
RBM1	0.92±0.0	0.92±0.0	0.92±0.0	**0.97**±0.0	**0.97**±0.0
RBM2	0.74±0.19	0.74±0.19	0.74±0.19	0.33±0.15	0.5±0.14
RBM3	0.73±0.21	0.73±0.21	0.73±0.21	0.21±0.1	0.2±0.1
ECPrints					
MLP1	0.93±0.0	0.93±0.0	0.93±0.0	0.95±0.0	0.95±0.0
MLP2	0.93±0.0	0.93±0.0	0.93±0.0	0.96±0.01	0.95±0.0
MLP3	0.93±0.0	0.93±0.0	0.93±0.0	0.95±0.0	0.95±0.0
SDA1	0.93±0.0	0.93±0.0	0.93±0.0	0.95±0.0	0.95±0.0
SDA2	0.94±0.0	0.94±0.0	0.94±0.0	**0.96**±0.0	**0.96**±0.0
SDA3	0.94±0.0	0.94±0.0	0.94±0.0	**0.96**±0.0	**0.96**±0.0
RBM1	0.9±0.01	0.9±0.01	0.9±0.01	0.95±0.0	0.95±0.0
RBM2	0.15±0.03	0.15±0.03	0.15±0.03	0.29±0.01	0.66±0.04
RBM3	0.16±0.03	0.16±0.03	0.16±0.03	0.22±0.03	0.43±0.06
ECProsite					
MLP1	0.94±0.01	0.94±0.01	0.94±0.01	**0.97**±0.0	0.97±0.01
MLP2	0.94±0.01	0.94±0.01	0.94±0.01	**0.97**±0.0	0.96±0.01
MLP3	0.94±0.01	0.94±0.01	0.94±0.01	**0.97**±0.0	**0.97**±0.0
SDA1	0.95±0.01	0.95±0.01	0.95±0.01	**0.97**±0.0	0.97±0.01
SDA2	0.94±0.01	0.94±0.01	0.94±0.01	**0.97**±0.0	**0.97**±0.0
SDA3	0.94±0.0	0.94±0.0	0.94±0.0	**0.97**±0.0	**0.97**±0.0
RBM1	0.94±0.01	0.94±0.01	0.94±0.01	**0.97**±0.0	0.97±0.01
RBM2	0.21±0.0	0.21±0.0	0.21±0.0	0.33±0.01	0.69±0.03
RBM3	0.21±0.0	0.21±0.0	0.21±0.0	0.19±0.02	0.19±0.02
ECInterpro					
MLP1	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	**0.98**±0.0
MLP2	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	**0.98**±0.0
MLP3	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	**0.98**±0.0
SDA1	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	**0.98**±0.0
SDA2	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	0.97±0.01
SDA3	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	**0.98**±0.0
RBM1	0.94±0.0	0.94±0.0	0.94±0.0	**0.98**±0.0	**0.98**±0.0
RBM2	0.93±0.02	0.93±0.02	0.93±0.02	0.4±0.09	0.4±0.09
RBM3	0.68±0.26	0.68±0.26	0.68±0.26	0.24±0.11	0.24±0.11

Tabela A.10: Resultados nos conjuntos de dados de TEs - Nível 4

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
PGSB					
MLP1	0.59±0.07	0.59±0.07	0.59±0.07	0.63±0.08	0.64±0.08
MLP2	0.59±0.07	0.59±0.07	0.59±0.07	0.64±0.08	0.65±0.08
MLP3	0.58±0.08	0.58±0.08	0.58±0.08	0.63±0.09	0.64±0.09
SDA1	0.59±0.07	0.59±0.07	0.59±0.07	0.63±0.08	0.64±0.08
SDA2	0.6±0.07	0.6±0.07	0.6±0.07	0.65±0.06	0.65±0.07
SDA3	0.59±0.07	0.59±0.07	0.59±0.07	**0.66**±0.07	**0.66**±0.07
RBM1	0.44±0.09	0.44±0.09	0.44±0.09	0.63±0.08	0.64±0.08
RBM2	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0	0.0±0.0
RBM3	0.14±0.0	0.14±0.0	0.14±0.0	0.14±0.01	0.14±0.01
REPBASE					
MLP1	0.69±0.02	0.69±0.02	0.69±0.02	0.69±0.05	0.7±0.05
MLP2	0.69±0.01	0.69±0.01	0.69±0.01	0.68±0.05	0.69±0.05
MLP3	0.69±0.01	0.69±0.01	0.69±0.01	0.68±0.05	0.69±0.05
SDA1	**0.7**±0.02	**0.7**±0.02	**0.7**±0.02	0.69±0.05	0.7±0.05
SDA2	**0.7**±0.02	**0.7**±0.02	0.7±0.02	0.69±0.05	0.69±0.05
SDA3	0.69±0.02	0.69±0.02	0.69±0.02	0.68±0.05	0.68±0.05
RBM1	0.6±0.05	0.6±0.05	0.6±0.05	0.69±0.05	0.7±0.05
RBM2	0.59±0.05	0.59±0.05	0.59±0.05	0.6±0.09	0.59±0.09
RBM3	0.3±0.01	0.3±0.01	0.3±0.01	0.29±0.0	0.29±0.0
PGSB + REPBASE					
MLP1	0.66±0.03	0.66±0.03	0.66±0.03	0.65±0.07	0.66±0.07
MLP2	**0.67**±0.02	**0.67**±0.02	0.67±0.02	0.65±0.05	0.66±0.05
MLP3	0.66±0.02	0.66±0.02	0.66±0.02	0.64±0.06	0.64±0.06
SDA1	**0.67**±0.02	**0.67**±0.02	**0.67**±0.02	0.65±0.07	0.66±0.07
SDA2	**0.67**±0.02	**0.67**±0.02	**0.67**±0.02	0.66±0.05	0.67±0.05
SDA3	0.67±0.03	0.67±0.03	0.67±0.03	0.67±0.04	0.67±0.05
RBM1	0.55±0.04	0.55±0.04	0.55±0.04	0.65±0.07	0.66±0.07
RBM2	0.36±0.26	0.36±0.26	0.36±0.26	0.0±0.0	0.0±0.0
RBM3	0.24±0.01	0.24±0.01	0.24±0.01	0.0±0.0	0.0±0.0

