Técnicas para o problema de dados desbalanceados em classificação hierárquica

Victor Hugo Barella

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Victor Hugo Barella

Técnicas para o problema de dados desbalanceados em classificação hierárquica

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional. *VERSÃO REVISADA*

Área de Concentração: Ciências de Computação e Matemática Computacional

Orientador: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

USP – São Carlos
Setembro de 2015

Ficha catalográfica elaborada pela Biblioteca Prof. Achille Bassi
e Seção Técnica de Informática, ICMC/USP,
com os dados fornecidos pelo(a) autor(a)

B248t Barella, Victor Hugo
 Técnicas para o problema de dados desbalanceados
em classificação hierárquica / Victor Hugo Barella;
orientador André Carlos Ponce de Leon Ferreira de
Carvalho. -- São Carlos, 2015.
 85 p.

 Dissertação (Mestrado - Programa de Pós-Graduação
em Ciências de Computação e Matemática
Computacional) -- Instituto de Ciências Matemáticas
e de Computação, Universidade de São Paulo, 2015.

 1. Classificação Hierárquica. 2. Aprendizado
Supervisionado. 3. Desbalanceamento de Dados. 4.
Dados Desbalanceados. I. Ponce de Leon Ferreira de
Carvalho, André Carlos, orient. II. Título.

Victor Hugo Barella

**Techniques for the problem of imbalanced data in
hierarchical classification**

Master dissertation submitted to the Instituto de Ciências Matemáticas e de Computação - ICMC-USP, in partial fulfillment of the requirements for the degree of the Master Program in Computer Science and Computational Mathematics. *FINAL VERSION*

Concentration Area: Computer Science and Computational Mathematics

Advisor: Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho

**USP – São Carlos
September 2015**

Agradecimentos

Agradeço a Deus.

Aos meus pais por todo o amor colocado em mim.

Ao prof. Dr. André Carvalho por me incentivar e orientar pacientemente durante o período do mestrado e por ser um exemplo a quem sempre seguirei em toda minha jornada profissional.

Aos professores do ICMC prof. Dr. Solange Rezende, prof. Dr. Rodrigo Mello, e todos os outros pelas ótimas discussões em aula e em corredores.

A todos os funcionários do ICMC pela atenção e cuidado em suas funções.

Aos meus amigos Glauco, Eduardo e Luís Paulo por me ajudarem com a escrita da monografia, artigo e discussões importantes sobre o trabalho.

Aos amigos que fiz em São Carlos, Dario, Lucas, Ever, Julio, Anderson, Vini e todos os outros por recarregarem minhas baterias. Esse texto tem um pouco da energia de cada um de vocês.

A todos que moraram comigo durante esse período, Amanda, Mari, Marcelo, Rafael e Lara por compreenderem momentos difíceis e promoverem um ambiente de convívio harmonioso.

Aos colegas de ICMC Rafael, Lucas, Valéria, Ricardo, Alinne, Carlos, Kemilly, Adriano, Giovana e todos os outros por tornarem o ambiente de trabalho o mais leve possível.

A todos os meus amigos de Piracicaba e Rio Claro pelo apoio e torcida.

À FAPESP, à CAPES e ao CNPq pelo apoio financeiro para a realização do projeto.

Os recentes avanços da ciência e tecnologia viabilizaram o crescimento de dados em quantidade e disponibilidade. Junto com essa explosão de informações geradas, surge a necessidade de analisar dados para descobrir conhecimento novo e útil. Desse modo, áreas que visam extrair conhecimento e informações úteis de grandes conjuntos de dados se tornaram grandes oportunidades para o avanço de pesquisas, tal como o Aprendizado de Máquina (AM) e a Mineração de Dados (MD). Porém, existem algumas limitações que podem prejudicar a acurácia de alguns algoritmos tradicionais dessas áreas, por exemplo o desbalanceamento das amostras das classes de um conjunto de dados. Para mitigar tal problema, algumas alternativas têm sido alvos de pesquisas nos últimos anos, tal como o desenvolvimento de técnicas para o balanceamento artificial de dados, a modificação dos algoritmos e propostas de abordagens para dados desbalanceados. Uma área pouco explorada sob a visão do desbalanceamento de dados são os problemas de classificação hierárquica, em que as classes são organizadas em hierarquias, normalmente na forma de árvore ou DAG (*Direct Acyclic Graph*). O objetivo deste trabalho foi investigar as limitações e maneiras de minimizar os efeitos de dados desbalanceados em problemas de classificação hierárquica. Os experimentos realizados mostram que é necessário levar em consideração as características das classes hierárquicas para a aplicação (ou não) de técnicas para tratar problemas dados desbalanceados em classificação hierárquica.

Palavras-chave: Classificação Hierárquica; Aprendizado Supervisionado; Desbalanceamento de Dados; Dados Desbalanceados

Abstract

Recent advances in science and technology have made possible the data growth in quantity and availability. Along with this explosion of generated information, there is a need to analyze data to discover new and useful knowledge. Thus, areas for extracting knowledge and useful information in large datasets have become great opportunities for the advancement of research, such as Machine Learning (ML) and Data Mining (DM). However, there are some limitations that may reduce the accuracy of some traditional algorithms of these areas, for example the imbalance of classes samples in a dataset. To mitigate this drawback, some solutions have been the target of research in recent years, such as the development of techniques for artificial balancing data, algorithm modification and new approaches for imbalanced data. An area little explored in the data imbalance vision are the problems of hierarchical classification, in which the classes are organized into hierarchies, commonly in the form of tree or DAG (*Direct Acyclic Graph*). The goal of this work aims at investigating the limitations and approaches to minimize the effects of imbalanced data with hierarchical classification problems. The experimental results show the need to take into account the features of hierarchical classes when deciding the application of techniques for imbalanced data in hierarchical classification.

Keywords: Hierarchical Classification; Supervised Learning; Imbalanced Data; Data Imbalance

Sumário

1	Introdução	1
1.1	Contextualização	1
1.2	Desafios e Definição do Problema	2
1.2.1	Distribuição Desbalanceada de Classes	2
1.2.2	Classificação Hierárquica	5
1.3	Objetivos	6
1.4	Contribuições	7
1.5	Organização do Trabalho	7
2	Fundamentação Teórica	9
2.1	Dados Desbalanceados	9
2.1.1	Pré-processamento de Dados	10
2.1.2	Adaptação de Algoritmos	15
2.2	Classificação Hierárquica	18
2.2.1	Classificadores Planos	18
2.2.2	Classificadores Locais	19
2.2.2.1	Classificadores Locais por Nó	19
2.2.2.2	Classificador Local por Nó-Pai	22
2.2.2.3	Classificador Local por Nível	22
2.2.3	Classificador Global	23
2.2.4	Correção de Inconsistência	23
2.2.5	Bloqueio em Problemas Não MLNP	24
2.3	Considerações Finais	26

3	Classificação Hierárquica Desbalanceada	27
3.1	Trabalhos relacionados	28
3.1.1	Exploração da Hierarquia	29
3.1.2	Tratando o Problema de Desbalanceamento	30
3.1.3	Medidas e Avaliação de Desempenho	34
3.2	Considerações Finais	37
4	Atividades Realizadas	39
4.1	Experimentos em Conjuntos de Dados Binários	39
4.1.1	Motivação	40
4.1.2	ClusterOSS	42
4.1.2.1	O Algoritmo	42
4.1.2.2	Exemplo Ilustrativo	42
4.1.3	Resultados Experimentais	44
4.1.3.1	Configurações Utilizadas	44
4.1.3.2	Resultados e Discussões	47
4.2	Resultados Experimentais em Conjuntos de Dados Hierárquicos	49
4.2.1	Motivação	49
4.2.2	Experimentos realizados	50
4.2.2.1	Conjuntos de Dados	51
4.2.2.2	Medidas de Avaliação	53
4.2.2.3	Configurações	54
4.2.2.4	Resultados e Discussões	54
4.3	Considerações Finais	74
5	Conclusão	75
5.1	Principais Contribuições	76
5.2	Trabalhos Futuros	77
5.3	Publicação	77
	Referências	85

Lista de Figuras

1.1	Exemplo de classes separáveis	4
1.2	Exemplo de classes sobrepostas	4
1.3	Estrutura em árvore (à esquerda) e DAG (à direita)	5
1.4	Hierarquia de audio Burred e Lerch (2003)	6
2.1	Ilustração de <i>EasyEnsemble</i>	11
2.2	Exemplo de Distribuição de Dados	12
2.3	Criação do Elemento C Através de SMOTE	13
2.4	Classificador plano induzido através de um algoritmo de classificação multi-classe	19
2.5	Classificador Local por Nó	20
2.6	Divisões de exemplos em exploração local por nó	21
2.7	Classificador Local por Nó Pai	23
2.8	Classificador Local por Nível	24
2.9	Classificador Global	25
2.10	Hierarquia de dois níveis	25
3.1	Ilustração de exemplo de SMOTE hierárquico	32
3.2	Exemplo de <i>Trimming Machine</i>	33
4.1	Etapas do OSS: a) Conjunto original b) Seleção aleatória c) Conjunto de Dados pré-processado.	41
4.2	Etapas do ClusterOSS: a) Conjunto original b) Seleção Informativa c) Conjunto de Dados pré-processado.	43
4.3	Conjuntos de Dados Artificiais.	46

4.4	Frequência de Melhores Desempenhos e Entre os 3 Melhores Desempenhos.	48
4.5	Divisão por irmãos em exploração local por nó	51
4.6	Divisão menos inclusiva em exploração local por nó	52
4.5	a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia de GPCR utilizando árvores de decisão	56
4.6	a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para GPCR com árvores de decisão	57
4.7	Árvore de decisão para aplicação de pré-processamento no conjunto GPCR com árvore de decisão considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico	58
4.8	Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para GPCR com árvore de decisão	59
4.9	a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia para GPCR com SVM	61
4.10	a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para GPCR com SVM	62
4.11	Árvore de decisão para aplicação de pré-processamento no conjunto GPCR com SVM considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico	63
4.12	Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para GPCR com SVM	64
4.13	a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia para EC com árvore de decisão	65
4.14	a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para EC com árvore de decisão	66
4.15	Árvore de decisão para aplicação de pré-processamento no conjunto EC com árvore de decisão considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico	67
4.16	Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para EC com árvore de decisão	68
4.17	a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia para EC com SVM	69
4.18	a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para EC com SVM	70
4.19	Árvore de decisão para aplicação de pré-processamento no conjunto EC com SVM considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico	72
4.20	Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para EC com SVM	73

Lista de Tabelas

2.1	Tabela de Custo de Classificação Multi-classe.	16
2.2	Abordagens para treinamento em exploração local por nó	22
3.1	Trabalhos correlatos que abordam a classificação hierárquica com dados desbalanceados	28
3.2	Contexto e bases de dados utilizadas	28
3.3	Tipos de exploração da hierarquia	29
3.4	Abordagens utilizadas nos experimentos	34
3.5	Métodos e Medidas de Avaliação	36
4.1	Informações dos Conjuntos de Dados	45
4.2	OSS x ClusterOSS	47
4.3	<i>SMOTE</i> x <i>ClusterOSS</i> com <i>sobreamostragem aleatória</i>	49
4.4	Tabela descritiva sobre as características das classes	55
4.5	Critérios para a aplicação de técnicas de pré-processamento para dados desbalanceados em classificação hierárquica	73

Introdução

1.1 Contextualização

Os recentes avanços da ciência e tecnologia, de forma geral, viabilizaram o crescimento de dados em quantidade e disponibilidade. Junto com esse crescimento, surgiu a necessidade de analisar esses dados para descobrir conhecimento novo e útil. Assim, áreas que visam extrair conhecimento de conjuntos de dados ganharam grandes oportunidades para avanços de pesquisa, como a Inteligência Artificial (IA). O Aprendizado de Máquina (AM), que é um sub-campo da IA, surgiu com o objetivo de se dedicar ao desenvolvimento de algoritmos e técnicas que permitam ao computador aprender, isto é, que permitam ao computador aperfeiçoar seu desempenho em alguma tarefa. Outra sub-área da IA que também merece destaque é a Mineração de Dados (MD), que tem o objetivo de extrair conhecimento através da exploração de grandes quantidades de dados à procura de padrões consistentes para detectar relacionamentos sistemáticos entre esses padrões, detectando assim novos subconjuntos de dados.

Um problema comumente encontrado em todas essas áreas, e investigado neste trabalho de mestrado, é a tarefa de classificação de dados, que visa a identificar qual classe um determinado dado pertence. Essa tarefa pode ser definida formalmente como um problema de se encontrar um modelo que, dado um conjunto de pares de treinamento

(T_i, y_i) , mapeie cada indivíduo T_i em sua respectiva classe y_i , tal que $i = 1, 2, \dots, m$, sendo m o tamanho do conjunto de treinamento.

Tarefas de classificação podem ser encontrados em todas as área do conhecimento humano. Na medicina, por exemplo, essa tarefa é comumente utilizada para predizer se um tumor é benigno ou maligno. Na área de processamento de texto, problemas de classificação permitem categorizar textos como da área de finanças, previsão de tempo, esportes, cultura, etc. Na área financeira, pode-se utilizar tarefas de classificação para definir certas transações de cartões de crédito como legítimas ou fraudulentas. O fato é que, classificar é fundamental para a atividade humana e o desenvolvimento de sistemas computacionais que permitam realizar essas tarefas de forma automática é imprescindível.

1.2 Desafios e Definição do Problema

Desenvolver um modelo (algoritmo) de classificação pode não ser uma tarefa fácil e trivial. O fato é que, algumas limitações podem prejudicar a acurácia de um algoritmo de classificação, entre eles está o desbalanceamento da quantidade de exemplos nas classes de um conjunto de dados. Os algoritmos tradicionais acabam gerando modelos com dificuldade de classificação nas classes com poucos representantes (classes minoritárias). Dessa forma, técnicas de balanceamento artificial de dados, modificação de algoritmos, possíveis causas e propostas de formalismos para dados desbalanceados têm sido objetos de pesquisa nos últimos anos.

Uma área pouco explorada sob a visão do desbalanceamento de dados são os problemas de classificação hierárquica. Diferentemente dos problemas tradicionais (chamados de problemas de classificação plana ou *flat*), em problemas desse tipo, as classes são organizadas em hierarquias, normalmente na forma de árvore ou DAG (*Direct Acyclic Graph* – Grafo de Direção Acíclica). Ou seja, nesses problemas mais complexos, os exemplos se relacionam em estruturas compostas por superclasses e subclasses.

Dependendo da abordagem utilizada para enfrentar um problema hierárquico, é possível se deparar com dados desbalanceados. Além disso, quanto mais específica é a classe sendo analisada (quanto mais fundo se vai na estrutura hierárquica), mais raros são os representantes das classes e assim, mais desbalanceado é o problema. Essa dissertação de mestrado aborda a questão de dados desbalanceados em classificação hierárquica.

1.2.1 Distribuição Desbalanceada de Classes

Um conjunto de dados é dito desbalanceado quando nele existe uma clara desproporção entre número de exemplos de uma ou mais classe em relação às demais classes. Pode-se

observar, por exemplo, em um estudo de caso de uma determinada doença rara em uma população, que o número de pessoas portadoras da doença é muito menor do que o número de não portadores, ou seja, existe uma grande desproporção entre o número de exemplos das classes. Alguns exemplos de casos reais são as detecções de fraudes em chamadas telefônicas (Fawcett e Provost, 1997) e transações realizadas com cartões de crédito (Stolfo e Chan, 1997), nas quais o número de operações legítimas é muito maior do que o de fraudulentas. Outros exemplos de classes desbalanceadas podem ser encontrados na literatura: reconhecimento de assinaturas (Souza e Tsang, 2010), diagnóstico médico (Braga et al., 2008; Moturu e Liu, 2010; Natowicz e Rouzier, 2008; Silva e Nunes, 2009; Sun e Wang, 2007), entre outros.

Em situações dessa natureza, os algoritmos de Aprendizado de Máquina (AM) tradicionais não têm conseguido obter classificadores satisfatórios, porque apesar dos exemplos das classes majoritárias (de maior proporção) serem classificados corretamente com grande frequência, normalmente os exemplos das classes minoritárias (de menor proporção) não são classificados corretamente. Ou seja, é dito que as classes majoritárias são favorecidas enquanto as classes minoritárias possuem baixa taxa de reconhecimento (Castro e Braga, 2011). Em grande parte das vezes, são estas as classes de maior interesse. Assim, o custo envolvendo erros de classificação da classe minoritária é normalmente maior do que os da classe majoritária.

Porém, o problema em questão não é causado apenas pelo desequilíbrio das quantidades dos representantes de cada classe. Um importante fator é a separabilidade das classes em questão (Prati e Monard, 2004b; Sun e Kamel, 2009). Considere a Figura 1.1a, que apresenta uma distribuição com classes desbalanceadas, porém separáveis. Nela, os exemplos da classe majoritária são representados por 'X' e os da minoritária por círculos. Provavelmente, um algoritmo de classificação conseguiria induzir um modelo que separe as diferentes classes da Figura 1.1a de forma satisfatória. Uma possível representação desse modelo é apresentada na Figura 1.1b, na qual, elementos que se encontram abaixo da reta pertencem à classe dos 'X' e os elementos acima pertencem à classe dos círculos.

Assim, uma distribuição de classes como a representada na Figura 1.1a não caracterizaria um problema para o processo de classificação, apesar de ser desbalanceada. Um exemplo de distribuição que pode caracterizar um problema para a classificação é quando em uma região do espaço de atributos ocorre uma sobreposição das classes. Assim, diferente da Figura 1.1a, a distribuição apresentada na Figura 1.2a pode provocar dificuldade na indução de um modelo adequado por um algoritmo de classificação. Nela, os exemplos da classe majoritária são representados por 'X' e os da minoritária por círculos. Além disso, a área destacada (escurecida) aponta uma região de sobreposição.

Figura 1.1: Exemplo de classes separáveis **Figura 1.2:** Exemplo de classes sobrepostas

Em um cenário de sobreposição com dados desbalanceados, a classificação correta de exemplos da classe minoritária é potencialmente prejudicada. Isso acontece porque os algoritmos de classificação tradicionais tendem a favorecer os exemplos da classe majoritária por serem de maior proporção na região. Assim, um provável modelo induzido por um algoritmo tradicional é representado na Figura 1.2b. Nela, elementos abaixo da reta são classificados como da classe majoritária e os acima da reta são classificados como da classe minoritária.

Dessa forma, muitos dos elementos da classe minoritária seriam classificados incorretamente enquanto haveria um alto índice de acerto para os elementos da classe majoritária. Um problema é que, na maioria dos casos, em cenários com classes desbalanceadas, a classe minoritária é a de maior interesse. Ou seja, deseja-se classificar corretamente os elementos da classe minoritária com um interesse muito maior do que classificar corretamente os da majoritária. Em situações dessa natureza, utilizando algoritmos de AM tradicionais, esse interesse não seria bem representado.

Alguns experimentos, com dados sintéticos e reais, podem ser encontrados na literatura da área (Japkowicz e Stephen, 2002)(Prati e Monard, 2004b). Estes trabalhos mostraram

que, fixando uma razão de proporção e variando o nível de sobreposição entre as classes, os níveis mais altos de sobreposição prejudicaram significativamente a quantidade de classificação correta principalmente para a classe minoritária. Além disso, os resultados demonstraram que distribuições com domínios linearmente separáveis não são afetados significativamente pelo desbalanceamento.

1.2.2 Classificação Hierárquica

Uma área pouco estudada sobre os efeitos do desbalanceamento de dados é a classificação hierárquica. Normalmente, os problemas descritos na literatura são de classificação plana. Neles, cada exemplo pertence a uma classe de um conjunto de classes finito, não considerando assim relacionamentos hierárquicos. No entanto, existem problemas em que classes são divididas em subclasses. Nesse caso, elas podem ser representadas em uma estrutura hierárquica, como uma árvore ou um grafo acíclico direcionado (DAG - *Directed Acyclic Graph*). A principal diferença entre esses dois tipos de estruturas é que em um DAG, um nó pode ter mais de um pai. A Figura 1.3 mostra um exemplo estruturado em árvore e outro em DAG. Esses problemas são conhecidos em AM como problemas de classificação hierárquica (Freitas e de Carvalho, 2007)

Figura 1.3: Estrutura em árvore (à esquerda) e DAG (à direita)

Em problemas de classificação hierárquica existe uma taxonomia de classes, ou seja, elas são organizadas de forma sistemática. De acordo com (Silla e Freitas, 2011), uma taxonomia de classes é definida sobre C , um conjunto finito de todas as classes do domínio de aplicação, e sobre a relação $\prec$, que representa um relacionamento do tipo “IS-A” (é um). Um relacionamento IS-A é definido como assimétrico, antirreflexivo e transitivo:

- O elemento de mais alto nível da hierarquia é o elemento “R”, raiz da árvore.
- $\forall c_i, c_j \in C$, se $c_i \prec c_j$ então $c_j \not\prec c_i$ (assimétrico)
- $\forall c_i \in C$, $c_i \not\prec c_i$ (antirreflexivo)

- $\forall c_i, c_j, c_k \in C, c_i \prec c_j \text{ e } c_j \prec c_k \text{ implica em } c_i \prec c_k$ (transitivo)

Alguns artigos da literatura abordam problemas multi-classe de forma hierárquica. Por exemplo, algoritmos podem agrupar classes gerando superclasses ou dividir classes em subclasses organizando-as em uma hierarquia. Soluções dessa natureza não tratam problemas de classificação hierárquica porque criam novas classes e não utilizam uma taxonomia preestabelecida. Um algoritmo de classificação, teoricamente, não cria novas classes porque esse é objetivo de algoritmos como os de clusterização (Silla e Freitas, 2011).

Diversos trabalhos com taxonomias preestabelecidas podem ser encontrados na literatura. As principais áreas de aplicação de classificação hierárquica são: categorização de texto (Soumen Chakrabarti e Indyk., 1998); predição de função de proteína (Ashburner et al., 2000; Clare e King, 2003; Costa et al., 2008); classificação de gênero musical (Barbedo e Lopes, 2007; Burred e Lerch, 2003). A Figura 1.4 apresenta a hierarquia de gêneros musicais utilizada em Burred e Lerch (2003).

Figura 1.4: Hierarquia de áudio Burred e Lerch (2003)

1.3 Objetivos

Existem diversos trabalhos com taxonomias preestabelecidas na literatura e muitas delas são desbalanceadas. Porém, pouco estudo especializado em desbalanceamento e em predição das classes minoritárias têm sido feito. Assim, o problema investigado neste trabalho de mestrado pode ser formulado pela seguinte pergunta:

“É possível melhorar a classificação de problemas hierárquicos com classes desbalanceadas por meio da melhora de desempenho nas classes minoritárias?”

O objetivo geral desse trabalho é minimizar os efeitos de dados desbalanceados em problemas de classificação hierárquica. Para que este objetivo geral fosse alcançado,

focou-se nas abordagens de pré-processamento das bases, visando promover uma classificação mais significativa das classes minoritárias. Para isso, técnicas para problemas de classificação binária com dados desbalanceados foram aplicadas em problemas de classificação hierárquica.

1.4 Contribuições

Em suma, as principais contribuições deste trabalho são destacadas a seguir:

- **Revisão Bibliográfica.** Foi realizada uma extensa revisão bibliográfica sobre dados desbalanceados em problemas de classificação hierárquica. Tal revisão bibliográfica é um trabalho inédito, já que não existe nenhuma contribuição na literatura listando uma coleção de trabalhos propostos que abordam o problema de desbalanceado em classificação hierárquica.
- **ClusterOSS.** É uma nova técnica proposta neste trabalho que permite realizar, de maneira efetiva, a tarefa de classificação binárias com dados desbalanceados. Essa nova proposta é inspirada no algoritmo OSS, porém possui uma acurácia de classificação superior e resultados comparáveis à técnica SMOTE.
- **Experimentos em bases hierárquicas.** Foram realizados avaliações e comparações de desempenho de diversas técnicas para dados desbalanceados aplicadas em problemas de classificação hierárquica. Esses experimentos permitiram concluir que a aplicação de técnicas para dados desbalanceados deve ser feita de forma cuidadosa, pois ela pode prejudicar a indução adequada de modelos. Foi possível concluir que características como representatividade da classe minoritária e tamanho da base de dados são informações a serem consideradas para decidir a aplicação ou não de técnicas para dados desbalanceados.

De fato, todas essas investigações e proposta permitiram uma contribuição profunda para o estado-da-arte no que se diz respeito ao problema de classificação hierárquica com dados desbalanceados.

1.5 Organização do Trabalho

O texto dessa dissertação está organizado da seguinte forma:

No Capítulo 2, uma revisão bibliográfica da área de dados desbalanceados e a área de classificação hierárquica. As principais abordagens para se minimizar o problema de

desbalanceamento de dados em problemas de classificação tradicionais são descritos e as diversas formas de se explorar a estrutura de um problema hierárquico são comentadas.

No Capítulo 3, uma visão das técnicas propostas na literatura para diminuir o efeito de desbalanceamento em problemas hierárquicos é comentada.

No Capítulo 4, os resultados experimentais obtidos são apresentados. A técnica proposta *ClusterOSS* é descrita e avaliada. Além disso, uma comparação de diferentes técnicas de pré-processamento para dados desbalanceados aplicadas a problemas hierárquicos é avaliada.

Fundamentação Teórica

Neste capítulo, as principais abordagens existentes na literatura para tratar o problema de dados desbalanceados e classificação hierárquica são apresentadas. A Seção 2.1 apresenta as principais técnicas para dados desbalanceados, tanto em abordagens de pré-processamento quanto em abordagens de adaptação de algoritmos tradicionais em AM. A Seção 2.2 apresenta as principais formas de explorar as relações hierárquicas durante o processo de classificação, bem como alguns problemas encontrados nesse contexto.

2.1 Dados Desbalanceados

O problema de aprender a partir de conjuntos de classes desbalanceadas tem sido estudado por vários pesquisadores (Pazzani e Brunk, 1994)(Ling e Li, 1998)(Kubat e Matwin, 1997)(Fawcett e Provost, 1997)(Weiss, 2004)(Han e Mao, 2005). As diversas abordagens estudadas nesses trabalhos podem ser divididas em duas linhas de pesquisa: pré-processamento de dados e adaptação de algoritmos. Essas duas linhas são discutidas a seguir.

2.1.1 Pré-processamento de Dados

Abordagens de pré-processamento de dados têm como objetivo balancear a distribuição das classes no conjunto de dados de treinamento por meio de mecanismos que alteram a distribuição original dos dados. Tais mecanismos incluem subamostragem do conjunto majoritário, sobreamostragem do conjunto minoritário ou uma combinação dessas duas técnicas.

Métodos de subamostragem (*undersampling*) removem elementos da classe majoritária a fim de promover o balanceamento. Os exemplos a serem eliminados podem ser escolhidos de forma aleatória (subamostragem aleatória) ou por meio de algum critério de seleção (subamostragem informativa).

Subamostragem aleatória funciona, basicamente, selecionando elementos da classe majoritária aleatoriamente e retirando-os do conjunto de treinamento. A quantidade de exemplos selecionados pode variar, porém normalmente ela é suficientemente grande para que a proporção entre as classes majoritária e minoritária seja 1:1 ao final do processo.

Um exemplo de subamostragem informativa é a técnica OSS (*One-sided Selection*), de (Kubat e Matwin, 1997), que cria um novo conjunto composto por todos os exemplos da classe minoritária e os exemplos mais representativos da classe majoritária. Para selecionar os exemplos mais representativos da classe majoritária, seleciona-se aleatoriamente apenas um elemento da classe majoritária. Com esse único elemento, juntamente com todo o conjunto minoritário, tenta-se classificar todos os exemplos conhecidos. Os exemplos classificados corretamente são considerados redundantes e são retirados do conjunto de treinamento. Assim, o exemplo escolhido aleatoriamente, juntamente com os elementos classificados de forma incorreta, são assumidos como os mais representativos da classe majoritária. Além disso, utiliza-se técnicas de limpeza de dados a fim de eliminar os exemplos de borda ou ruidosos. Originalmente, Kubat e Matwin (1997) utilizam *Tomek Links* (Tomek, 1976) para limpeza de dados. *Tomek Links* e OSS são apresentados na Definição 2.1.1 e no Algoritmo 2.1 respectivamente.

Definição 2.1.1. Considere a base $\{E_1, \dots, E_n\} \subset R_k$. O par (E_i, E_j) é chamado de um *Tomek Link* se E_i e E_j são de classes diferentes e não existe um E_l tal que $d(E_i, E_l) < d(E_i, E_j)$ ou $d(E_j, E_l) < d(E_i, E_j)$, na qual $d(x, y)$ é a distância entre x e y .

Outras duas técnicas frequentemente utilizadas são *EasyEnsemble* e *BalanceCascade*, propostas por Liu e Zhou (2006). Na técnica *EasyEnsemble*, o conjunto da classe majoritária é dividido em subconjuntos aleatórios de mesmo tamanho da classe minoritária. Em seguida, cada um desses subconjuntos é unido com os exemplos da classe minoritária, para assim, obter diversos conjuntos balanceados que podem ser usados em

Algoritmo 2.1 Algoritmo que implementa a técnica OSS

- 1: Seja S o conjunto de treinamento original
- 2: Faça C conter todos os elementos da classe minoritária de S e apenas um elemento da classe majoritária escolhido aleatoriamente
- 3: Classifique S utilizando o algoritmo KNN (com $k=1$) e os elementos de C . Insira em C todos os elementos classificados incorretamente.
- 4: Remova de C todos os elementos da classe majoritária que participam no *Tomek Links*.

um ensemble. A Figura 2.1 representa graficamente esse processo. Inicialmente, os dados estão desbalanceados (1). Posteriormente, os exemplos da classe majoritária são divididos (2) e unidos com os da minoritária (3), para finalmente, serem gerados modelos para o *ensemble* (4).

Figura 2.1: Ilustração de *EasyEnsemble*

Similar ao *EasyEnsemble*, porém de forma supervisionada, *BalanceCascade* elimina do conjunto majoritário os exemplos que são classificados corretamente a partir dos subconjuntos balanceados gerados. Assim, busca-se excluir os representantes redundantes da classe de maior proporção. Para isso, um subconjunto dos exemplos da classe majoritária é selecionado e unido com os da minoritária. Eles são utilizados para gerar um modelo que tenta classificar todo o conjunto majoritário. Os elementos corretamente classificados são retirados do conjunto original. O processo se repete até um que um determinado critério de parada seja atingido.

Técnicas de sobreamostragem (*oversampling*) adicionam exemplos na classe minoritária com a finalidade de balanceamento. Essas técnicas podem replicar representantes da classe minoritária (sobreamostragem com repetição), de forma aleatória ou informativa, ou gerar dados artificialmente.

Para o processo de sobreamostragem aleatória, um subconjunto de elementos da classe minoritária é selecionado aleatoriamente, ou o conjunto minoritário inteiro é selecionado, e este é replicado para o conjunto de treinamento. Esse processo é repetido até que se tenha a proporção de exemplos desejada, que normalmente é 1:1.

SMOTE (*Synthetic Minority Oversampling Technique*) é uma técnica proposta por Chawla et al. (2002) que gera dados artificiais por meio de interpolação. Ela utiliza o algoritmo KNN para criar representantes entre exemplos e seus k vizinhos. Para isso, a cada iteração, um exemplo da classe minoritária é escolhido e o vetor obtido pela diferença entre o elemento em consideração e seu vizinho é multiplicado por um número aleatório entre 0 e 1. O novo vetor é aplicado sobre o elemento em consideração, selecionando um ponto no espaço, que fica na reta entre o elemento e seu vizinho. Neste ponto, o novo representante da classe minoritária é criado. Esta etapa é representada pela Equação 2.1, na qual x_i é o exemplo selecionado, $\hat{x}_i$ é seu vizinho mais próximo e γ é o número aleatório. Para ilustrar, considere a Figura 2.2 abaixo com um conjunto de dados em um espaço de dois atributos, no qual os 'X' são da classe majoritária e os círculos da minoritária.

$$x_{novo} = x_i + (\hat{x}_i - x_i) \cdot \gamma \quad (2.1)$$

Figura 2.2: Exemplo de Distribuição de Dados

Considere agora que o exemplo selecionado seja o exemplo A, representado na Figura 2.3, e que seu vizinho escolhido seja o exemplo B. Assim, um possível novo exemplo a ser criado é o representado por C, que necessariamente se encontra na semi-reta (na figura, tracejada) entre A e B.

Este processo é repetido n vezes, em que n é o número de elementos que se deseja gerar a partir da classe minoritária. O número de vizinhos, k , é definido pelo usuário.

Figura 2.3: Criação do Elemento C Através de SMOTE

O SMOTE desconsidera a vizinhança entre as classes e por isso pode gerar um aumento de sobreposição entre elas (Castro e Braga, 2009; He e Garcia, 2009). Assim, a fim de evitar esse problema, diversas adaptações foram propostas na literatura (Han e Mao, 2005; He et al., 2008).

Em Han e Mao (2005) uma adaptação de SMOTE é proposta, chamada de *Borderline - SMOTE* (SMOTE de Borda), na qual somente os exemplos minoritários das bordas da vizinhança entre classes são utilizados para gerar novos exemplos sintéticos. Considere que o conjunto de treinamento é T , o conjunto minoritário é P e o conjunto majoritário é N . Então, para cada exemplo $p \in P$, são calculados seus m exemplos mais próximos. Desses m , a quantidade de exemplos da classe majoritária é chamada de m' ($0 \leq m' \leq m$). Se $m/2 \leq m' < m$, p é considerado como de difícil classificação e é rotulado como *DANGER* (PERIGOSO). Se $m = m'$, p é considerado ruidoso e se $0 \leq m' < m/2$, p é considerado de fácil classificação. Nos dois últimos casos, p não é utilizado para a próxima etapa, que é a geração de exemplos artificiais. A geração de exemplos artificiais ocorre de forma muito similar ao SMOTE, porém apenas os exemplos rotulados como *DANGER* são sobreamostrados utilizando seus vizinhos mais próximos

Em He et al. (2008) o ADASYN, uma abordagem de SMOTE adaptativo, é proposta. Seu objetivo é ponderar a quantidade de exemplos artificiais gerados através da dificuldade de classificação de cada exemplo da classe minoritária. Este objetivo é atingido, primeiramente, calculando-se a quantidade total de exemplos minoritários a serem gerados. A Equação 2.2 representa essa quantidade, na qual m_n é quantidade de exemplos da classe majoritária, m_p é a quantidade de exemplos da classe minoritária e $\beta \in [0,1]$ especifica o nível de balanceamento após a criação dos exemplos.

$$G = (m_n - m_p) \cdot \beta \quad (2.2)$$

Então, para cada exemplo $p_i \in P$, na qual P é o conjunto minoritário, encontra-se os k vizinhos mais próximos e calcula-se a proporção de exemplos da classe majoritária vizinhos

de p_i . Ela é calculada através da Equação 2.3, na qual Δ é o número de exemplos da classe majoritária dentre os k vizinhos mais próximos e Z é uma constante de normalização para que $\sum r_i = 1$.

$$r_i = \frac{\Delta_i/k}{Z}, i = 1, \dots, |P| \quad (2.3)$$

Assim, pode-se definir a quantidade de exemplos gerados para cada exemplo da classe minoritária através da Equação 2.4. O processo de criação de exemplos artificiais é idêntico ao SMOTE.

$$g_i = r_i \cdot G \quad (2.4)$$

A ideia principal do ADASYN é utilizar a densidade r para ponderar a quantidade de exemplos gerados para cada exemplo da classe minoritária. Assim, para os exemplos mais difíceis de se classificar (com o maior número de exemplos majoritários próximos), são criados mais exemplos artificiais.