Tabela A.11: Resultados nos conjuntos de dados de GPCR - Nível 4

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			GPCRPfam		
MLP1	0.3±0.01	0.3±0.01	0.3±0.01	0.28±0.01	0.24±0.01
MLP2	0.33±0.02	0.33±0.02	0.33±0.02	0.33±0.02	0.21±0.01
MLP3	0.33±0.01	0.33±0.01	0.33±0.01	0.32±0.02	0.18±0.04
SDA1	0.3±0.01	0.3±0.01	0.3±0.01	0.28±0.01	0.24±0.01
SDA2	**0.34**±0.02	**0.34**±0.02	**0.34**±0.02	**0.34**±0.02	0.21±0.04
SDA3	0.34±0.03	0.34±0.03	0.34±0.03	0.33±0.02	0.24±0.03
RBM1	0.01±0.0	0.01±0.0	0.01±0.0	0.28±0.01	0.24±0.01
RBM2	0.1±0.07	0.1±0.07	0.1±0.07	0.03±0.04	0.05±0.07
RBM3	0.2±0.04	0.2±0.04	0.2±0.04	0.2±0.01	0.18±0.0
			GPCRPrints		
MLP1	0.73±0.02	0.73±0.02	0.73±0.02	0.79±0.02	**0.87**±0.02
MLP2	0.73±0.02	0.73±0.02	0.73±0.02	0.8±0.02	**0.87**±0.02
MLP3	0.73±0.02	0.73±0.02	0.73±0.02	0.79±0.02	0.8±0.12
SDA1	0.73±0.02	0.73±0.02	0.73±0.02	0.79±0.02	**0.87**±0.02
SDA2	0.74±0.02	0.74±0.02	0.74±0.02	0.81±0.02	0.84±0.03
SDA3	0.73±0.02	0.73±0.02	0.73±0.02	0.82±0.02	0.8±0.08
RBM1	0.67±0.01	0.67±0.01	0.67±0.01	0.79±0.02	**0.87**±0.02
RBM2	0.66±0.03	0.66±0.03	0.66±0.03	0.73±0.05	0.76±0.13
RBM3	0.67±0.05	0.67±0.05	0.67±0.05	0.56±0.08	0.55±0.1
			GPCRProsite		
MLP1	0.41±0.02	0.41±0.02	0.41±0.02	0.41±0.02	0.38±0.02
MLP2	0.42±0.02	0.42±0.02	0.42±0.02	0.42±0.02	0.36±0.03
MLP3	0.42±0.02	0.42±0.02	0.42±0.02	0.42±0.01	0.35±0.04
SDA1	0.42±0.01	0.42±0.01	0.42±0.01	0.41±0.02	0.38±0.02
SDA2	0.42±0.02	0.42±0.02	0.42±0.02	**0.43**±0.02	0.36±0.04
SDA3	0.42±0.02	0.42±0.02	0.42±0.02	0.44±0.02	0.35±0.04
RBM1	0.31±0.03	0.31±0.03	0.31±0.03	0.41±0.02	0.38±0.02
RBM2	0.32±0.04	0.32±0.04	0.32±0.04	0.32±0.03	0.24±0.06
RBM3	0.21±0.03	0.21±0.03	0.21±0.03	0.0±0.0	0.0±0.0
			GPCRInterpro		
MLP1	0.77±0.02	0.77±0.02	0.77±0.02	**0.89**±0.01	0.82±0.05
MLP2	0.78±0.02	0.78±0.02	0.78±0.02	0.88±0.02	0.81±0.06
MLP3	0.79±0.02	0.79±0.02	0.79±0.02	**0.89**±0.01	0.76±0.13
SDA1	0.78±0.02	0.78±0.02	0.78±0.02	**0.89**±0.01	0.82±0.05
SDA2	0.77±0.01	0.77±0.01	0.77±0.01	**0.89**±0.01	0.83±0.05
SDA3	0.77±0.02	0.77±0.02	0.77±0.02	**0.89**±0.01	0.79±0.1
RBM1	0.79±0.02	0.79±0.02	0.79±0.02	**0.89**±0.01	0.82±0.05
RBM2	0.67±0.1	0.67±0.1	0.67±0.1	0.65±0.07	0.65±0.07
RBM3	0.16±0.02	0.16±0.02	0.16±0.02	0.47±0.02	0.45±0.03