Outra técnica utilizada para sobreamostragem é chamada de CBO (*Cluster-Based Oversampling* - Sobreamostragem Baseada em Grupos) (Jo e Japkowicz, 2004). Nela, busca-se melhorar o desempenho não só sobre desbalanceamento entre as classes, mas também sobre o desbalanceamento dentro da classe. O desbalanceamento entre as classes diz respeito a desproporção da quantidade de exemplos entre as classes, enquanto o desbalanceamento dentro da classe diz respeito a desproporção da quantidade de exemplos entre os subconjuntos que formam cada classe.

Para esta técnica, os exemplos das classes minoritária e majoritária devem ser *clusterizadas* (agrupadas) separadamente. Os autores utilizam o algoritmo *k-means*, porém sugerem que qualquer algoritmo de agrupamento pode ser utilizado. O algoritmo *k-means* funciona da seguinte maneira: k exemplos são escolhidos aleatoriamente como representantes de cada grupo. Os exemplos mais próximos de cada k representante, calculados através de alguma medida de distância pré-definida, formam um grupo. Então, os representantes são atualizados de forma que cada um seja a média dos exemplos pertencentes aos grupos que representam. Com estes novos representantes, os exemplos são rearranjados nos k grupos através do critério de menor distância novamente. Este processo é iterativo e termina quando não há mudança dos representantes ou quando um número n de iterações pré-definido é atingido.

Assim que cada grupo de cada classe é formado, é iniciado o processo de sobreamostragem. Para a classe majoritária, em todos os grupos formados com exceção do maior grupo, é feita uma sobreamostragem aleatória. Dessa forma, todos os grupos da classe majoritária teriam a mesma quantidade de exemplos que o maior grupo. Para

exemplificar, suponha que os exemplos da classe majoritária fossem agrupados em 4 grupos com 10, 10, 10 e 24 exemplos cada. Após o processo de sobreamostragem, os grupos ficariam com 24, 24, 24 e 24 exemplos. Considere que $tamanhomaj$ seja a soma de todos os exemplos da classe majoritária após o processo de sobreamostragem, ou seja, no exemplo $tamanhomaj = 96$ ($24 + 24 + 24 + 24$). Então, a classe minoritária passaria por um processo de sobreamostragem aleatória de forma que cada grupo tenha $tamanhomaj/N_{min}$, na qual N_{min} é a quantidade de grupos da classe minoritária formados pelo algoritmo de agrupamento. Assim, supondo que tenham-se formado 3 grupos de 2, 3 e 2 exemplos, como $tamanhomaj/N_{min} = 96/3 = 32$, os 3 grupos teriam 32 exemplos cada ao final da sobreamostragem. Dessa forma, os subconjuntos de cada classe são sobreamostrados e igualados fazendo com que não haja desbalanceamento dentro da classe.

As técnicas citadas, assim como outras técnicas já propostas para subamostragem e sobreamostragem, modificam o conjunto de dados original, introduzindo características que podem influenciar não só positivamente, mas também negativamente no aprendizado. Por exemplo, a subamostragem aleatória pode eliminar exemplos importantes para a caracterização da classe. As técnicas de subamostragem informativa visam descaracterizar menos a classe, selecionando representantes redundantes, ruídos e elementos das fronteiras que separam as classes. Porém, a escolha desses critérios de seleção não é uma tarefa trivial.

2.1.2 Adaptação de Algoritmos

Outra alternativa para conseguir melhores classificadores em cenários com classes desbalanceadas é a adaptação de algoritmos. Diferentemente das técnicas apresentadas anteriormente, a adaptação de algoritmos não busca alterar a base de dados conhecida a fim de se conseguir uma nova base balanceada ou um conjunto de bases balanceadas. Ela visa adaptar as já conhecidas técnicas de classificação que muitas vezes não são boas ferramentas em um cenário desbalanceado.

Algumas dessas propostas consideram associar custos à classificação incorreta de exemplos. Estas técnicas associam, em estruturas chamadas tabela de custo por exemplo, o erro de classificação com um determinado custo para qualquer elemento do conjunto de exemplos.

Considere que, para o caso de uma classificação binária, $C(+,-)$ é o custo de classificar erroneamente um elemento positivo (ou seja, o custo de gerar um falso-negativo) e $C(-,+)$ é o custo de classificar erroneamente um elemento negativo (ou seja, o custo de gerar um falso-positivo). No cenário de classificação com dados desbalanceados, na

maioria dos casos é dada maior importância para a detecção de elementos da classe minoritária, considerada aqui como positiva, do que para a detecção de elementos majoritários, considerados negativos. Assim, geralmente atribui-se um custo maior para falsos-negativos do que para falsos-positivos (ou seja, $C(+,-) > C(-,+)$). Normalmente, classificar corretamente um exemplo tem custo zero ($C(+,+) = C(-,-) = 0$). Esse conceito pode ser aplicado para os problemas multi-classe, considerando que $C(i,j)$ define o custo de classificar um exemplo da classe i como da classe j . A Tabela 2.1 mostra uma representação de tabela de custo multi-classe, na qual normalmente a diagonal principal é zero. Dessa forma, o aprendizado tem por objetivo minimizar o custo total de erros de classificação e a quantidade de exemplos classificados erroneamente em situações de alto custo de erro.

Tabela 2.1: Tabela de Custo de Classificação Multi-classe.

		Predição de Classificação			
		1	2	...	k
Classificação Verdadeira	1	$C(1,1)$	$C(1,2)$	...	$C(1,k)$
	2	$C(2,1)$	$C(2,2)$	...	$C(2,k)$
	...	...	...	...	...
	k	$C(k,1)$	$C(k,2)$	...	$C(k,k)$

Diversos algoritmos que utilizam custos de classificação, chamados de sensíveis a custo, foram propostos motivados pelo trabalho dos algoritmos AdaBoost (Freund e Schapire, 1997). São exemplos desses os algoritmos AdaC1, AdaC2 e AdaC3 (Sun e Wang, 2007). A ideia principal do AdaBoost é iterativamente atualizar pesos atribuídos a cada elemento do conjunto de treinamento. Assim, exemplos diferentes podem ter pesos diferentes no processo de classificação. Neste processo, utiliza-se classificadores fracos (*weak classifiers*), que são classificadores que obtêm desempenho de classificação pouco superior a classificadores aleatórios. Um exemplo deles é uma árvore de decisão de um nível.

A Equação 2.5 representa a atualização do Adaboost, na qual t representa a iteração atual; $\alpha_t = \frac{1}{2} \ln(\frac{1-\epsilon_t}{\epsilon_t})$ é o parâmetro de atualização do peso; $h_t(x_i)$ é a saída da predição da hipótese h_t sobre a entrada x_i ; ϵ_t é o erro da hipótese h_t sobre o conjunto de treinamento; e Z_t é o fator de normalização, assim $\sum D_{t+1}(i) = 1$.

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t h_t(x_i) y_i)}{Z_t} \quad (2.5)$$

Assim, Sun e Wang (2007), visando favorecer a classe minoritária em um cenário desbalanceado, atribuiu a cada elemento um custo de classificação. Exemplos da classe minoritária possuem um custo maior de classificação do que os da majoritária. Este custo, pode ser incluído de três maneiras diferentes na equação de atualização (Equação 2.5):

dentro da exponencial, fora da exponencial e em ambos (dentro e fora da exponencial). Essas equações estão representadas analiticamente a seguir pelas equações 2.6, 2.7 e 2.8 respectivamente.

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t} \quad (2.6)$$

$$D_{t+1}(i) = \frac{C_i D_t(i) \exp(-\alpha_t h_t(x_i) y_i)}{Z_t} \quad (2.7)$$

$$D_{t+1}(i) = \frac{C_i D_t(i) \exp(-\alpha_t C_i h_t(x_i) y_i)}{Z_t} \quad (2.8)$$

As equações 2.6, 2.7 e 2.8 correspondem aos algoritmos AdaC1, AdaC2 e AdaC3, respectivamente. Assim, procura-se iterativamente alterar a probabilidade de selecionar um exemplo mais caro (com maior custo) do conjunto de treinamento. O trabalho observou que a inclusão de custo, além de colocar um viés direcionado à classe minoritária, também aumenta a utilização de dados mais relevantes a cada nova hipótese.

Outra forma de alteração de algoritmos está relacionada à modificação da função de custo, que mapeia penalidades para predições incorretas. Desconsiderando a possibilidade de atribuição de custos diferentes para diferentes erros de classificação comentada anteriormente, as técnicas de AM, em geral, têm por objetivo minimizar o erro de classificação sobre o conjunto de treinamento. Com esse objetivo, foram feitas algumas propostas de modificação para obter critérios de decisão que classificam melhor a classe minoritária (Alejo e Sánchez, 2006; Kukar e Kononenko, 1998; Veropoulos e Cristianini, 1999). As redes neurais, por exemplo, geralmente procuram minimizar o somatório do quadrado dos erros, considerando custos uniformes de erros de classificação. Ao considerar custos dessa forma, pode-se desfavorecer a classificação de exemplos da classe minoritária. Assim, alguns dos trabalhos já feitos nessa área utilizam penalidades associadas aos erros (Kukar e Kononenko, 1998).

Outros estudos procuram adaptar redes neurais para dados desbalanceados, como em Alejo e Sánchez (2006); Castro e Braga (2009). Além deste algoritmo, outros também foram alterados, como a *Support Vector Machine* (SVM), em (Joachims, 2002; Karakoulas e Shawe-Taylor, 1999; Morik e Joachims, 1999; Veropoulos e Cristianini, 1999; Wu e Chang, 2005); entre outros.

Esta seção tratou das principais abordagens para minimizar o problema de bases com classes desbalanceadas. Foram tratadas questões como pré-processamento de dados, tanto de subamostragem como de sobreamostragem, e adaptação de algoritmos tradicionais de AM. A próxima seção tratará de questões pertinentes à classificação hierárquica, mais

especificamente com relação as diferentes formas de explorar a estrutura hierárquica no processo de classificação.

2.2 Classificação Hierárquica

Existem duas estruturas que um problema de classificação hierárquica admite: árvore e DAG. A diferença entre elas é o fato de nós de DAG serem passíveis de possuir mais de um nó pai. De acordo com Freitas e de Carvalho (2007), e Sun (2001), este é apenas um dos critérios de um método de classificação hierárquico. Outro critério é com relação a profundidade na hierarquia em que ocorrerá a classificação. Um método pode sempre classificar novos exemplos como nós folhas, conhecidos como MLNP - predição de nó folha obrigatório (*mandatory leaf-node prediction*), ou classificar novos exemplos em qualquer outro nó da estrutura e em qualquer nível, conhecidos como non-mandatory leaf-node prediction (predição de nó folha não obrigatório).

O terceiro critério é de como a estrutura hierárquica é explorada. Este critério pode ser dividido em três diferentes tipos: classificadores planos (*flat*), desconsiderando a relação entre as classes; classificadores locais; e classificadores globais (*big-bang*), os quais um único classificador lida com a hierarquia inteira.

A seguir será descritos as diferentes formas de exploração da estrutura hierárquica. Algumas dessas explorações apresentam o problema de inconsistência, que é descrito posteriormente.

2.2.1 Classificadores Planos

A utilização de Classificadores Planos pode ser considerada a abordagem mais simples para tratar problemas de classificação hierárquica, porque funciona de forma similar a um classificador multi-classe tradicional. Normalmente são utilizadas apenas as classes dos nós-folhas, sendo que cada uma é uma classe diferente. Dessa forma, pode-se utilizar qualquer abordagem multi-classe. Uma vez que um novo exemplo é rotulado em um dos nós-folhas, a etapa seguinte diferencia o processo de classificação ao utilizar a regra do caminho verdadeiro (*true path rule*), regra que é utilizada para implicar os antecessores de um exemplo. A Figura 2.4 ilustra essa abordagem. Nela, o classificador é representado pelo contorno tracejado e as classes que se encontram dentro desse contorno são as utilizadas pelo classificador. Assim, supondo que um exemplo é rotulado como pertencente à classe 1.1.2, pela regra do caminho verdadeiro, ele também pertence às classes 1.1 e 1.

Figura 2.4: Classificador plano induzido através de um algoritmo de classificação multi-classe

Apesar de simples, essa abordagem tem a desvantagem de não explorar a relação entre os nós pais e filhos, sendo assim, obrigada a utilizar classificadores para prever entre as diferentes classes de nós-folhas.

2.2.2 Classificadores Locais

Diferente da classificação plana, classificadores locais consideram informações provenientes da hierarquia, porém com uma perspectiva local. Existem diversas formas na literatura de abordar essa perspectiva. Elas podem ser divididas em três categorias: classificadores locais por nó (LCN – *Local Classifier per Node*); classificadores locais por nó-pai (LCPN – *Local Classifiers per Parent Node*); e classificadores locais por nível (LCL – *Local Classifiers per Level*). Cada uma dessas categorias será descrita nas próximas subseções.

2.2.2.1 Classificadores Locais por Nó

Classificadores locais por nó são os mais utilizados na literatura (Silla e Freitas, 2011). Neles, para cada nó da estrutura hierárquica, exceto a raiz, treina-se um classificador binário. Uma representação dessa abordagem está na Figura 2.5, na qual cada quadrado com linha tracejada representa um classificador binário.

As divisões da base de dados entre os conjuntos positivos e negativos de cada classificador pode variar. Assim, pode-se separar em diferentes políticas cada forma diferente de divisão. Eisner e Greiner (2005) definem quatro abordagens diferentes que são exclusiva, menos exclusiva, menos inclusiva e inclusiva. Elas são descritas a seguir.

Figura 2.5: Classificador Local por Nó

A divisão “exclusiva” considera como positivos apenas os exemplos cuja classe mais específica é a classe em questão, e como negativo todo o resto. Considere o classificador do nó 1.2 da Figura 2.5. Apenas os exemplos cuja classe mais específica é 1.2 serão utilizados como positivos, enquanto os exemplos das classes 1, 1.1, 1.1.1, 1.1.2, 1.2.1, 1.2.2, 2, 2.1 e 2.2 são negativos.

A divisão “menos exclusiva” também considera apenas os exemplos da classe mais específica como positivos, porém os descendentes desse nó não compõe o conjunto dos negativos. Considere o classificador da classe 1.2, para ele apenas os exemplos cuja classe mais específica é 1.2 são positivos e os da 1, 1.1, 1.1.1, 1.1.2, 2, 2.1 e 2.2 são negativos.

A divisão “menos inclusiva” considera que os exemplos positivos são não só os elementos cuja classe mais específica é a de interesse, porém os das classes descendentes também. Assim, considerando o modelo da classe 1.2, os exemplos positivos são os das classes 1.2, 1.2.1 e 1.2.2. Os negativos são os das classes restantes, 1, 1.1, 1.1.1, 1.1.2, 2, 2.1 e 2.2.

A divisão “inclusiva” considera que os exemplos positivos (como na “menos inclusiva”) são os da classe de interesse e classes descendentes. Os exemplos negativos desconsideram a classe de interesse, as classes descendentes e as classes ancestrais. Assim, tomando como exemplo a classe 1.2 novamente, os exemplos positivos são da classe 1.2, 1.2.1 e 1.2.2 e os negativos são das classes 1.1, 1.1.1, 1.1.2, 2, 2.1 e 2.2.

Em seu trabalho, Eisner e Greiner (2005) observou que as divisões inclusiva e menos inclusiva obtiveram melhores performances na medida de avaliação *F-measure*.

Em Fagni e Sebastiani (2007), outras duas abordagens foram propostas para divisão dos exemplos positivos e negativos. Elas consideram os irmãos dos nós do classificador em questão. A divisão de “irmãos” considera os exemplos da classe e seus descendentes como

positivos e os irmãos (nós filhos do mesmo nó pai) com seus descendentes como negativos. Assim, para o modelo da classe 1.2, os exemplos positivos são os da classe 1.2, 1.2.1 e 1.2.2 enquanto os negativos são apenas os da classe 1.1, 1.1.1 e 1.1.2.

Para “irmãos exclusivos” somente o nó da classe de interesse para o classificador e seus nós irmãos são considerados. Para os exemplos positivos, são selecionados os exemplos cuja classe mais específica é a classe em questão, e como negativos os exemplos da classe dos nós-irmãos. Como exemplo, o classificador da classe 1.2 tem como conjunto positivo apenas os exemplos da classe 1.2 e, como negativos, os exemplos de 1.1.

Os exemplos são ilustrados na Figura 2.6.

Figura 2.6: Divisões de exemplos em exploração local por nó

As distribuições dos elementos positivos e negativos para o exemplo da classe 1.2 utilizado estão sumarizadas na Tabela 2.2.

Tabela 2.2: Abordagens para treinamento em exploração local por nó

Abordagem	Exemplos Positivos	Exemplos Negativos	Exemplos Não Utilizados
Exclusiva	1.2	Não 1.2	-
Menos Exclusiva	1.2	Não 1.2 + Não Descendentes(1.2)	Descendentes(1.2)
Menos Inclusiva	1.2 + Descendentes(1.2)	Não 1.2 + Não Descendentes(1.2)	-
Inclusiva	1.2 + Descendentes(1.2)	Não 1.2 + Não Descendentes(1.2) + Não Ancestrais(1.2)	Ancestrais(1.2)
Irmãos	1.2 + Descendentes(1.2)	1.1 + Descendentes(1.1)	Ancestrais(1.2)
Irmãos Exclusivos	1.2	1.1	Não 1.2 + Não 1.1

O trabalho de Fagni e Sebastiani (2007) compara sua política de irmãos com a política menos inclusiva. A observação feita é de que não há melhora significativa da acurácia da abordagem de irmãos sobre a menos inclusiva, porém a primeira utiliza menos exemplos.

Um problema que pode ser encontrado na abordagem por nó é inconsistência (Silla e Freitas, 2011). Para exemplificar esse conceito, considere que um elemento, após ser testado contra todos os modelos, tenha uma saída como sendo pertencente à classe 2.2 mas não pertencente à classe 2. Isso é possível porque os modelos em uma abordagem local por nó são treinados e testados independentemente.

2.2.2.2 Classificador Local por Nó-Pai

Na abordagem por classificador local por nó-pai, também conhecida como abordagem *top-down*, cada classificador é treinado para distinguir entre seus nós-filhos. Assim, diferente de classificadores locais por nó, o nó raiz também é um classificador e os nós folhas não são. Assim, considerando a Figura 2.7, o classificador da raiz identifica se o novo exemplo pertence à classe 1 ou 2. Supondo que pertença à classe 2, somente o classificador do nó 2 fará a classificação do próximo nível, indicando se ele pertence à classe 2.1 ou 2.2. Dessa forma, impede-se o problema de inconsistência.

2.2.2.3 Classificador Local por Nível

Nesse tipo de classificador local, um modelo multi-classe é criado para cada nível da hierarquia. Assim, um novo exemplo é classificado de acordo com todos os modelos. A Figura 2.8 apresenta uma representação para essa abordagem.

Nota-se que, como na abordagem Local por Nó, a classificação Local por Nível não está livre de inconsistência, ou seja, um novo exemplo pode ser classificado como pertencente à classe 1 e à classe 2.1. Uma maneira de lidar com este problema é limitar a classificação a apenas às classes dos nós filhos da classe selecionada no nível anterior. Assim, se o elemento foi classificado como pertencente à classe 1, no segundo nível, limita-se a escolha das classes a apenas 1.1 e 1.2.

Figura 2.7: Classificador Local por Nó Pai

2.2.3 Classificador Global

Uma outra abordagem é a de Classificadores Globais, também chamada de *big-bang*. Nela, diferente das abordagens locais que utilizam diversos modelos, um único modelo é utilizado. Geralmente, esse modelo único, apesar de mais complexo do que cada modelo local separado, é menor quando comparado com todos os modelos juntos utilizados pelas abordagens locais. Além disso, as dependências entre as classes e a estrutura hierárquica de forma geral, são consideradas de uma só vez pelo algoritmo de classificação, que, geralmente é uma adaptação de algoritmos tradicionais de AM (Silla e Freitas, 2011). A Figura 2.9 representa um classificador global.

Um exemplo de classificador global é encontrado em (Labrou, 1999). Inspirados em (Rocchio, 1971), a proposta baseia-se na ideia de *clusters*, na qual para cada novo exemplo, as distâncias dele para cada uma das classes é calculada para então ele ser classificado como pertencente à classe mais próxima. Outros exemplos são encontrados na literatura (Kiritchenko et al., 2006; Vens et al., 2008).

2.2.4 Correção de Inconsistência

Como mencionado nas seções anteriores, as abordagens de classificação locais por nó e por nível podem apresentar problema de inconsistência. Este problema ocorre quando, depois de combinar as saídas dos classificadores, é gerada uma saída impossível. Por exemplo, na figura 2.5, em uma situação de classificação local por nó, ou na figura 2.8, em uma situação de classificação local por nível, um exemplo pode ser classificado como pertencente a classe 1 e a classe 2.1.

Figura 2.8: Classificador Local por Nível

Uma abordagem simples é interromper a classificação uma vez que a inconsistência aconteça. Por exemplo, considere que o classificador da classe 2 retornou *verdadeiro* como saída, porém os classificadores das classes 2.1 e 2.2 retornaram *falso* como saída. O processo deve nesta abordagem, finalizar a classificação e classificar o exemplo como pertencente apenas à classe 2. Essa forma de classificação, iniciando pelos nós de níveis mais altos e descendo para os níveis mais baixos é chamada de *top-down*.

Outras duas abordagens são propostas em Dumais e Chen (2000), as quais envolvem *threshold*. A primeira delas consiste em calcular as probabilidades *a posteriori* ou *scores* de confiança das classes e somente considerar a classificação caso as probabilidades sejam maiores do que um determinado *threshold*. A segunda, utiliza um método multiplicativo, no qual leva em consideração a multiplicação das probabilidades *a posteriori* ou *scores* de confiança. Para a primeira abordagem, considere no exemplo da figura 2.10, as probabilidades ou *scores* sejam $p(c_1) = 0.6$, $p(c_2) = 0.2$, $p(c_{1.1}) = 0.55$, $p(c_{1.2}) = 0.1$, $p(c_{2.1}) = 0.2$ e $p(c_{2.2}) = 0.3$ para um determinado exemplo. Supondo um *threshold* de 0.5, o exemplo será classificado como pertencente às classes 1 e 1.1 por possuírem valores maiores do que o *threshold*. Na segunda abordagem, ele seria classificado como pertence à classe 1, porém não pertencente à classe 1.1, já que $p(c_{1.1}) * p(c_1) = 0.33 < 0.5$.

Outros trabalhos podem ser encontrados na literatura (Valentini, 2009).

2.2.5 Bloqueio em Problemas Não MLNP

Um problema Não MLNP aceita que o classificador rotule um exemplo como pertencente a qualquer nó da hierarquia, não sendo necessário classificá-lo como uma classe de nó-folha.

Figura 2.9: Classificador Global

Figura 2.10: Hierarquia de dois níveis

Uma maneira simples de tratar problemas dessa natureza é definir *threshold* (limiares) para cada nó (Ceci e Malerba, 2007). Assim, se o nível de confiança de um classificador é menor do que o estabelecido pelo *threshold*, o processo de classificação encerra-se para aquele exemplo.

Utilizar *threshold* pode levar ao problema de bloqueio (Sun et al., 2004). Bloqueio ocorre quando o processo de classificação *top-down* decide que, em um determinado nível da hierarquia, o exemplo não pertence a nenhuma classe dos classificadores em questão. Assim, a classificação é bloqueada e não se pode classificar o exemplo com nenhum dos descendentes. A Definição 2.2.1 formaliza esse conceito.

Definição 2.2.1. *Seja $c_0, c_1, \dots, c_n$ a lista de classes do nó raiz até o nó folha c_n ; e seja $M_0, M_1, \dots, M_{n-1}, M_n$ a lista de classificadores que devem aceitar um exemplo "e" antes*

de "e" ser rotulado como c_n . Bloqueio acontece quando "e" é rejeitado por qualquer um dos classificadores $M_0, \dots, M_{n-1}$ da sub-árvore

Três abordagens para evitar o bloqueio são discutidas em (Sun et al., 2004):

- Método de Redução de *Threshold*:

Uma maneira de possibilitar que mais exemplos sejam permitidos de passar para os classificadores de níveis mais específicos na hierarquia é a redução dos valores de *Threshold*. O desafio dessa abordagem é como definir os valores para cada classificador.

- Método de Voto Restrito:

Apesar do Método de Redução de *Threshold* possibilitar que mais exemplos passem para níveis mais específicos, ainda assim esse método tem alto índice de bloqueio. De forma diferente, o Método de Voto Restrito liga um nó com seu nó neto através de um classificador secundário. Dessa forma, caso aconteça de algum nó bloquear a classificação, pode-se utilizar o classificador secundário a fim de permitir que o nó filho evite o bloqueio. Em outras palavras, pode-se evitar o bloqueio através da decisão do nó filho do nó que causou o bloqueio.

- Método de Extensão Multiplicativa:

Extensão multiplicativa é um método recursivo que, basicamente, divide a hierarquia original em hierarquias de dois níveis. Assim, multiplica-se os valores de confiança da classificação dos classificadores dos dois níveis e verifica-se se este é maior do que o valor de *threshold* definido.

Nesta seção, foram descritas as principais formas de explorar a estrutura de uma hierarquia em um processo de classificação. Apresentaram-se os classificadores planos, globais e os locais (estes com diversas formas de organizar os conjuntos positivos e negativos de cada classificador). Além disso, alguns dos problemas que podem ser encontrados durante a classificação hierárquica foram discutidos.

2.3 Considerações Finais

Este capítulo apresentou os principais conceitos e técnicas existentes na literatura para dados desbalanceados e classificação hierárquica de forma separada. Todavia, o desempenho de muitos problemas de classificação hierárquica podem sofrer negativamente com distribuições desbalanceadas de classes. No próximo capítulo, uma revisão de técnicas para tratar dados desbalanceados em classificação hierárquica é apresentada.

Classificação Hierárquica Desbalanceada

No Capítulo 2 foram discutidos os problemas e soluções para mitigar a influência do desbalanceamento de dados no processo de classificação plana, bem como as principais abordagens para tratar problemas de classificação hierárquica. O fato é que, muitos problemas de classificação hierárquica possuem distribuições de classes desproporcionais, tal como nos contextos de predição de função de proteínas e classificação de documentos. Alguns trabalhos da literatura apontam que este problema se agrava conforme se analisa os níveis mais profundos da hierarquia.

Neste capítulo, será apresentada uma revisão bibliográfica detalhada do estado-da-arte na área. Serão apresentados os trabalhos correlatos, bem como seus contextos de aplicação, as bases de dados utilizadas, a exploração da hierarquia, medidas e métodos de avaliação dos experimentos.

3.1 Trabalhos relacionados

Na literatura existem poucos trabalhos que abordam o problema de dados desbalanceados em classificação hierárquica. A tabela 3.1 apresenta quatro desses trabalhos que foram investigados e detalhados durante o desenvolvimento desse trabalho.

Tabela 3.1: Trabalhos correlatos que abordam a classificação hierárquica com dados desbalanceados

Identificação	Referência
T1	Chen et al. (2010)
T2	Wang e Gong (2008)
T3	Chen e Hu (2010)
T4	Wang e Lee (2007)

A Tabela 3.2 apresenta um resumo das informações de contexto e bases de dados dos trabalhos relacionados. Dentre os trabalhos levantados durante a revisão, os problemas de classificação em bioinformática e classificação de texto são os mais comuns em classificação hierárquica com dados desbalanceados. Mais especificamente, no contexto de bioinformática, T1 aborda predição de função de proteína e T3 predição de função de gene, enquanto que no contexto de classificação de texto, T2 aborda classificação de páginas Web e T4 categorização automática de texto.

As bases de dados utilizadas em T1 são *cellcycle*, *church*, *derisi* e *borat*. Elas vêm de duas estruturas hierárquicas diferentes: *funCat* - bases *cellcycle*, *church* e *derisi* - e *Gene Ontology(GO)* - base *borat*. Em T2, foram utilizadas a base *20newgroups* e um conjunto de páginas web de informações de produtos extraídas do portal de comércio eletrônico *www.alibaba.com*. Em T3, foram utilizadas as seguintes bases com estrutura *funCat* e informações sobre fungos: *seq*, *cellcycle*, *gasch1* e *expr*. Em T4, é utilizada a base *BDS&T*, que contém apenas textos sobre ciência e tecnologia extraídos do sistema de registros bibliográficos *OCLC WorldCat*.

Tabela 3.2: Contexto e bases de dados utilizadas

Identificação	Contexto	Bases de Dados
T1	Predição de Função de Proteína	<i>cellcycle</i> , <i>church</i> , <i>derisi</i> (<i>FunCat</i>) e <i>borat</i> (<i>GO</i>)
T2	Classificação de Páginas Web	<i>20newgroups</i> e <i>alibaba.com</i>
T3	Predição de Função de Gene	<i>seq</i> , <i>cellcycle</i> , <i>gasch1</i> e <i>expr</i>
T4	Categorização Automática de Texto	<i>BDS&T</i>

A seguir, as principais explorações da estrutura hierárquica utilizadas são apresentadas e discutidas.

3.1.1 Exploração da Hierarquia

Todos os trabalhos utilizam abordagens locais para explorar a hierarquia. Enquanto alguns utilizam exploração local por nó pai, outros trabalhos realizam a exploração local por nó.

T1 utiliza exploração local por nó. Nele, 5 divisões de seleção de exemplos de treinamento são utilizadas: exclusiva, menos inclusiva, inclusiva, irmãos e uma nova proposta chamada de irmãos inclusivos fracos. Nesta proposta, os exemplos positivos são os da classe de interesse e seus descendentes, enquanto os exemplos negativos são os de classes irmãs. A fim de evitar inconsistência, observa-se as predições de forma *top-down* durante a fase de teste. Ou seja, primeiramente seleciona-se a classe com maior confiança entre as classes de nível mais alto. Repete-se a operação com a subárvore enraizada na classe selecionada anteriormente. Além disso, compara-se threshold estático com dinâmico.

T3 utiliza exploração local por nó. Nele, a classificação de um exemplo depende em parte do classificador do nó em questão e dos classificadores dos nós filhos. Mais precisamente, a probabilidade do exemplo pertencer àquela classe depende 50% do classificador da classe e 50% das classes filhas. Além disso, propõe a utilização de pesos diferentes para cada nó filho. Esse peso é calculado através de uma validação cruzada de 5 *folds* durante a etapa de treinamento e tem o objetivo de evitar que uma classificação errada com probabilidade muito alta influencie demasiadamente na classificação final do exemplo. Durante a etapa de teste, as predições negativas são propagadas de forma *top-down* e as positivas de forma *bottom-up*.

O trabalho T2 descreve uma exploração por nível, enquanto T4 aplica uma exploração local por nó pai, que no trabalho é chamada de *pachinko machine*.

A tabela 3.3 resume os tipos de exploração da hierarquia utilizados nos trabalhos relacionados.

Tabela 3.3: Tipos de exploração da hierarquia

Identificação	Forma de Exploração
T1	local por nó
T2	local por nível
T3	local por nó
T4	local por nó pai

A seguir, as abordagens propostas e utilizadas para tratar o desbalanceamento em classificação hierárquica encontradas nos trabalhos relacionados são apresentadas.

3.1.2 Tratando o Problema de Desbalanceamento

As abordagens utilizadas nos trabalhos estudados são algorítmicas e de pré-processamento. T1 apresenta soluções de *threshold* dinâmico para o desbalanceamento entre níveis. T2 apresenta uma combinação de predições de dois classificadores diferentes com a finalidade de favorecer a classe minoritária. T3 apresenta uma versão hierárquica para a técnica binária SMOTE. T4 apresenta uma técnica de pré-processamento que altera a estrutura hierárquica promovendo uma nova estrutura mais balanceada.

Em T1, busca-se diminuir o efeito do desbalanceamento existente entre as classes de nível mais alto e as de nível mais baixo. Quanto mais fundo se entra na hierarquia, menor o número de exemplos positivos encontrados, aumentando assim o desbalanceamento entre os níveis. A etapa de predição é feita de forma *top-down* com *threshold*. Em caso de desbalanceamento entre níveis, um *threshold* estático, tradicionalmente fixado em 0.5, não permitiria que exemplos fossem classificados até classes mais específicas (de nível mais baixo), interrompendo a predição em classes mais gerais (de nível mais alto). Assim, o trabalho apresenta duas propostas de cálculo de *threshold* dinâmico.

A primeira delas é baseada em *pass rate*. *Pass rate* é a taxa de exemplos de uma classe c em relação a alguma classe $c_i \in Par(c)$, sendo $Par(c)$ o conjunto de classes pai de c . *Pass rate* é calculada de acordo com a Equação 3.1.

$$Passrate_{c_i \Rightarrow c} = \frac{|exs(c)|}{|exs(c_i)|} \quad (3.1)$$

O *threshold* da classe c θ_c é calculado através da Equação 3.2

$$\theta_c = \frac{\sum_{c_i \in Par(c)} \theta_{c_i} * Passrate_{c_i \Rightarrow c}}{|Par(c)|} \quad (3.2)$$

Quanto menor for o número de exemplos transferidos de c_i para c , menor o θ_c . Assim, conforme $Passrate_{c_i \Rightarrow c}$ decresce, o *threshold* também decresce. A Equação 3.2 pode ser usada tanto em estruturas DAG quanto árvores. Nas árvores, a equação se torna mais simples porque não há necessidade do somatório.