Tabela A.12: Resultados nos conjuntos de dados de EC - Nível 4

	SWV	SimplePrune	LCPN	nlLCPN	LCPNB
			ECPfam		
MLP1	0.87±0.0	0.87±0.0	0.87±0.0	0.94±0.01	0.94±0.01
MLP2	0.87±0.0	0.87±0.0	0.87±0.0	0.95±0.01	0.94±0.01
MLP3	0.87±0.0	0.87±0.0	0.87±0.0	**0.95**±0.0	0.94±0.0
SDA1	0.87±0.0	0.87±0.0	0.87±0.0	0.94±0.01	0.94±0.01
SDA2	0.87±0.0	0.87±0.0	0.87±0.0	**0.95**±0.0	**0.95**±0.0
SDA3	0.87±0.0	0.87±0.0	0.87±0.0	**0.95**±0.0	**0.95**±0.0
RBM1	0.86±0.0	0.86±0.0	0.86±0.0	0.94±0.01	0.94±0.01
RBM2	0.7±0.2	0.7±0.2	0.7±0.2	0.26±0.11	0.38±0.12
RBM3	0.69±0.22	0.69±0.22	0.69±0.22	0.12±0.06	0.13±0.05
			ECPrints		
MLP1	0.86±0.0	0.86±0.0	0.86±0.0	0.92±0.01	**0.93**±0.0
MLP2	0.86±0.0	0.86±0.0	0.86±0.0	0.93±0.01	0.93±0.01
MLP3	0.85±0.0	0.85±0.0	0.85±0.0	0.93±0.01	0.93±0.01
SDA1	0.86±0.0	0.86±0.0	0.86±0.0	0.92±0.01	**0.93**±0.0
SDA2	0.86±0.0	0.86±0.0	0.86±0.0	**0.93**±0.0	0.93±0.01
SDA3	0.86±0.0	0.86±0.0	0.86±0.0	**0.93**±0.0	0.93±0.01
RBM1	0.82±0.0	0.82±0.0	0.82±0.0	0.92±0.01	**0.93**±0.0
RBM2	0.12±0.03	0.12±0.03	0.12±0.03	0.27±0.02	0.62±0.05
RBM3	0.12±0.03	0.12±0.03	0.12±0.03	0.17±0.02	0.3±0.06
			ECProsite		
MLP1	0.88±0.0	0.88±0.0	0.88±0.0	0.93±0.01	0.93±0.01
MLP2	0.88±0.0	0.88±0.0	0.88±0.0	0.93±0.01	0.93±0.01
MLP3	0.87±0.0	0.87±0.0	0.87±0.0	0.93±0.01	0.93±0.01
SDA1	0.88±0.0	0.88±0.0	0.88±0.0	0.93±0.01	0.93±0.01
SDA2	0.88±0.01	0.88±0.01	0.88±0.01	**0.94**±0.0	0.93±0.0
SDA3	0.88±0.0	0.88±0.0	0.88±0.0	**0.94**±0.0	**0.94**±0.0
RBM1	0.86±0.0	0.86±0.0	0.86±0.0	0.93±0.01	0.93±0.01
RBM2	0.16±0.0	0.16±0.0	0.16±0.0	0.25±0.01	0.57±0.02
RBM3	0.16±0.0	0.16±0.0	0.16±0.0	0.11±0.03	0.11±0.03
			ECInterpro		
MLP1	0.88±0.0	0.88±0.0	0.88±0.0	0.96±0.01	0.97±0.01
MLP2	0.88±0.0	0.88±0.0	0.88±0.0	0.96±0.01	0.97±0.01
MLP3	0.88±0.0	0.88±0.0	0.88±0.0	0.96±0.01	0.96±0.01
SDA1	0.88±0.0	0.88±0.0	0.88±0.0	0.96±0.01	0.97±0.01
SDA2	0.87±0.0	0.87±0.0	0.87±0.0	0.97±0.0	**0.97**±0.0
SDA3	0.88±0.0	0.88±0.0	0.88±0.0	0.97±0.0	**0.97**±0.0
RBM1	0.87±0.0	0.87±0.0	0.87±0.0	0.96±0.01	0.97±0.01
RBM2	0.86±0.01	0.86±0.01	0.86±0.01	0.27±0.08	0.27±0.08
RBM3	0.61±0.25	0.61±0.25	0.61±0.25	0.16±0.05	0.16±0.05