A segunda proposta de *threshold* é baseada em Utilidade e é uma adaptação de uma proposta descrita em Clare (2003). Considerando que os rótulos mais específicos na hierarquia são mais úteis aos especialistas, a equação da entropia foi alterada para calcular-se a Utilidade. A Equação 3.3 descreve a Utilidade da classe c .

$$Utilidade(c) = 1 - \frac{\log_2 tam_arvore(c)}{max} \quad (3.3)$$

Nela, $tam_arvore(c) = Desc(c) + 1$, sendo o tamanho do DAG com raiz c ; $max = \max_{c_i \in C} \log_2 tam_arvore(c_i)$, com a finalidade de normalização dos valores no intervalo $[0, 1]$. Nesta abordagem, o *threshold* da classe c θ_c é calculado através da Equação 3.4

$$\theta_c = \frac{\sum_{c_i \in Par(c)} \theta_{c_i}}{|Par(c)|} * (1 - Utilidade(c)) \quad (3.4)$$

Além disso, para tratar o desbalanceamento no conjunto de teste, classificadores SVM são treinados com diferentes pesos para a classe positiva. Os classificadores SVM com melhor desempenho no conjunto de treinamento são utilizados durante a etapa de teste.

Em T2, o algoritmo VOTEM é proposto e aplicado. Ele é uma combinação dos algoritmos SVM e BEV (*Bagging Ensemble Variation*) (Li, 2007). Em BEV, um comitê de classificadores balanceados é utilizado, de forma similar ao *EasyEnsemble*. A combinação entre os dois algoritmos consiste em um operador “ou” (*or*). Assim, caso o classificador SVM ou o BEV determine que o exemplo avaliado pertence à classe minoritária, este é rotulado como sendo da classe minoritária. Em outras palavras, basta que um dos dois classificadores determine-o como positivo, para ele ser considerado como tal.

O Algoritmo 3.2 descreve o processo de classificação do VOTEM. Nele, X_t é o exemplo que deseja-se rotular; C_d é a classe densa (majoritária); C_r é a classe minoritária; $D_1, D_2, \dots, D_n$ são divisões da classe majoritária, sendo que cada D_k possui aproximadamente o mesmo número de exemplos de C_r .

Algoritmo 3.2 Algoritmo que implementa a técnica VOTEM

Se $Classificador_SVM(X_t, C_d, C_r) = C_r$ **ou** $\sum_k Classificador_BEV(X_t, D_k, C_r) = C_r$
 $\geq \frac{k}{2}$
 Então rotular $X_t \in C_r$
 Senão rotular $X_t \in C_d$

Em T3, uma adaptação do SMOTE para problemas hierárquicos foi proposta. Ela consiste em gerar exemplos artificiais da classe minoritária, através do SMOTE, apenas para os nós-folhas para que assim, esses exemplos sejam reutilizados, compondo o conjunto de treinamento de seus respectivos nós pais.

A Figura 3.1 representa o SMOTE hierárquico. Nela, a taxonomia é representada pelas elipses ligadas pelas semi-retas e os exemplos gerados artificialmente pelo SMOTE são representados por retângulos. Assim, para as classes dos nós-folhas (classes 3, 4 e 6 da figura 3.1), são gerados artificialmente os conjuntos de exemplos e3, e4 e e6. Eles são reutilizados pelas classes dos nós pais, as classes 2 e 5, para as quais também são gerados exemplos através do SMOTE. Dessa forma, para cada classe de nós não folhas, a

sobreamostragem é a união dos exemplos gerados para aquela classe através do SMOTE com os exemplos gerados para os nós filhos.

Figura 3.1: Ilustração de exemplo de SMOTE hierárquico

Em T4, uma abordagem de pré-processamento da hierarquia, chamada de *trimming machine*, é apresentada. Nela, as classes definidas como minoritárias são reagrupadas e unidas com a classe pai, gerando classes mais densas. Além disso, subárvores compostas apenas de classes majoritárias são planificadas e reagrupadas em uma nova superclasse. As superclasses geradas passam a compor o segundo nível da hierarquia, e o terceiro nível é composto pelas classes majoritárias. Assim, a abordagem tem como entrada uma taxonomia desbalanceada e gera como saída uma taxonomia mais simples e mais balanceada. A seguir são apresentadas as etapas da *trimming machine* no Algoritmo 3.3 e uma aplicação em um exemplo.

Algoritmo 3.3 Algoritmo que implementa a técnica *Trimming Machine*

Selecione uma subárvore T de 2 níveis em U .

Se $\exists$ classe $c \in T$ tal que o número de exemplos em $c \leq H_m$

Então UNIR c na raiz de T

Se T é uma árvore completa

Então CORTAR T

Senão PLANIFICAR T

Repetir tudo até que o topo de U seja atingido

Gerar uma hierarquia virtual U' com as subárvores geradas pela operação CORTAR

Sendo U a taxonomia original. H_m o limiar para unir uma classe minoritária com sua raiz. Uma classe c é definida como minoritária se o número de exemplos em $c \leq H_m$, senão c é uma classe densa. H_c é o limiar para cortar subárvores. Ou seja, uma subárvore T é definida como completa se o número de classes densas em $T \geq H_c$. A operação UNIR une a classe filho com a classe pai. A operação PLANIFICAR torna irmãs todas

as classes em uma subárvore, inclusive o nó raiz. A operação CORTAR retira da árvore uma subárvore T . Assim, se o nó raiz de T é uma classe densa, ela é cortada também, senão ela é mantida na hierarquia. Todas as classes cortadas são concentradas em uma nova superclasse e em uma nova hierarquia U' .

Figura 3.2: Exemplo de *Trimming Machine*

A fim de ilustrar uma execução da *Trimming Machine*, considere a Figura 3.2. No exemplo, foram considerados $H_m = 3$ e $H_c = 3$. Na Figura 3.2a, é apresentada uma estrutura hierárquica desbalanceada. É possível observar que as classes minoritárias, ou seja, com número de exemplos menor ou igual a H_m , são representadas pelos blocos brancos enquanto as classes densas são representadas pelos blocos pretos. Inicialmente, a subárvore com raiz em C6 é selecionada. Como a classe C10 é minoritária, ela é unida com a classe C6. A classe passa a ter 2 exemplos e, como ainda é minoritária, ela é unida com C2 gerando uma classe densa C2' com 4 exemplos. Em seguida, a subárvore com raiz em C8 é selecionada e, sendo uma subárvore incompleta (número de exemplos é menor do que H_c), ela é planificada.

A Figura 3.2b representa a árvore após a aplicação das operações comentadas. Depois, a subárvore com raiz em C2' é selecionada e, sendo uma subárvore completa, ela é cortada

e suas classes agrupadas em B2. A subárvore com raiz em C4 também é uma subárvore completa e também é cortada, mantendo apenas a classe C4 na hierarquia, por ser minoritária. As classes cortadas são separadas em B3. Por fim, a subárvore com raiz em C1 é selecionada. A classe C4 é unida em C1, gerando C1' com 10 exemplos. O restante das classes da árvore é agrupado em B1.

A Figura 3.2c representa os grupos B1, B2 e B3 formados após os cortes. Os grupos formam superclasses e são organizadas em uma nova hierarquia virtual, mais simples e balanceada. A hierarquia virtual formada é representada em 3.2d.

A Tabela 3.4 resume as abordagens utilizadas nos experimentos.

Tabela 3.4: Abordagens utilizadas nos experimentos

Identificação	Abordagem
T1	Threshold dinâmico e diferentes pesos para a classe positiva
T2	Algoritmo VOTEM
T3	SMOTE hierárquico
T4	Trimming Machine

A seguir são apresentadas as metodologias e as medidas de avaliação dos experimentos realizados nos trabalhos relacionados.

3.1.3 Medidas e Avaliação de Desempenho

A metodologia de avaliação é parte fundamental para análise dos experimentos realizados. No contexto de dados desbalanceados em classificação hierárquica, é importante observar se a metodologia utilizada considera as relações hierárquicas entre as classes ou se considera o desempenho por nível da hierarquia. Além disso, é interessante que as medidas de desempenho permitam uma visão sobre o desbalanceamento.

Em T1, foram utilizados conjuntos fixos e separados para treinamento, validação e teste. Foram utilizadas três medidas de avaliação *flat*, Precisão, Revocação e F-measure; e três medidas hierárquicas, Precisão hierárquica, Revocação Hierárquica e F-measure hierárquico.

Para o cálculo das medidas *flat* são utilizadas as equações 3.5, 3.6 e 3.7 para Precisão, Revocação e F-measure respectivamente. Nelas, TP representa o número de exemplos positivos classificados corretamente, FP representa o número de exemplo negativos classificados incorretamente e FN representa o número de exemplos positivos classificados incorretamente.

$$P = \frac{TP}{TP + FP} \quad (3.5)$$

$$R = \frac{TP}{TP + FN} \quad (3.6)$$

$$F = \frac{2 * P * R}{P + R} \quad (3.7)$$

As medidas *flat* não levam em consideração um possível acerto parcial de uma classificação. Considere que um exemplo pertencente à classe 2.1 seja classificado como pertencente à classe 2. Apesar desta classificação não estar completa, ela não está completamente incorreta, já que pela regra do caminho verdadeiro, o exemplo pertence às classes 2 e 2.1. Assim, as medidas hierárquicas tem como objetivo considerar de alguma forma as predições incompletas.

São utilizadas adaptações das tradicionais medidas Precisão, Revocação e F-measure para contextos hierárquicos (Canada et al., 2006). Elas são descritas nas equações 3.8, 3.9 e 3.10, que representam as medidas para cada exemplo. Nelas, P_i é um conjunto que contém a classe predita e todos os ancestrais do exemplo i e T_i é um conjunto que contém o rótulo verdadeiro e todos os ancestrais do exemplo i .

$$hP = \frac{|P_i \cap T_i|}{|P_i|} \quad (3.8)$$

$$hR = \frac{|P_i \cap T_i|}{|T_i|} \quad (3.9)$$

$$hF = \frac{2 * hP * hR}{hP + hR} \quad (3.10)$$

Em T1, são utilizadas micro-médias de hP e hR . Elas são descritas pelas equações 3.11, 3.12.

$$hP = \frac{\sum_{i=1}^N |P_i \cap T_i|}{\sum_{i=1}^N |P_i|} \quad (3.11)$$

$$hR = \frac{\sum_{i=1}^N |P_i \cap T_i|}{\sum_{i=1}^N |T_i|} \quad (3.12)$$

Em T2, os dados são separados por *hold-out*. O conjunto de dados é separado aleatoriamente em 75% para treinamento e 25% para teste. As medidas *flat* de

desempenho adotadas foram Precisão, Revocação e F-measure. Além disso, uma medida hierárquica chamada de *H-loss* foi utilizada (Cesa-Bianchi et al., 2006). Ela é descrita na Equação 3.13. Nela, $c_1, \dots, c_N$ são coeficientes de custo fixos, sendo que cada custo está relacionado a uma classe; y' representa as predições e v representa os rótulos verdadeiros.

$$H - loss(y', v) = \sum_{i=1}^N c_i \{y'_i \neq v_i \quad \wedge \quad y'_j = v_j, j \in par(i)\} \quad (3.13)$$

O objetivo em utilizar a medida *H-loss* é avaliar o desempenho sem considerar propagação de erros para classes filhas. Ou seja, caso uma classe pai seja predita de forma incorreta, o desempenho não é prejudicado pela predição das classes filhas. Em T2, os coeficientes c foram definidos como $c_{raiz} = 1$ e $c_i = \frac{c_{par(i)}}{|irmaos(i)|}$, nos quais $par(i)$ é o nó pai de i e $irmaos(i)$ é o conjunto de classes irmãs de i , considerando o próprio i .

Em T3, os dados são separados por *hold-out* e foram utilizados conjuntos fixos de treinamento e teste. As medidas de avaliação utilizadas foram Precisão, Revocação, F-measure, Precisão Hierárquica, Revocação Hierárquica e F-measure Hierárquico. Diferente de T1, T2 adota a macro-média dos valores de medida hierárquicas, descritas pelas equações 3.14 e 3.15.

$$hP = \frac{1}{|D|} \sum_{i=1}^N \frac{|P_i \cap T_i|}{|P_i|} \quad (3.14)$$

$$hR = \frac{1}{|D|} \sum_{i=1}^N \frac{|P_i \cap T_i|}{|T_i|} \quad (3.15)$$

Em T4, os dados são separados por *hold-out*, sendo conjuntos fixos para treinamento e teste. A medida de desempenho utilizada foi similaridade (Ganesan et al., 2003), descrita pela Equação 3.16 e mede a similaridade entre dois nós. Nessa equação, a profundidade(c_i) é o número de arestas ligando o nó raiz a c_i e $LCA(c_i, c_j)$ (*Lowest Common Ancestor*) é o ancestral mais específico comum entre as classes c_i e c_j .

$$similaridade(c_i, c_j) = \frac{2 * profundidade(LCA(c_i, c_j))}{profundidade(c_i) + profundidade(c_j)} \quad (3.16)$$

A Tabela 3.5 resume os métodos e medidas de avaliação utilizados nos trabalhos.

Tabela 3.5: Métodos e Medidas de Avaliação

Identificação	Métodos de Avaliação	Medidas de Avaliação
T1	Hold-out	Precisão, Revocação, F-measure, Precisão hierárquica, Revocação Hierárquica e F-measure hierárquico.
T2	Hold-out	H-loss
T3	Hold-out	Precisão, Revocação, F-measure, Precisão Hierárquica, Revocação Hierárquica e F-measure Hierárquico
T4	Hold-out	Similaridade

3.2 Considerações Finais

Neste capítulo foi realizada uma descrição detalhada dos trabalhos encontrados na literatura que abordam o problema de dados desbalanceados em classificação hierárquica. Foram apresentados os principais contextos de aplicação em que esse problema é encontrado, os principais tipos de exploração da hierarquia utilizados neste cenário, as abordagens utilizadas para minimizar os efeitos do desbalanceamento e os métodos e medidas de avaliação comumente utilizados. Os principais contextos dos trabalhos abordados estão relacionados às áreas da bioinformática e classificação de textos. Uma parte dessas técnicas realizam explorações locais por nó e por nó pai e as abordagens variam entre algorítmicas e de pré-processamento. Eficientes medidas de avaliação para cenários hierárquicos também foram apresentados.

Atividades Realizadas

Este capítulo apresenta uma extensa avaliação experimental realizada com o objetivo de analisar a influência e o impacto de dados desbalanceados em classificação hierárquica. Para isso, os experimentos foram divididos em duas etapas: na primeira, o cenário de classificação binária é analisado e uma nova técnica para dados desbalanceados é proposta; na segunda, são aplicadas técnicas de dados desbalanceados em problemas de classificação hierárquica.

4.1 Experimentos em Conjuntos de Dados Binários

Um dos objetivos deste trabalho é investigar técnicas binárias de dados desbalanceados. Assim, esta seção apresenta uma comparação experimental de diversas técnicas binárias. Além disso, é apresentada uma nova proposta de adaptação da técnica *OSS* utilizando agrupamento de dados, aqui chamada de *ClusterOSS* (*OSS* com Agrupamento de Dados). Será mostrado que o *ClusterOSS* supera o *OSS* em desempenho nos conjuntos de dados utilizados e ainda, quando combinado com a subamostragem aleatória, é comparável a técnica SMOTE.

4.1.1 Motivação

A técnica *OSS* assume que é suficiente escolher apenas um exemplo de forma aleatória para iniciar todo o processo de seleção dos exemplos da classe majoritária mais significativos do conjunto em questão. Entretanto, essa escolha é de suma importância para a subamostragem e selecioná-lo de forma aleatória pode prejudicar o desempenho da técnica. Assim, para o *ClusterOSS*, um conjunto de exemplos da classe majoritária é selecionado de modo informativo (não aleatório). Para isso, os exemplos majoritários são agrupados por algum algoritmo de agrupamento e os exemplos mais próximos aos centroides dos grupos são selecionados.

Considere a Figura 4.1 que representa o pré-processamento feito pelo algoritmo *OSS* em um conjunto de dados gerado artificialmente, no qual, os triângulos vermelhos são os exemplos da classe minoritária e os círculos pretos os exemplos da classe majoritária. Note que o conjunto majoritário é distribuído em dois grupos no espaço de atributos, um em cada extremidade do conjunto minoritário. a) Conjunto de dados gerado artificialmente. b) é o conjunto da classe minoritária unido com o exemplo da classe majoritária escolhido aleatoriamente. c) é o conjunto pré-processado com os exemplos mais relevantes e sem os exemplos participantes do Tomek Links.

Nota-se que o processo de subamostragem foi prejudicado pelo fato do exemplo ter sido selecionado aleatoriamente em uma região distante da classe minoritária. Suponha que o exemplo tenha sido selecionado no centro da região majoritária da direita. Neste caso, a subamostragem teria obtido um bom efeito na região da direita, porém nenhum efeito na região majoritária da esquerda. Assim, *ClusterOSS* é um método que evita essas situações.

A técnica *ClusterOSS* é uma adaptação da estratégia utilizada em *OSS*. Antes da descrição algorítmica do *ClusterOSS*, são apresentadas as duas principais diferenças entre as técnicas *OSS* e *ClusterOSS*.

A primeira diferença é que *ClusterOSS* pode iniciar o processo de subamostragem através de mais de uma instância. Essa característica já aborda a desvantagem do *OSS* de ser dependente de apenas um exemplo escolhido. A segunda diferença é que a seleção dos exemplos não é feita de forma aleatória, e sim de forma informativa. Primeiro, o conjunto majoritário é agrupado através de uma técnica de agrupamento. Em seguida, são selecionados os exemplos mais próximos dos centróides dos subgrupos formados. Dessa forma, o efeito de subamostragem é melhorado, uma vez que o processamento ocorrerá em diferentes regiões do espaço de atributos.

Figura 4.1: Etapas do OSS: a) Conjunto original b) Seleção aleatória c) Conjunto de Dados pré-processado.

4.1.2 ClusterOSS

Com base na análise das situações não favoráveis da técnica *OSS* apresentada na seção anterior, nesta seção é apresentada uma nova abordagem, denominada *ClusterOSS*, visando ampliar o desempenho da técnica *OSS* em problemas de classificação binária com dados desbalanceados.

4.1.2.1 O Algoritmo

O técnica *ClusterOSS* é formalizada pelo Algoritmo 4.4. Primeiramente, o conjunto majoritário é agrupado utilizando alguma técnica de agrupamento, por exemplo *k-médias*. Em seguida, para cada subgrupo formado, o exemplo mais próximo do centro é selecionado. Então, é realizado o processo de subamostragem de forma idêntica ao *OSS*. Finalmente, a técnica *Tomek Links* é utilizada para realizar uma limpeza dos dados. No algoritmo, a função *ExemplosMajoritarios(D)* retorna os exemplos pertencentes à classe majoritária do conjunto D; *Agrupar()* retorna um conjunto de agrupamentos identificados; *SelecionarExemploProximoCentro(C_c)* seleciona o exemplo majoritário mais próximo do centro do subconjunto C_c ; *ExemplosMinoritarios(D)* retorna os exemplos pertencentes à classe minoritária do conjunto D; *KNN(Treino, Teste)* utiliza o conjunto Treino para classificar o conjunto Teste com um KNN, com $k = 1$; *ErrosDeClassificacao* retorna os exemplos classificados de forma incorreta; *TomekLinks()* retorna os exemplos pertencentes ao Tomek Links.

4.1.2.2 Exemplo Ilustrativo

Utilizando o mesmo conjunto de dados da Figura 4.1, o funcionamento da técnica *ClusterOSS* é apresentada através de um exemplo. Considere a Figura 4.2, na qual, os triângulos vermelhos são os exemplos da classe minoritária e os círculos pretos os exemplos da classe majoritária. a) é o conjunto de dados gerado artificialmente. b) é o conjunto da classe minoritária unido com os exemplos da classe majoritária mais próximos aos centros dos subgrupos. c) é o conjunto pré-processado com os exemplos mais relevantes e sem os exemplos participantes do Tomek Links.

Através das Figuras 4.1 e 4.2, é possível observar que a técnica proposta reduziu significativamente o tamanho do conjunto de dados. Isso porque, o algoritmo *OSS* permite que o exemplo seja escolhido aleatoriamente e possivelmente pertencente a uma área não interessante para a etapa que utiliza o algoritmo *KNN*, como regiões periféricas distantes dos exemplos minoritários, enquanto o *ClusterOSS* sempre seleciona exemplos centrais de concentrações majoritárias. Além disso, o *OSS* sempre seleciona apenas

Algoritmo 4.4 Algoritmo que implementa a técnica ClusterOSS

```

D <- Conjunto de Dados
Treino <- {}
Teste <- {}
ConjMajoritario <- ExemplosMajoritarios(D)
C <- Agrupar(ConjMajoritario)
para cada subgrupo  $C_c \in C$ 
    x <- SelecionarExemploProximoCentro( $C_c$ )
    Treino <- Treino  $\cup$  x
    Teste <- Teste  $\cup$  ( $C_c - \{x\}$ )
Treino <- Treino  $\cup$  ExemplosMinoritarios(D)
Resultado <- KNN(Treino, Teste)
Erros <- ErrosDeClassificacao(Resultado)
D' <- Treino  $\cup$  Erros
TLinks <- TomekLinks(D')
para cada exemplo z  $\in$  TLinks
    se z  $\in$  ConjMajoritario
        D' <- D' - {z}
retornar D'

```

Figura 4.2: Etapas do ClusterOSS: a) Conjunto original b) Seleção Informativa c) Conjunto de Dados pré-processado.

um exemplo, enquanto o *ClusterOSS* seleciona representantes centrais de diferentes regiões de concentração. O conjunto de dados original possui proporção de 1:40, enquanto as proporções dos conjuntos resultantes do exemplo por *OSS* e *ClusterOSS* são respectivamente 1:30 e 1:5 aproximadamente. É importante destacar que ambas as técnicas reduzem o conjunto majoritário apenas em regiões mais distantes das regiões minoritárias. Pode haver modificação das regiões de sobreposição apenas no momento de limpeza dos dados, como por exemplo com a técnica *Tomek Links*.

4.1.3 Resultados Experimentais

A seguir, é apresentada uma avaliação empírica da técnica *ClusterOSS*. O Objetivo é verificar se a técnica provê melhor desempenho de classificação quando comparada com *OSS*. *ClusterOSS* também é comparado com outras técnicas de pré-processamento da literatura: subamostragem aleatória, sobreamostragem aleatória, SMOTE, CBO e OSS. Além disso, *ClusterOSS* é combinado com sobreamostragem aleatória a fim de obter dados com uma distribuição mais equilibrada entre as classes.

4.1.3.1 Configurações Utilizadas

ClusterOSS foi implementado utilizando a técnica *k-médias* de agrupamento de dados. Para definir o número de grupos, foi utilizada a média de silhueta do conjunto de treinamento. Para o *k-médias*, consideramos a distância Euclidiana como medida de proximidade e 10 como número máximo de iterações. Para obter a média de silhueta, foi considerada a distância Euclidiana como medida de proximidade. Para o CBO, a mesma estratégia foi utilizada. Sobreamostragem aleatória e subamostragem aleatória fazem com que o conjunto fique com proporção final de 1:1. A técnica SMOTE foi combinada com subamostragem aleatória, como sugere seus autores. O SMOTE altera a distribuição do conjunto de treinamento de forma que aumenta em 200% o conjunto minoritário e subamostra o conjunto majoritário de forma que, ao final do processo, a classe minoritária representa 75% da classe majoritária. Três algoritmos de classificação diferentes foram aplicados para cada conjunto processado, sendo eles KNN(K=3), C5.0 e SVM.

Foram utilizados 10 conjuntos de dados durante a avaliação, que são apresentadas na Tabela 4.1. Na tabela consta o nome dos conjuntos, o número de atributos (incluindo o atributo alvo), número de exemplos e a proporção entre as classes.

Os conjuntos Vowel, Haberman, Pima Diabetes e Yeast foram obtidos através do repositório UCI (Bache e Lichman, 2014), e Cleveland, Poker e Vehicle através do repositório Keel (Alcalá-Fdez et al., 2011). Vowel, Yeast, Cleveland, Poker e Vehicle são originalmente problemas multi-classe e foram transformados em problemas binários

Tabela 4.1: Informações dos Conjuntos de Dados

Conjuntos	# Atributos	# Exemplos	Proporção
Artificial (a)	3	410	1 : 40
Artificial (b)	3	510	1 : 50
Artificial (c)	3	520	1 : 25
Vowel0	11	990	1 : 10
Haberman	4	306	1 : 3
Yeast4	8	1479	1 : 28
Pima Diabetes	9	768	1 : 1.86
Cleveland0x4	14	173	1 : 12.31
Poker8x6	11	1477	1 : 85.88
Vehicle2	19	846	1 : 2.88

escolhendo uma classe específica como positiva e relacionando com outra(s) como exemplos negativos. As relações classe positiva x classe(s) negativa(s) são 0 x demais para Vowel, 4 x demais para Yeast, 0 x 4 para Cleveland, 8 x 6 para Poker e 2 x demais para Vehicle, respectivamente.

Os conjuntos gerados artificialmente são utilizados com o objetivo de observar o desempenho dos pré-processamentos para diferentes situações. Elas foram geradas através de distribuições normais para cada subgrupo que as classes apresentam. Elas possuem três atributos, dos quais dois são numéricos e um é a classe, e além disso, são todas problemas de classificação binária. Elas são representadas na figura 4.3, na qual os 'X' pretos representam a classe majoritária e os círculos azuis a classe minoritária. O conjunto (a) possui duas concentrações de exemplos majoritários e uma concentração minoritária entre elas e sua proporção é de 1:40. O conjunto (b) possui uma concentração majoritária com uma concentração minoritária no centro dela e sua proporção é de 1:50. O conjunto (c) possui uma concentração majoritária com duas concentrações minoritárias em periferias opostas a ela e sua proporção é 1:25.

No experimento foi utilizado *k-fold cross-validation* (Validação Cruzada), com $k = 5$. A validação cruzada é feita 100 vezes. A amostragem foi feita de forma estratificada, ou seja, cada *fold* possui a mesma distribuição de classes do conjunto de dados original. O número de *folds* escolhido foi 5 para garantir que todos contêm ao menos um exemplo minoritário.

As medidas utilizadas nesse trabalho foram a acurácia da classe positiva (classe minoritária), a acurácia da classe negativa (classe majoritária) e a média geométrica entre elas. A equação da acurácia da classe positiva, da classe negativa e a média geométrica entre elas são apresentadas pelas equações 4.1, 4.2 e 4.3.

$$apos = \frac{TP}{FN + TP} \quad (4.1)$$

(a)

(b)

(c)

Figura 4.3: Conjuntos de Dados Artificiais.

$$aneg = \frac{TN}{TN + FP} \quad (4.2)$$

$$g = \sqrt{apos * aneg} \quad (4.3)$$

4.1.3.2 Resultados e Discussões

Primeiramente, o desempenho do *ClusterOSS* é comparado com o desempenho do *OSS*. A Tabela 4.2 resume os resultados mostrando o número de vitórias e empates das técnicas sobre os 10 conjuntos de dados e os 3 algoritmos de classificação. É possível observar que o *ClusterOSS* se sobressai na medida acurácia positiva, que é a acurácia sobre a classe de interesse. Na medida acurácia negativa, as duas técnicas obtêm números de vitória próximos, com uma pequena vantagem ao *OSS*. Esse *trade-off* entre perder em desempenho na classe negativa para melhorar o desempenho na classe positiva é esperado. Assim, quando observa-se a medida Média Geométrica entre as acurácias, que considera o desempenho em ambas as classes, o *ClusterOSS* se destaca em relação ao *OSS*.

Tabela 4.2: OSS x ClusterOSS

Medidas	# vitórias OSS	# vitórias ClusterOSS	Empates
Acurácia Positiva	4	19	7
Acurácia Negativa	13	12	5
Média Geométrica	5	20	5

A Figura 4.4 resume os resultados comparativos entre as 8 técnicas utilizadas nos 10 conjuntos de dados e nos 3 algoritmos de classificação. Nela, as barras azuis apresentam a porcentagem de vitórias da técnica de pré-processamento e as barras amarelas apresentam a porcentagem das vezes que a técnica esteve entre os 3 melhores desempenhos.

É possível observar que o conjunto de treinamento original, sem pré-processamento, é o que apresenta melhores resultados para a classe negativa, porém com os piores resultados para a classe positiva. Pode-se explicar com o fato de que o desbalanceamento promove um viés no classificador fazendo com que a classe minoritária seja desfavorecida.

A sobreamostragem aleatória apresenta comportamento inverso. Enquanto o seu desempenho é muito superior na classe positiva, seu desempenho na classe negativa fica muito aquém das outras técnicas no comparativo. Por esse fato, seu desempenho na média geométrica pode ser explicado. Nessa medida ele obteve um bom desempenho, porém ainda inferior a outras técnicas.

Apesar de ter sido mostrado que o *ClusterOSS* obtém melhor desempenho preditivo do que o *OSS*, nas análises comparativas ele não se destaca. Porém, quando *ClusterOSS* é

(a) Acurácia Positiva

(b) Acurácia Negativa

(c) Média Geométrica

Figura 4.4: Frequência de Melhores Desempenhos e Entre os 3 Melhores Desempenhos.

combinado com a sobreamostragem aleatória, ele se torna comparável ao SMOTE, estando entre as duas técnicas com melhor desempenho, juntamente com SMOTE.

A Tabela 4.3 apresenta uma comparação entre os resultados de SMOTE e ClusterOSS exclusivamente. Pode-se observar que, enquanto o SMOTE obtém melhores resultados para a classe positiva, ClusterOSS apresenta melhores resultados para a classe negativa. Na média geométrica das medidas, os dois resultados são comparáveis, com uma pequena vantagem para o SMOTE.

Tabela 4.3: *SMOTE x ClusterOSS com sobreamostragem aleatória*

Medidas	# vitórias <i>SMOTE</i>	# vitórias do <i>ClusterOSS</i> com sobreamostragem	Empates
Acurácia Positiva	18	10	2
Acurácia Negativa	9	21	0
Média Geométrica	16	14	0

4.2 Resultados Experimentais em Conjuntos de Dados Hierárquicos

O principal objetivo deste trabalho é investigar a aplicação de técnicas capazes de lidar com dados desbalanceados em tarefas de classificação hierárquica. Para isso, a tarefa de classificação hierárquica foi dividida em sub-tarefas de classificação binária, por meio de uma exploração local por nó, utilizando a divisão por irmãos.

Esta seção apresenta uma comparação de diversas técnicas binárias para dados desbalanceados aplicadas em tarefas de classificação hierárquica. De acordo com os resultados obtidos nos experimentos realizados, a aplicação desregrada de técnicas de pré-processamento pode, em alguns casos, degradar o desempenho de técnicas de classificação binária. Os resultados mostram ainda que algumas características de uma tarefa de classificação binária podem guiar a decisão da aplicação ou não de técnicas para tratar dados desbalanceados.

4.2.1 Motivação

A maior parte dos conjuntos de dados utilizados na área de biologia e na classificação de texto apresentam uma estrutura hierárquica de classes e sofrem de desbalanceamento nos dados pertencentes a cada uma das classes. Mais especificamente, dois conjuntos de dados de classificação de função de proteínas utilizadas neste trabalho, GPCR e EC, possuem estruturas hierárquicas e dados desbalanceados.

O desbalanceamento pode ocorrer por causa da diferença entre a quantidade de exemplos das classes, da diferença entre a quantidade de exemplos dos níveis e da própria estrutura hierárquica, na qual alguns ramos chegam a níveis mais baixos e outros não.

A maior parte das técnicas de pré-processamento para dados desbalanceados foram desenvolvidas para problemas de classificação binária. Por isso, neste trabalho foi utilizada a exploração local por nó, que gera sub-tarefas de classificação binária. A abordagem utilizada neste trabalho foi a abordagem de *divisão por irmãos*, por apresentar melhor desempenho preditivo combinado com menor número de exemplos utilizados (Fagni e Sebastiani, 2007).

A divisão por irmãos considera que os exemplos positivos são aqueles cuja classe mais específica é a classe de interesse e os exemplos das classes que descendem dessa classe. Os exemplos negativos são aqueles das classes irmãs e de suas classes descendentes. A Figura 4.5 mostra a divisão de exemplos positivos e negativos das classes 1, 1.2 e 1.2.2 utilizando a divisão por irmãos.

É possível observar que, nessa abordagem, as sub-tarefas de classificação binária associadas às classes de nível mais baixo tendem a ser menos desbalanceados do que em outras divisões, como a menos inclusiva. A divisão menos inclusiva para as classes 1, 1.2 e 1.2.2 é apresentada na Figura 4.6 para efeito de comparação. Nela, os exemplos ancestrais são incluídos na classe negativa, tornando os dados cada vez mais desbalanceados quando se desce na hierarquia.

O problema de desbalanceamento ainda existe na divisão por irmãos, mesmo ela apresentando um equilíbrio de classes maior do que as demais divisões. A Figura 4.5a apresenta o desbalanceamento no primeiro nível, no qual existem mais classes positivas e em maior profundidade. A Figura 4.5b apresenta um desbalanceamento no número de classes, apesar de não possuírem profundidades diferentes. A Figura 4.5c, apesar de não apresentar desbalanceamento no número de classes, pode apresentar no número de exemplos. Assim, este experimento objetiva utilizar técnicas de pré-processamento para dados desbalanceados a fim de melhorar o desempenho preditivo obtido na divisão por irmãos.

4.2.2 Experimentos realizados

A seguir, é apresentada uma avaliação experimental das técnicas sobreamostragem aleatória, subamostragem aleatória, SMOTE, CBO, OSS, ClusterOSS com sobreamostragem e ADASYN em tarefas de classificação hierárquica. O objetivo desse experimento é verificar se a aplicação de técnicas de pré-processamento provê um melhor desempenho

Figura 4.5: Divisão por irmãos em exploração local por nó

preditivo nessas tarefas. Foi utilizada exploração local por nó com divisão por irmãos e foram utilizados dois conjuntos de dados biológicos, que serão descritos a seguir.

4.2.2.1 Conjuntos de Dados

Nos experimentos foram utilizados dois conjuntos de dados biológicos com classes hierarquicamente estruturadas. O conjunto GPCR, que contém dados da família de proteínas *G-Protein-Coupled Receptor*, e ao conjunto EC, que contém dados de enzimas classificadas de acordo com a nomenclatura definida em *Enzyme Commission* (Hoffmann-Ostenhof, 1974).

Os modelos gerados a partir desses conjuntos de dados têm o objetivo de prever classes funcionais de uma proteína. Essa tarefa possibilita a inferência de funções de proteínas. Esses modelos são de grande utilidade na biologia, uma vez que proteínas pertencentes à mesma classe funcional possuem a mesma função ou funções similares ou estão envolvidas em processos biológicos relacionados.