REFERÊNCIAS

ALTSCHUL, S. F. et al. Basic local alignment search tool. *Journal of molecular biology*, Elsevier, v. 215, n. 3, p. 403–410, 1990.

ALVES, R. T.; DELGADO, M. R.; FREITAS, A. A. Knowledge discovery with artificial immune systems for hierarchical multi-label classification of protein functions. In: *International Conference on Fuzzy Systems*. [S.l.: s.n.], 2010. p. 1–8. ISSN 1098-7584.

BABBAR, R. et al. On flat versus hierarchical classification in large-scale taxonomies. In: BURGES, C. et al. (Ed.). *Advances in Neural Information Processing Systems 26*. [S.l.: s.n.], 2013. p. 1824–1832.

BARROS, R. C. et al. Probabilistic clustering for hierarchical multi-label classification of protein functions. In: SPRINGER. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. [S.l.], 2013. p. 385–400.

BENGIO, Y. et al. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, MIT; 1998, v. 19, p. 153, 2007.

BENGIO, Y. et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, Now Publishers, Inc., v. 2, n. 1, p. 1–127, 2009.

BERGMAN, C. M.; QUESNEVILLE, H. Discovering and detecting transposable elements in genome sequences. *Briefings in Bioinformatics*, v. 8, n. 6, p. 382–392, 2007. Disponível em: <http://bib.oxfordjournals.org/content/8/6/382.abstract>.

BIÉMONT, C.; VIEIRA, C. Genetics: junk dna as an evolutionary force. *Nature*, Nature Publishing Group, v. 443, n. 7111, p. 521–524, 2006.

BORGES, H. B.; NIEVOLA, J. C. Multi-label hierarchical classification using a competitive neural network for protein function prediction. In: *The 2012 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2012. p. 1–8. ISSN 2161-4393.

CASEY, R. Blast sequences aid in genomics and proteomics. *Business Intelligence Network*, 2005.

CECI, M.; MALERBA, D. Classifying web documents in a hierarchy of categories: A comprehensive study. *J. Intell. Inf. Syst.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 28, n. 1, p. 37–78, fev. 2007. ISSN 0925-9902. Disponível em: <http://dx.doi.org/10.1007/s10844-006-0003-2>.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. de. Hierarchical multi-label classification using local neural networks. *Journal of Computer and System Sciences*, v. 80, n. 1, p. 39 – 56, 2014. ISSN 0022-0000. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0022000013000718>.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. P. L. F. d. Neural networks for hierarchical classification of g-protein coupled receptors. In: *2013 Brazilian Conference on Intelligent Systems*. [S.l.: s.n.], 2013. p. 125–130.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. P. L. F. de. A genetic algorithm for hierarchical multi-label classification. In: *Proceedings of the 27th Annual ACM Symposium on Applied Computing*. New York, NY, USA: ACM, 2012. (SAC '12), p. 250–255. ISBN 978-1-4503-0857-1. Disponível em: <http://doi.acm.org/10.1145/2245276.2245325>.

CERRI, R.; BARROS, R. C.; CARVALHO, A. C. P. L. F. de. Hierarchical classification of gene ontology-based protein functions with neural networks. In: *2015 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2015. p. 1–8. ISSN 2161-4393.

CERRI, R. et al. Reduction strategies for hierarchical multi-label classification in protein function prediction. *BMC Bioinformatics*, v. 17, n. 1, p. 373, 2016.

CESA-BIANCHI, N.; RE, M.; VALENTINI, G. Synergy of multi-label hierarchical ensembles, data fusion, and cost-sensitive methods for gene functional inference. *Machine Learning*, Springer Netherlands, p. 1–33, 2011. ISSN 0885-6125.

COSTA, E. D. P. et al. Annotating transposable elements in the genome using relational decision tree ensembles. *Online preprints 23th Conference on Inductive Logic Programming*, p. 1–6, 2013.

COSTA, E. P. et al. Comparing several approaches for hierarchical classification of proteins with decision trees. In: SPRINGER. *Brazilian Symposium on Bioinformatics*. [S.l.], 2007. p. 126–137.

COSTA, E. P. et al. Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In: ____. *Advances in Bioinformatics and Computational Biology: Third Brazilian Symposium on Bioinformatics, BSB 2008, Santo André, Brazil, August 28-30, 2008. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. p. 35–46. ISBN 978-3-540-85557-6.