GPCR - G-Protein-Coupled Receptor

Proteínas da família GPCR são importantes para estudos na área médica devido a sua influência em reações químicas no interior das células. As classes nessa hierarquia são dispostas em uma estrutura de árvore de cinco níveis. Foram considerados apenas os primeiros quatro níveis, excluindo o quinto nível da estrutura, uma vez que apenas

Figura 4.6: Divisão menos inclusiva em exploração local por nó

esses quatro níveis apresentam número de exemplos suficiente para o treinamento dos algoritmos de classificação utilizados.

O conjunto de dados GPCR possui originalmente 75 atributos e 7077 exemplos. Ele está distribuído em 12/52/79/49 classes para os níveis 1, 2, 3 e 4, respectivamente. É importante observar que o número de classes no quarto nível é menor, indicando que muitos ramos da hierarquia possuem nós folha no terceiro nível. Classes com menos de cinco exemplos foram unidas com sua classe pai. Nós que não possuem irmãos também foram unidos com a classe pai. Os atributos com valores constantes foram removidos. Assim, o conjunto pré-processado adquiriu a seguinte configuração: 74 atributos; 7077 exemplos; e 12/52/77/39 classes por nível, do nível 1 ao nível 4.

Enzimas

Enzimas são um conjunto de proteínas que têm a função de acelerar processos químicos dentro da célula. Enzimas são importantes catalisadores e cada enzima é específica para algum tipo de reação. A codificação EC produz códigos de classes funcionais de enzimas e sua hierarquia é estruturada em uma árvore de quatro níveis.

O conjunto de dados EC possui originalmente 709 atributos e 13995 exemplos. Esse conjunto possui 6/45/105/234 classes para os níveis 1, 2, 3 e 4, respectivamente. Devido ao elevado custo computacional decorrente do grande número de subtarefas de classificação binária para as quais um classificador deve ser induzido, foram consideradas neste trabalho

apenas as classes 1 e 3 (e seus descendentes), por possuírem maior representatividade e chegarem a níveis mais baixos na hierarquia. Classes com menos de cinco exemplos foram unidas com sua classe pai. Nós que não possuem irmãos, também foram unidos com a classe pai. Os atributos com valores constantes foram removidos. Assim, o conjunto pré-processado adquiriu a seguinte configuração: 415 atributos; 7872 exemplos; e 2/21/55/98 classes por nível, do nível 1 ao nível 4.

4.2.2.2 Medidas de Avaliação

As medidas de desempenho utilizadas nos experimentos foram Precisão, Revocação e F-measure, tradicionalmente utilizadas em classificação plana, que foram aplicadas em cada nível da hierarquia; e as medidas hierárquicas Precisão Hierárquica, Revocação Hierárquica e F-measure Hierárquico. As medidas Precisão, Revocação, F-measure, Precisão Hierárquica, Revocação Hierárquica e F-measure Hierárquico são representadas, respectivamente, pelas equações 4.4, 4.5, 4.6, 4.7, 4.8 e 4.9. Nelas, TP significa verdadeiro positivo, FP significa falso positivo e FN significa falso negativo; β é um valor real positivo utilizado para dar mais peso a uma das medidas. Quando β é igual a 1, as duas medidas têm o mesmo peso; P_i é um conjunto que contém a classe predita e todos os ancestrais do exemplo i e T_i é um conjunto que contém o rótulo verdadeiro e todos os ancestrais do exemplo i .

$$Pre = \frac{TP}{TP + FP} \quad (4.4)$$

$$Rev = \frac{TP}{TP + FN} \quad (4.5)$$

$$F = (1 + \beta^2) \frac{Pre \cdot Rev}{\beta^2 \cdot Pre + Rev} \quad (4.6)$$

$$hP = \frac{|P_i \cap T_i|}{|P_i|} \quad (4.7)$$

$$hR = \frac{|P_i \cap T_i|}{|T_i|} \quad (4.8)$$

$$hF = (1 + \beta^2) \frac{hP \cdot hR}{\beta^2 \cdot hP + hR} \quad (4.9)$$

4.2.2.3 Configurações

A técnica *ClusterOSS* foi implementada utilizando o algoritmo *k-médias* de agrupamento de dados. Para definir o número de grupos, foi utilizada a média de silhueta do conjunto de treinamento. Para o *k-médias*, foi utilizada a distância Euclidiana como medida de proximidade e 10 como número máximo de iterações. Para obter a média de silhueta, foi considerada a distância Euclidiana como medida de proximidade. Para o CBO, a mesma estratégia foi utilizada. Sobreamostragem aleatória e subamostragem aleatória fazem com que o conjunto de dados fique com proporção final de 1:1 para o número de exemplos em cada classe. A técnica SMOTE foi combinada com subamostragem aleatória, como sugerido por seus autores. O SMOTE altera a distribuição do conjunto de treinamento promovendo um aumento de 200% no número de exemplos no conjunto minoritário e subamostragem do conjunto majoritário, de forma que, ao final do processo, a classe minoritária representa 75% da classe majoritária. Dois algoritmos de classificação diferentes foram aplicados para cada conjunto processado, sendo eles o algoritmo RPART, como árvore de decisão, e SVM. Eles foram escolhidos por serem duas das abordagens mais tradicionais em AM.

A abordagem de exploração da hierarquia foi realizada por meio da divisão por irmãos. Para evitar inconsistência na etapa de classificação, foi utilizada a abordagem *top-down*, classificando inicialmente entre as classes do primeiro nível, e utilizando apenas as subárvores de interesse para classificar exemplos nos demais níveis. Foi utilizado *threshold* = 0.4 para a abordagem *top-down*. Foi realizada validação cruzada estratificada com 5 *folds*, que foi executada 10 vezes.

4.2.2.4 Resultados e Discussões

Os experimentos apresentados a seguir são referentes ao conjunto de dados GPCR e EC utilizando RPART (árvore de decisão) e SVM como algoritmos de classificação. Inicialmente, são apresentados os resultados de GPCR com árvore de decisão; em seguida GPCR com SVM; EC com árvore de decisão; e EC com SVM.

GPCR com árvore de decisão

Os resultados apresentados a seguir são referentes ao conjunto de dados EC com árvore de decisão. A Figura 4.5 apresenta os resultados obtidos para cada nível com as medidas Precisão, Revocação e *F-measure*. O desempenho preditivo diminui para todas as técnicas, conforme pode ser observado nos níveis mais profundos da hierarquia. Pode ser visto ainda que, para o primeiro nível, não há diferença entre o desempenho do conjunto de dados sem pré-processamento ou gerado pela aplicação de qualquer técnica para dados desbalanceados. Nos demais níveis, pode ser observado que a utilização dessas técnicas

pode melhorar os resultados para a medida Revocação, bem como piorar os resultados para a medida Precisão.

As medidas Precisão Hierárquica, Revocação Hierárquica e *F-measure* Hierárquico foram calculadas separadamente para cada classe. A Figura 4.6 apresenta o desempenho calculado por meio da média das classes. É possível observar que a aplicação de técnicas de pré-processamento em todas as classes, sem utilização de algum critério para decidir quando ou não aplicar, levaram a indução de modelos com pior desempenho preditivo, quando comparado com o uso dos dados originais. Além disso, a abordagem sem pré-processamento tem seu desempenho prejudicado principalmente na medida revocação.

Observando detalhadamente o desempenho preditivo nas classes de forma separada, nota-se que a utilização do conjunto de dados sem pré-processamento mostrou-se vantajosa em apenas 65 das 166 sub-tarefas de classificação para a medida *F-measure* Hierárquica. Os modelos das outras 101 classes mostraram melhor desempenho quando pré-processadas por alguma técnica. A Figura 4.7 apresenta uma meta-análise dos resultados através de árvores de decisão. Elas representam cenários mais favoráveis a aplicação de técnicas de pré-processamento. As árvores foram geradas a partir dos resultados obtidos por classe. Foi gerada uma árvore para cada medida de avaliação. As características utilizadas foram informações sobre o conjunto de dados associado a cada classe da hierarquia, que estão representados na Tabela 4.4. Nós folhas com número 1 representam cenários em que aplicar alguma técnica de pré-processamento produz um melhor desempenho, enquanto o número 0 representa que a abordagem sem pré-processamento obtém melhor desempenho.

Tabela 4.4: Tabela descritiva sobre as características das classes

Característica	Descrição
nmin	Número de exemplos da classe minoritária
nmaj	Número de exemplos da classe majoritária
interesse	Se o número de exemplos positivos é maior do que o número de exemplos negativos, interesse = maj; senão interesse = min.
nnivel	Nível da hierarquia no qual a classe se encontra
nirmaos	Número de irmãos que a classe possui
nexemplos	Número de exemplos no conjunto de treinamento
X1.	Proporção entre exemplos majoritários e minoritários
min.	Porcentagem de exemplos minoritários
maj.	Porcentagem de exemplos majoritários

O critério que foi associado a raiz de duas das árvores é nmin, mostrando que a representatividade da classe minoritária é importante para a decisão da aplicação de pré-processamento. Considerando a árvore da Figura 4.7c, se nmin for maior do que 128, significa que a classe minoritária está representada em quantidade suficiente de exemplos, e assim não há necessidade de pré-processamento para a maior parte das sub-tarefas de

(d)

(e)

(f)

Figura 4.5: a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia de GPCR utilizando árvores de decisão

Figura 4.6: a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para GPCR com árvores de decisão

Figura 4.7: Árvore de decisão para aplicação de pré-processamento no conjunto GPCR com árvore de decisão considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico

classificação binária. Caso n_{min} seja menor do que 128, o critério seguinte é o número de exemplos do conjunto de dados associados à classe hierárquica. Caso esses critérios fossem conhecidos antes do processo de indução dos modelos, eles poderiam ser utilizados para decidir em quais casos seria aplicada alguma técnica de pré-processamento para dados desbalanceados. Essa abordagem que aplica o pré-processamento em alguns casos e em outros não, será chamada neste texto de híbrida. A Figura 4.8 apresenta o desempenho preditivo caso os critérios tivessem sido considerados para a abordagem híbrida. Vale ressaltar a necessidade de induzir tais árvores de decisão antes da geração dos modelos, que fica como trabalho futuro desta dissertação.

Figura 4.8: Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para GPCR com árvore de decisão

É possível observar ainda que não há melhora significativa de precisão quando é utilizada abordagem híbrida. Já para a medida revocação, é possível notar uma clara melhora obtida com a adoção da abordagem híbrida. Considere a medida Precisão Hierárquica, $hP = \frac{|P_i \cap T_i|}{|P_i|}$, ela é menor quando a predição é mais específica do que o rótulo verdadeiro. O oposto acontece com a Revocação, $hR = \frac{|P_i \cap T_i|}{|T_i|}$, que é menor quando o rótulo verdadeiro é mais específico do que a predição. Assim, pode-se concluir que a utilização de técnicas de pré-processamento faz com que a classificação se torne mais específica, ou seja, desça mais na hierarquia, comparada a classificação sem pré-processamento. Na medida F-measure Hierárquico, que é um equilíbrio entre a Precisão e Revocação, o desempenho preditivo da abordagem híbrida é superior.

GPCR com SVM

Os resultados apresentados a seguir são referentes ao conjunto de dados GPCR com SVM. A Figura 4.9 apresenta os resultados obtidos para cada nível com as medidas Precisão, Revocação e *F-measure*. Neste cenário, o desempenho preditivo também diminui para todas as técnicas, conforme pode ser observado nos níveis mais profundos da hierarquia. O desempenho preditivo piorou com a utilização do SVM com os parâmetros adotados em comparação com a utilização de árvore de decisão, apresentada anteriormente. Observa-se que em muitos casos a predição no nível 4 chegou a ser nula e os níveis 2 e 3 obtiveram desempenhos preditivos muito baixos.

A Figura 4.10 apresenta o desempenho preditivo com medidas hierárquicas calculadas por meio da média das classes. Novamente, a aplicação sem critério das técnicas de pré-processamento em todas as classes piorou o desempenho preditivo em todos os cenários, com exceção das técnicas ADASYN, OSS e sobreamostragem aleatória na medida revocação, que obtiveram desempenho preditivo médio levemente superior ao obtido pela abordagem sem pré-processamento.

A Figura 4.11 apresenta as árvores de decisão que representam cenários mais favoráveis a aplicação de técnicas de pré-processamento. Da mesma forma que as árvores apresentadas em GPCR com árvore de decisão, conjuntos de dados com pouca representação minoritária e alta representação majoritária possuem desempenho preditivo superior quando aplicada alguma técnica de pré-processamento. É interessante observar que, em todos os casos, um número grande de irmãos indica que a abordagem sem pré-processamento é a mais indicada. Um número grande de irmãos pode indicar que o conjunto majoritário é composto por diversos subconceitos, e que cada um desses subconceitos pode estar balanceado em relação à classe de interesse.

A Figura 4.12 apresenta o desempenho preditivo caso os critérios tivessem sido considerados para a abordagem híbrida. É possível observar que os resultados da abordagem híbrida para o conjunto GPCR com classificadores SVM foram superiores quando comparados à abordagem sem pré-processamento. A medida de desempenho com a maior diferença entre as abordagens foi novamente a revocação.

EC com árvore de decisão

A Figura 4.13 apresenta os resultados obtidos para o conjunto de dados EC utilizando o algoritmo de indução de árvores de decisão para cada nível com as medidas Precisão, Revocação e *F-measure*. Neste cenário, o desempenho preditivo das classes não mostrou o comportamento semelhante aos observado nos experimentos anteriores. É possível observar que muitas técnicas obtiveram desempenhos melhores em níveis mais profundos da hierarquia. Além disso, algumas técnicas apresentaram desempenho superior à

(a)

(b)

(c)

Figura 4.9: a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia para GPCR com SVM

(a)

(b)

(c)

Figura 4.10: a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para GPCR com SVM

(a)

(b)

(c)

Figura 4.11: Árvore de decisão para aplicação de pré-processamento no conjunto GPCR com SVM considerando as medidas a) Precisão Hierárquica, b) Revocação Hierárquica e c) F-measure Hierárquico

Figura 4.12: Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para GPCR com SVM

abordagem sem pré-processamento, como no nível 2 com ADASYN, ClusterOSS e sobreamostragem aleatória em todas as medidas.

A Figura 4.14 apresenta o desempenho calculado por meio da média do desempenho preditivo das classes. Como em outros cenários, a aplicação sem um critério das técnicas em todas as classes piorou o desempenho preditivo. A abordagem sem pré-processamento novamente mostrou desempenho inferior na medida revocação hierárquica quando comparada com seu desempenho na medida precisão hierárquica.

A Figura 4.15 apresenta as árvores de decisão que representam cenários mais favoráveis a aplicação de técnicas de pré-processamento. As árvores apontam que conjuntos de dados com número grande de exemplos obtêm desempenho superior com a abordagem sem pré-processamento. Uma possível explicação é que conjuntos com muitos exemplos possuem maior representatividade e tem menos possibilidade de sofrer com o problema de classes raras.

A Figura 4.16 apresenta o desempenho preditivo caso os critérios sejam considerados para a abordagem híbrida. Observa-se uma melhora significativa no desempenho com a abordagem híbrida.

EC com SVM

A Figura 4.17 apresenta os resultados obtidos para cada nível referentes às medidas Precisão, Revocação e *F-measure* para o conjunto de dados EC utilizando SVM

(a)

(b)

(c)

Figura 4.13: a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia para EC com árvore de decisão

(a)

(b)

(c)

Figura 4.14: a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para EC com árvore de decisão

Figura 4.15: Árvore de decisão para aplicação de pré-processamento no conjunto EC com árvore de decisão considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico

Figura 4.16: Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para EC com árvore de decisão

como algoritmo de classificação. Neste cenário, o desempenho preditivo também diminui para todas as técnicas, conforme pode ser observado nos níveis mais profundos da hierarquia. Para o primeiro nível, não há diferença entre o desempenho do conjunto sem pré-processamento ou gerado pela aplicação de qualquer técnica para dados desbalanceados. Nos demais níveis, pode ser observado que a utilização dessas técnicas pode prejudicar a predição. Diferente de GPCR, EC apresentou melhor desempenho com modelos induzidos pelo algoritmo SVM.

A Figura 4.18 apresenta o desempenho preditivo calculado por meio da média do desempenho das classes. A aplicação sem um critério das técnicas de pré-processamento em todas as classes piorou o desempenho preditivo em todos os cenários, com exceção da sobreamostragem aleatória na medida revocação.

A Figura 4.19 apresenta as árvores de decisão que representam cenários mais favoráveis a aplicação de técnicas de pré-processamento com o conjunto de dados EC utilizando classificadores induzidos por SVM. Novamente, duas das árvores possuem a característica *nmin* como raiz, enfatizando a importância do número de exemplos minoritários nos cenários favoráveis à aplicação de técnicas. Outras duas características recorrentes foram a proporção entre exemplos majoritários e minoritários e o número de exemplos no conjunto de treinamento. Nesse cenário, quanto maior *X1*., mais propício é para aplicação de técnicas de pré-processamento. Já a característica *nexemplos* é fortemente dependente de

(a)

(b)

(c)

Figura 4.17: a)Precisão, b)Revocação e c)F1 para os 4 níveis da hierarquia para EC com SVM

(a)

(b)

(c)

Figura 4.18: a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico para EC com SVM

nmin. Por exemplo, se o valor de nmin é baixo e o de nexemplos é alto, o cenário é mais favorável a aplicação de técnicas.