DEMSAR, 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. Disponível em: <http://dl.acm.org/citation.cfm?id=1248547.1248548>.

DEMUTH, H. B. et al. *Neural network design*. [S.l.]: Martin Hagan, 2014.

DENG, L.; YU, D. et al. Deep learning: methods and applications. *Foundations and Trends® in Signal Processing*, Now Publishers, Inc., v. 7, n. 3–4, p. 197–387, 2014.

DUMAIS, S.; CHEN, H. Hierarchical classification of web content. In: *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. New York, NY, USA: ACM, 2000. (SIGIR '00), p. 256–263. ISBN 1-58113-226-3. Disponível em: <http://doi.acm.org/10.1145/345508.345593>.

FABRIS, F.; FREITAS, A. A. An efficient algorithm for hierarchical classification of protein and gene functions. In: *DEXA Workshops*. [S.l.: s.n.], 2014.

FERRANDIN, M. et al. A centroid-based approach for hierarchical classification. In: *Proceedings of the 17th International Conference on Enterprise Information Systems - Volume 1: ICEIS,*. [S.l.: s.n.], 2015. p. 25–33. ISBN 978-989-758-096-3.

FERRANDIN, M.; ROMAO, L. M. Hierarchical classification with jumping emerging patterns. *IEEE Latin America Transactions*, IEEE, v. 14, n. 9, p. 4143–4149, 2016.

FESCHOTTE, C. et al. Exploring Repetitive DNA Landscapes Using REPCLASS, a Tool That Automates the Classification of Transposable Elements in Eukaryotic Genomes. *Genome Biology and Evolution*, v. 1, p. 205–220, 2010.

FINNEGAN, D. J. Eukaryotic transposable elements and genome evolution. *Trends in Genetics*, v. 5, p. 103 – 107, 1989. ISSN 0168-9525. Disponível em: <http://www.sciencedirect.com/science/article/pii/0168952589900395>.

FINNEGAN, D. J. Transposable elements. *Current Opinion in Genetics & Development*, v. 2, n. 6, p. 861 – 867, 1992. ISSN 0959-437X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0959437X0580108X>.

FREUND, Y.; HAUSSLER, D. Unsupervised learning of distributions of binary vectors using two layer networks. Computer Research Laboratory [University of California, Santa Cruz], 1994.

GHAZI, D.; INKPEN, D.; SZPAKOWICZ, S. Hierarchical versus flat classification of emotions in text. In: *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text*. Stroudsburg, PA, USA: Association for Computational Linguistics, 2010. (CAAGET '10), p. 140–146. Disponível em: <http://dl.acm.org/citation.cfm?id=1860631.1860648>.

GLOROT, X.; BENGIO, Y. Understanding the difficulty of training deep feedforward neural networks. In: *Aistats*. [S.l.: s.n.], 2010. v. 9, p. 249–256.

GROSSBERG, S. Nonlinear neural networks: Principles, mechanisms, and architectures. *Neural networks*, Elsevier, v. 1, n. 1, p. 17–61, 1988.

HAYASHI, K.; YOSHIDA, H. Refunctionalization of the ancient rice blast disease resistance gene pit by the recruitment of a retrotransposon as a promoter. *The Plant Journal*, Wiley Online Library, v. 57, n. 3, p. 413–425, 2009.

HAYKIN, S. *Neural Networks: A Comprehensive Foundation*. 2nd. ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998. ISBN 0132733501.

HERNANDEZ, J.; SUCAR, L.; MORALES, E. A hybrid global-local approach for hierarchical classification. 2013. Disponível em: <http://www.aaai.org/ocs/index.php/FLAIRS/FLAIRS13/paper/view/5878>.

HERNáNDEZ, J.; SUCAR, L. E.; MORALES, E. F. Multidimensional hierarchical classification. *Expert Systems with Applications*, v. 41, n. 17, p. 7671 – 7677, 2014. ISSN 0957-4174. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0957417414003492>.

HINTON, G. E.; OSINDERO, S.; TEH, Y.-W. A fast learning algorithm for deep belief nets. *Neural computation*, MIT Press, v. 18, n. 7, p. 1527–1554, 2006.

HINTON, G. E.; SALAKHUTDINOV, R. R. Reducing the dimensionality of data with neural networks. *science*, American Association for the Advancement of Science, v. 313, n. 5786, p. 504–507, 2006.

HINTON, G. E.; SEJNOWSKI, T. J. Learning and releaming in boltzmann machines. *Parallel Distrilmted Processing*, v. 1, 1986.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural computation*, MIT Press, v. 9, n. 8, p. 1735–1780, 1997.

HOEDE, C. et al. Pastec: An automatic transposable element classification tool. *PLoS ONE*, 2014.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, National Acad Sciences, v. 79, n. 8, p. 2554–2558, 1982.

JR, C. N. S.; FREITAS, A. A. A global-model naive bayes approach to the hierarchical prediction of protein functions. In: IEEE. *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. [S.l.], 2009. p. 992–997.

JR, C. N. S.; FREITAS, A. A. Selecting different protein representations and classification algorithms in hierarchical protein function prediction. *Intelligent Data Analysis*, IOS Press, v. 15, n. 6, p. 979–999, 2011.

JURKA, J. et al. Repbase update, a database of eukaryotic repetitive elements. *Cytogenetic and genome research*, v. 110, p. 462–467, 2005.

JURKA, J. et al. Censor—a program for identification and elimination of repetitive elements from dna sequences. *Computers & Chemistry*, v. 20, n. 1, p. 119 – 121, 1996. ISSN 0097-8485. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0097848596800131>.

JúNIOR, T. C. et al. Structural features and transcript-editing analysis of sugarcane (saccharum officinarum l.) chloroplast genome. *Current Genetics*, Springer-Verlag, v. 46, n. 6, p. 366–373, 2004. ISSN 0172-8083. Disponível em: <http://dx.doi.org/10.1007/s00294-004-0542-4>.

KIM, Y.-J.; LEE, J.; HAN, K. Transposable elements: no more'junk dna'. *Genomics & informatics*, v. 10, n. 4, p. 226–233, 2012.

KINGMA, D.; BA, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

KINGMA, D. P.; BA, J. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014. Disponível em: <http://arxiv.org/abs/1412.6980>.

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

KOHANY, O. et al. Annotation, submission and screening of repetitive elements in repbase: Repbasesubmitter and censor. *BMC bioinformatics*, BioMed Central, v. 7, n. 1, p. 474, 2006.

KRUGER, N. et al. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE transactions on pattern analysis and machine intelligence*, IEEE, v. 35, n. 8, p. 1847–1871, 2013.

LANDER, E. S. et al. Initial sequencing and analysis of the human genome. *Nature*, Nature Publishing Group, v. 409, n. 6822, p. 860–921, 2001.

LAROCHELLE, H. et al. Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, v. 10, n. Jan, p. 1–40, 2009.

LAROCHELLE, H. et al. An empirical evaluation of deep architectures on problems with many factors of variation. In: *Proceedings of the 24th International Conference on Machine Learning*. New York, NY, USA: ACM, 2007. (ICML '07), p. 473–480. ISBN 978-1-59593-793-3. Disponível em: <http://doi.acm.org/10.1145/1273496.1273556>.

LECUN, Y. et al. A tutorial on energy-based learning. *Predicting structured data*, v. 1, p. 0, 2006.

LIU, W. et al. A survey of deep neural network architectures and their applications. *Neurocomputing*, Elsevier, 2016.

MASTELINI, S. M. et al. Multi-output tree chaining: An interpretative modelling and lightweight multi-target approach. *Journal of Signal Processing Systems*, Springer, p. 1–25, 2018.

MCCLINTOCK, B. The origin and behavior of mutable loci in maize. *Proceedings of the National Academy of Sciences*, National Acad Sciences, v. 36, n. 6, p. 344–355, 1950.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943.

MERSCHMANN, L. H. de C.; FREITAS, A. A. An extended local hierarchical classifier for prediction of protein and gene functions. In: SPRINGER. *International Conference on Data Warehousing and Knowledge Discovery*. [S.l.], 2013. p. 159–171.

MIN, S.; LEE, B.; YOON, S. Deep learning in bioinformatics. *CoRR*, abs/1603.06430, 2016. Http://arxiv.org/abs/1603.06430.

MOUNT, D. W. *Bioinformatics: sequence and genome analysis*. [S.l.]: Cold Spring Harbor Laboratory Press, 2004.

NAIK, A.; RANGWALA, H. Inconsistent node flattening for improving top-down hierarchical classification. In: *2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. [S.l.: s.n.], 2016. p. 379–388.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. [S.l.: s.n.], 2010. p. 807–814.

NAKANO, F. K.; CERRI, R. Denoising auto-encoders as feature extractors in hierarchical classification problems. In: SBC ENIAC. *National meeting on artificial and computational intelligence*. [S.l.], 2017. p. 343–354.

NAKANO, F. K. et al. Top-down strategies for hierarchical classification of transposable elements with neural networks. In: *2017 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2017. p. 2539–2546.

NAKANO, F. kenji et al. Stacking methods for hierarchical classification. In: IEEE. *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*. [S.l.], 2017. p. 289–296.

NAKAYASHIKI, H. The trickster in the genome: contribution and control of transposable elements. *Genes to Cells*, Wiley Online Library, v. 16, n. 8, p. 827–841, 2011.

NG, P. C.; HENIKOFF, S. Predicting deleterious amino acid substitutions. *Genome research*, Cold Spring Harbor Lab, v. 11, n. 5, p. 863–874, 2001.

NUSSBAUMER, T. et al. Mips plantsdb: a database framework for comparative plant genome research. *Nucleic Acids Research*, v. 41, n. D1, p. D1144–D1151, 2013.

OHNO, S. So much"junk"dna in our genome. In: *Brookhaven symposia in biology*. [S.l.: s.n.], 1972. v. 23, p. 366–370.

ORGEL, L. E.; CRICK, F. Selfish dna: the ultimate parasite. *Nature*, v. 284, p. 604–607, 1980.

OTERO, F. E. B.; 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. ISSN 1865-9292. Disponível em: <http://dx.doi.org/10.1007/s12293-010-0045-4>.

PAES, B. C.; PLASTINO, R.; FREITAS, A. A. *Improving Local Per Level Hierarchical Classification*. 2012.

PEDREGOSA, F. et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, v. 12, p. 2825–2830, 2011.

PHACHONGKITPHIPHAT, N.; VATEEKUL, P. An improvement of flat approach on hierarchical text classification using top-level pruning classifiers. In: *2014 11th International Joint Conference on Computer Science and Software Engineering (JCSSE)*. [S.l.: s.n.], 2014. p. 86–90.

PIéGU, B. et al. A survey of transposable element classification systems – a call for a fundamental update to meet the challenge of their diversity and complexity. *Molecular Phylogenetics and Evolution*, v. 86, p. 90 – 109, 2015. ISSN 1055-7903. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1055790315000664>.

PK MITTAL VK, G. S. N. Retropred: A tool for prediction, classification and extraction of non-ltr retrotransposons (lines & sines) from the genome by integrating pals, piler, meme and ann. *BioInformation*, p. 263–270, jan 2008.

POLYANOVSKY, V. O.; ROYTBERG, M. A.; TUMANYAN, V. G. Comparative analysis of the quality of a global algorithm and a local algorithm for alignment of two sequences. *Algorithms for molecular biology*, BioMed Central, v. 6, n. 1, p. 25, 2011.

POSADA, D. *Bioinformatics for DNA sequence analysis*. [S.l.]: Springer, 2009.

RAMíREZ-CORONA, M.; SUCAR, L. E.; MORALES, E. F. Hierarchical multilabel classification based on path evaluation. *International Journal of Approximate Reasoning*, v. 68, p. 179 – 193, 2016. ISSN 0888-613X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0888613X15001073>.

ROUAULT, J.-D. et al. Automatic classification within families of transposable elements: Application to the mariner family. *Gene*, v. 448, n. 2, p. 227 – 232, 2009. ISSN 0378-1119. Genomic Impact of Eukaryotic Transposable Elements. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0378111909004570>.

ROUX, N. L.; BENGIO, Y. Representational power of restricted boltzmann machines and deep belief networks. *Neural computation*, MIT Press, v. 20, n. 6, p. 1631–1649, 2008.

RUDER, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. In: RUMELHART, D. E.; MCCLELLAND, J. L.; GROUP, C. P. R. (Ed.). Cambridge, MA, USA: MIT Press, 1986. cap. Learning Internal Representations by Error Propagation, p. 318–362. ISBN 0-262-68053-X. Disponível em: <http://dl.acm.org/citation.cfm?id=104279.104293>.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2. ed. [S.l.]: Pearson Education, 2003. ISBN 0137903952.

SCHMIDHUBER, J. Deep learning in neural networks: An overview. *Neural networks*, Elsevier, v. 61, p. 85–117, 2015.

SEBASTIANI, F. Machine learning in automated text categorization. *ACM COMPUTING SURVEYS*, v. 34, p. 1–47, 2002.

SEBERG, O.; PETERSEN, G. A unified classification system for eukaryotic transposable elements should reflect their phylogeny. *Nat Rev Genet*, Nature Publishing Group, v. 10, n. 4, p. 276–276, Apr 2009. ISSN 1471-0056. Disponível em: <http://dx.doi.org/10.1038/nrg2165-c3>.

SECKER, A. et al. *An Experimental Comparison of Classification Algorithms for the Hierarchical Prediction of Protein Function*. 2007.

SECKER, A. et al. Hierarchical classification of g-protein-coupled receptors with data-driven selection of attributes and classifiers. *International journal of data mining and bioinformatics*, Inderscience Publishers, v. 4, n. 2, p. 191–210, 2010.

SILLA, C. N.; FREITAS, A. A. A global-model naive bayes approach to the hierarchical prediction of protein functions. In: IEEE. *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*. [S.l.], 2009. p. 992–997.

SILLA, C. N.; FREITAS, A. A. Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In: *2009 IEEE International Conference on Systems, Man and Cybernetics*. [S.l.: s.n.], 2009. p. 3499–3504. ISSN 1062-922X.

SILLA, C. N. J.; FREITAS, A. A. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, Springer US, v. 22, n. 1-2, p. 31–72, 2010. ISSN 1384-5810.

SMIT, A.; HUBLEY, R.; GREEN, P. Repeatmasker open-4.0. 2013–2015. *Institute for Systems Biology. http://repeatmasker. org*, 2015.

SMITH, T. F.; WATERMAN, M. S. Identification of common molecular subsequences. *Journal of molecular biology*, Elsevier, v. 147, n. 1, p. 195–197, 1981.

SMOLENSKY, P. *Information processing in dynamical systems: Foundations of harmony theory*. [S.l.], 1986.

SOTERO-CAIO, C. G. et al. Evolution and diversity of transposable elements in vertebrate genomes. *Genome Biology and Evolution*, v. 9, n. 1, p. 161, 2017. Disponível em: <+ http://dx.doi.org/10.1093/gbe/evw264>.

STEINBISS, S. et al. Fine-grained annotation and classification of de novo predicted ltr retrotransposons. *Nucleic Acids Research*, 2009. Disponível em: <http://nar.oxfordjournals.org/content/early/2009/09/28/nar.gkp759.abstract>.

SUN, A. et al. Blocking reduction strategies in hierarchical text classification. *IEEE Transactions on Knowledge and Data Engineering*, v. 16, n. 10, p. 1305–1308, Oct 2004. ISSN 1041-4347.

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

TEMPEL, S. Using and understanding repeatmasker. *Mobile Genetic Elements: Protocols and Genomic Applications*, Springer, p. 29–51, 2012.

TEMPEL, S.; POLLET, N.; TAHI, F. ncrnaclassifier: a tool for detection and classification of transposable element sequences in rna hairpins. *BMC Bioinformatics*, v. 13, n. 1, p. 1–13, 2012. ISSN 1471-2105. Disponível em: <http://dx.doi.org/10.1186/1471-2105-13-246>.

Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, maio 2016. Disponível em: <http://arxiv.org/abs/1605.02688>.

VENS, C. et al. Decision trees for hierarchical multi-label classification. *Machine Learning*, v. 73, n. 2, p. 185, 2008. ISSN 1573-0565. Disponível em: <http://dx.doi.org/10.1007/s10994-008-5077-3>.

VETTORE, A. L. et al. Analysis and functional annotation of an expressed sequence tag collection for tropical crop sugarcane. *Genome Research*, v. 13, n. 12, p. 2725–2735, 2003. Disponível em: <http://genome.cshlp.org/content/13/12/2725.abstract>.

VINCENT, P. et al. Extracting and composing robust features with denoising autoencoders. In: ACM. *Proceedings of the 25th international conference on Machine learning*. [S.l.], 2008. p. 1096–1103.

VINCENT, P. et al. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, v. 11, n. Dec, p. 3371–3408, 2010.

WANG, X. L.; ZHAO, H.; LU, B. L. A meta-top-down method for large-scale hierarchical classification. *IEEE Transactions on Knowledge and Data Engineering*, v. 26, n. 3, p. 500–513, March 2014. ISSN 1041-4347.

WICKER, T. et al. A unified classification system for eukaryotic transposable elements. *Nat Rev Genet*, 2007.

WICKER, T. et al. Reply: A unified classification system for eukaryotic transposable elements should reflect their phylogeny. *Nat Rev Genet*, Nature Publishing Group, v. 10, n. 4, p. 276–276, Apr 2009. ISSN 1471-0056. Disponível em: <http://dx.doi.org/10.1038/nrg2165-c4>.

XU, Z.; WANG, H. Ltr_finder: an efficient tool for the prediction of full-length ltr retrotransposons. *Nucleic acids research*, Oxford Univ Press, v. 35, n. suppl 2, p. W265–W268, 2007.

YING, C.; YING, D. run. Novel top-down methods for hierarchical text classification. *Procedia Engineering*, v. 24, p. 329 – 334, 2011. ISSN 1877-7058. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1877705811055032>.

ZHANG, F. et al. Grasp the implicit features: Hierarchical emotion classification based on topic model and svm. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2016. p. 3592–3599.

ZHU, S.; WEI, X.-Y.; NGO, C.-W. Collaborative error reduction for hierarchical classification. *Computer Vision and Image Understanding*, v. 124, p. 79 – 90, 2014. ISSN 1077-3142. Large Scale Multimedia Semantic Indexing. Disponível em: <http://www.sciencedirect.com/science/article/pii/S1077314214000769>.

ZIMEK, A. et al. A study of hierarchical and flat classification of proteins. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 7, n. 3, p. 563–571, jul. 2010. ISSN 1545-5963. Disponível em: <http://dx.doi.org/10.1109/TCBB.2008.104>.