A Figura 4.20 apresenta o desempenho preditivo caso os critérios tivessem sido considerados para a abordagem híbrida. É possível notar que os resultados da abordagem híbrida foram levemente superiores quando comparados a abordagem sem pré-processamento. Novamente, a medida com a maior diferença foi revocação. A diferença é menor do que as apresentadas em GPCR com árvore de decisão e SVM e EC com árvore de decisão porque a abordagem sem pré-processamento já apresenta bons resultados neste cenário.

Todos os resultados das abordagens híbrida e sem pré-processamento, para todas as bases e todas as medidas, foram submetidos ao teste de Wilcoxon pareado. Em todos os cenários há evidências de que há diferença entre os desempenhos.

Os experimentos realizados mostram que aplicar técnicas sem um critério atrapalha a indução de modelos adequados para problemas de classificação hierárquica utilizando abordagem local por nó com divisão por irmãos. Experimentos preliminares apontam comportamento semelhante utilizando a abordagem menos inclusiva. Em todos os cenários, praticamente todas as técnicas apresentaram desempenho inferior em todas as medidas de avaliação quando aplicadas em todos os conjuntos no processo de classificação. Contudo, é possível melhorar o desempenho de classificação utilizando técnicas de pré-processamento para alguns subconjuntos escolhidos por meio de alguns critérios.

Esses critérios são características dos conjuntos de treinamento. As características consideradas neste trabalho foram: número de exemplos da classe minoritária; número de exemplos da classe majoritária; se a classe hierárquica é minoritária ou majoritária no conjunto; nível da hierarquia no qual a classe se encontra; número de irmãos que a classe possui; número de exemplos do conjunto; proporção entre exemplos majoritários e minoritários; porcentagem de exemplos minoritários e porcentagem de exemplo majoritários.

Para identificar critérios e definir os cenários interessantes para aplicação de técnicas de pré-processamento, foram geradas árvores de decisão que consideram as características apresentadas e os desempenhos preditivos obtidos. As árvores apontam importantes aspectos que devem ser considerados. Alguns dos critérios apontados pelas análises experimentais se encontram na Tabela 4.5. Classes com quantidade considerável de exemplos minoritários, quando aplicado técnicas, podem ser descaracterizadas e prejudicadas no processo de indução do modelo. Em contra ponto, se a base possui um número elevado de exemplos majoritários, a utilização de técnicas mostra melhor desempenho. Quando a classe de interesse é majoritária, a abordagem sem pré-processamento é a mais indicada por não prejudicar a classificação dos exemplos de interesse. Em classes com muitos

Figura 4.19: Árvore de decisão para aplicação de pré-processamento no conjunto EC com SVM considerando as medidas a)Precisão Hierárquica, b)Revocação Hierárquica e c)F-measure Hierárquico

Figura 4.20: Gráfico comparativo da abordagem sem Pré-processamento e Híbrida para EC com SVM

irmãos, a abordagem sem pré-processamento foi superior possivelmente por existir grande possibilidade da classe majoritária ser composta por diversos subconceitos (referentes a cada um dos irmãos) e eles estarem mais balanceados com a classe minoritária. O número de exemplos, nos cenários abordados, geralmente está relacionado à representatividade. Quanto maior o número de exemplos, mais representativo é o conjunto e existe menos necessidade de utilizar técnicas. Além disso, conjuntos grandes têm menos possibilidade de possuírem classes raras.

Tabela 4.5: Critérios para a aplicação de técnicas de pré-processamento para dados desbalanceados em classificação hierárquica

Característica	Melhor Desempenho	
	Sem pré-processamento	Com pré-processamento
Muitos exemplos minoritários	✓	
Poucos exemplos minoritários		✓
Muitos exemplos majoritários		✓
Poucos exemplos majoritários	✓	
Classe de interesse é majoritária	✓	
Classe de interesse é minoritária		✓
Muitos irmãos	✓	
conjuntos grandes	✓	

Os resultados com a abordagem híbrida mostraram que a utilização de técnicas de dados desbalanceados pode melhorar o desempenho de algoritmos de classificação em problemas hierárquicos, desde que sejam consideradas as características adequadas dos subconjuntos gerados. A abordagem sem pré-processamento apresenta menor desempenho na medida revocação, por não se aprofundar na hierarquia durante a classificação. As técnicas de pré-processamento, quando utilizadas criteriosamente, podem fazer com que o modelo hierárquico seja capaz fazer classificações mais específicas, melhorando assim o desempenho nas medidas revocação e F1.

4.3 Considerações Finais

Neste capítulo foram apresentadas as avaliações experimentais para conjuntos de dados de classificação binárias e hierárquica. No contexto de classificação binária, a técnica *ClusterOSS* foi apresentada e avaliada comparativamente com técnicas da literatura. Quando combinada com sobreamostragem aleatória, seu desempenho mostrou-se superior a grande parte das técnicas e comparável à técnica SMOTE.

No contexto de classificação hierárquica, buscou-se melhorar o desempenho preditivo por meio da aplicação de técnicas para dados desbalanceados. Observou-se que aplicar técnicas sem um critério pode prejudicar a indução adequada de modelos e assim prejudicar o desempenho preditivo. Porém, quando alguns critérios se levados em conta, como representatividade da classe minoritária e número de exemplos do conjunto de dados da classe hierárquica, para decidir sobre a aplicação de técnicas, permitem conseguir uma melhora significativa no desempenho preditivo em classificação hierárquica com dados desbalanceados.

Conclusão

Um conjunto de dados é dito desbalanceado se a distribuição de suas classes for desigual. Tal problema interfere na indução de um modelo de classificação com boa acurácia preditiva pela maioria dos algoritmos de classificação de dados. Diversos cenários de classificação podem ser afetados pelo desbalanceamento, como: classificação binária, classificação multiclasse, classificação hierárquica e classificação multirótulo. As técnicas que tentam solucionar o problema de desbalanceamento de dados podem ser divididas em duas principais abordagens: pré-processamento e algorítmica. Nesse trabalho, foram utilizadas técnicas de pré-processamento, são elas: sobreamostragem aleatória, subamostragem aleatória, OSS, SMOTE, CBO, ADASYN.

Este trabalho focou nos cenários de classificação binária e hierárquica, que é um tipo de classificação que considera relações entre as classes, por meio de subclasses e superclasses. As abordagens para classificação hierárquica podem ser divididas em classificação plana, classificação por nível, classificação local e classificação global. A abordagem utilizada nesse trabalho foi classificação local, que transforma o problema hierárquico original em um conjunto de subproblemas binários. Os algoritmos de classificação binários utilizados foram RPART, para indução de árvore de decisão, e SVM.

Em classificação hierárquica, técnicas binárias para dados desbalanceados foram utilizadas. Para isso, o problema foi dividido em subproblemas binários por meio de uma abordagem local por nó com divisão por irmãos, sendo os exemplos classificados de forma

top-down. Os experimentos indicaram que a aplicação de técnicas de forma descriteriosa, ou seja a aplicação em todos os subproblemas binários, piora o desempenho preditivo dos modelos. Além disso, foram encontrados indícios que apontam cenários mais propícios para a aplicação de técnicas, como conjuntos com poucos exemplos na classe minoritária, com pouca representatividade (poucos exemplos minoritários e majoritários), cenários em que a classe de interesse é minoritária e classes em níveis mais fundos na hierarquia. As análises apresentaram ainda que, quando as características supracitadas são consideradas, o desempenho preditivo do classificador local pode ser melhorado por meio da aplicação de técnicas de pré-processamento para lidar com dados desbalanceados.

5.1 Principais Contribuições

Resumidamente, as principais contribuições do presente trabalho para a área de classificação binária e classificação hierárquica são descritas a seguir:

- Revisão bibliográfica sobre técnicas de dados desbalanceados em classificação hierárquica. Além disso, este trabalho avaliou o desempenho de diversas técnicas para dados desbalanceados aplicadas em cenários hierárquicos.
- Proposta de uma nova técnica, chamada de ClusterOSS, para classificação binária. A técnica ClutserOSS é baseada na técnica OSS, e as duas maiores diferenças entre as duas é que OSS seleciona apenas um exemplo da classe majoritária para a subamostragem e faz a seleção de forma aleatória, enquanto ClusterOSS pode selecionar mais de um exemplo e sempre seleciona representantes centrais de regiões de concentração majoritária. A técnica proposta apresentou desempenho competitivo quando comparada com a técnica SMOTE e quando foi utilizada uma combinação com subamostragem aleatória.
- Aplicação de técnicas binárias para dados desbalanceados em cenários hierárquicos. Não foram encontradas comparações de abordagens semelhantes à realizada neste trabalho na literatura.
- Utilização de medidas de avaliação para cada classe, calculadas durante a análise dos resultados. Essa abordagem permite que um cenário de dados desbalanceados seja considerado na avaliação dos modelos induzidos.

5.2 Trabalhos Futuros

As contribuições apresentadas neste trabalho geraram a necessidade de novos estudos, tanto para estender as técnicas estudadas quanto para abordar outros aspectos que surgiram no decorrer da pesquisa. A seguir são apresentadas algumas sugestões de tópicos que podem ser abordados em estudos futuros:

- Utilização de outras características, como medidas de complexidade e separabilidade, da base para avaliação de classes mais propícias a utilização de técnicas de pré-processamento.
- Implementação de um meta-aprendiz capaz de decidir, para cada classe da hierarquia, cenários favoráveis à aplicação de técnicas para dados desbalanceados dependendo de suas características.
- Utilização de bases hierárquicas multirótulo.
- Investigação de outras abordagens de exploração da hierarquia e divisão do conjunto de treinamento. Neste trabalho, a abordagem utilizada foi exploração local por nó com divisão por irmãos. Essa divisão foi utilizada por haver indícios de que torna os conjuntos mais balanceados. Como trabalho futuro, pretende-se investigar quais divisões possuem o menor efeito de desbalanceamento. Além disso, outras explorações podem ser investigadas, como exploração por nível e global.

5.3 Publicação

O desenvolvimento deste projeto de mestrado resultou em um artigo científico intitulado “*ClusterOSS: a new undersampling method for imbalanced learning*”, publicado no ENIAC (Encontro Nacional de Inteligência Artificial e Computacional) em 2014.

Referências

- Alcalá-Fdez, J.; Fernandez, A.; Luengo, J.; Derrac, J.; García, S.; Sánchez, L.; Herrera, F. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. 2011.
Disponvel em <http://sci2s.ugr.es/keel/datasets.php>
- Alejo, R.; García, V. S. J. M. R. A.; Sánchez, J. Improving the performance of the rbf neural networks trained with imbalanced samples, *proc. of intell. data eng. autom. learn.*, vol. 7 of *lecture notes in computer science*, springer, pp. 720–747. 2006.
- Ashburner, M.; Ball, C. A.; Blake, J. A.; Botstein, D.; Butler, H.; Cherry, J. M.; Davis, A. P.; Dolinski, K.; Dwight, S. S.; Eppig, J. T.; Harris, M. A.; Hill, D. P.; Issel-Tarver, L.; Kasarskis, A.; Lewis, S.; Matese, J. C.; Richardson, J. E.; Ringwald, M.; Rubin, G. M.; Sherlock, G. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. *Nature genetics*, v. 25, n. 1, p. 25–29, 2000.
Disponvel em <http://dx.doi.org/10.1038/75556>
- Bache, K.; Lichman, M. UCI machine learning repository. 2014.
Disponvel em <http://archive.ics.uci.edu/ml>
- Barbedo, J. G. A.; Lopes, A. Automatic genre classification of musical signals. *EURASIP J. Appl. Signal Process.*, v. 2007, n. 1, p. 157–157, 2007.
Disponvel em <http://dx.doi.org/10.1155/2007/64960>
- Braga, A.; Horta, E.; Natowicz, R.; Rouzier, R.; Incitti, R.; Rodrigues, T.; Costa, M.; Pataro, C.; Çela, A. Bayesian classifiers for predicting the outcome of breast cancer preoperative chemotherapy. In: Prevost, L.; Marinai, S.; Schwenker, F., eds. *Artificial Neural Networks in Pattern Recognition*, v. 5064 de *Lecture Notes in Computer Science*,

- Springer Berlin Heidelberg, p. 263–266, 2008.
Disponvel em http://dx.doi.org/10.1007/978-3-540-69939-2_25
- Burred, J. J.; Lerch, A. A hierarchical approach to automatic musical genre classification. In: *in Proc. Of the 6 th Int. Conf. on Digital Audio Effects (DAFx, 2003*, p. 8–11.
- Canada, C.; Kiritchenko, S.; Famili, A. F. Learning and evaluation in the presence of class hierarchies: Application to text categorization. 2006.
- Castro, C.; Braga, A. Artificial neural networks learning in roc space, proc. of the 1st international conference on neural computation (icnc'09), insticc, pp. 219– 224. 2009.
- Castro, C.; Braga, A. Aprendizado supervisionado com conjuntos de dados desbalanceados. 2011.
- Ceci, M.; Malerba, D. Classifying web documents in a hierarchy of categories: a comprehensive study. *Journal of Intelligent Information Systems*, v. 28, n. 1, p. 37–78, 2007.
Disponvel em <http://dx.doi.org/10.1007/s10844-006-0003-2>
- Cesa-Bianchi, N.; Gentile, C.; Zaniboni, L. Hierarchical classification: Combining bayes with svm. In: *Proceedings of the 23rd International Conference on Machine Learning, ICML '06*, New York, NY, USA: ACM, 2006, p. 177–184 (*ICML '06*,).
Disponvel em <http://doi.acm.org/10.1145/1143844.1143867>
- Chawla, N. V.; Bowyer, K. W.; Hall, L. O.; Kegelmeyer, W. P. Smote: Synthetic minority over-sampling technique. *J. Artif. Int. Res.*, v. 16, n. 1, p. 321–357, 2002.
Disponvel em <http://dl.acm.org/citation.cfm?id=1622407.1622416>
- Chen, B.; Hu, J. Hierarchical multi-label classification incorporating prior information for gene function prediction. In: *Intelligent Systems Design and Applications (ISDA), 2010 10th International Conference on*, 2010, p. 231–236.
- Chen, Y.; Li, Z.; Hu, X.; Liu, J. Hierarchical classification with dynamic-threshold svm ensemble for gene function prediction. In: Cao, L.; Zhong, J.; Feng, Y., eds. *Advanced Data Mining and Applications*, v. 6441 de *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, p. 336–347, 2010.
Disponvel em http://dx.doi.org/10.1007/978-3-642-17313-4_33
- Clare, A. *Machine learning and data mining for yeast functional genomics*. Tese de Doutoramento, The University of Wales, 2003.

- Clare, A.; King, R. D. Predicting gene function in *saccharomyces cerevisiae*. *Bioinformatics*, v. 19, p. 42–49, 2003.
- Costa, E. P.; Lorena, A. C.; Carvalho, A. C.; Freitas, A. A. Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In: *Proceedings of the 3rd Brazilian Symposium on Bioinformatics: Advances in Bioinformatics and Computational Biology*, BSB '08, Berlin, Heidelberg: Springer-Verlag, 2008, p. 35–46 (BSB '08,).
Disponvel em http://dx.doi.org/10.1007/978-3-540-85557-6_4
- 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*, SIGIR '00, New York, NY, USA: ACM, 2000, p. 256–263 (SIGIR '00,).
Disponvel em <http://doi.acm.org/10.1145/345508.345593>
- Eisner, R.; Poulin, B. S. D. L. P.; Greiner, R. Improving protein function prediction using the hierarchical structure of the gene ontology. in: *Proceedings of the ieee symposium on computational intelligence in bioinformatics and computational biology*, pp 1–10. 2005.
- Fagni, T.; Sebastiani, F. On the selection of negative examples for hierarchical text categorization. in: *Proceedings of the 3rd language technology conference*, pp 24–28. 2007.
- Fawcett, T.; Provost, F. Adaptive fraud detection, *data min. knowl. discov.* 1(3): 291–316. 1997.
- Freitas, A.; de Carvalho, A. Research and trends in data mining technologies and applications, idea group, chap a: tutorial on hierarchical classication with applications in bioinformatics, pp 175–208. 2007.
- Freund, Y.; Schapire, R. E. A decision-theoretic generalization of on-line learning and an aplication to boosting, *j. comput. syst. sci.* 55(1) 119–139. 1997.
- Ganesan, P.; Garcia-Molina, H.; Widom, J. Exploiting hierarchical domain structure to compute similarity. *ACM Trans. Inf. Syst.*, v. 21, n. 1, p. 64–93, 2003.
Disponvel em <http://doi.acm.org/10.1145/635484.635487>
- Han, H., W. W.-Y.; Mao, B.-H. Borderlinesmote: A new over-sampling method in imbalanced data sets learning, *advances in intelligent computing*, vol. 3644 of lecture notes in computer science, springer berlin, heidelberg, pp. 878–887. 2005.

- He, H.; Bai, Y.; Garcia, E.; Li, S. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, 2008, p. 1322–1328.
- He, H.; Garcia, E. Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, v. 21, n. 9, p. 1263–1284, 2009.
- Hoffmann-Ostenhof, O. Enzyme nomenclature, recommendations, 1972. *European Journal of Biochemistry*, v. 45, n. 1, p. 1–3, 1974.
Disponvel em <http://dx.doi.org/10.1111/j.1432-1033.1974.tb03521.x>
- Japkowicz, N.; Stephen, S. The class imbalance problem: A systematic study, *intell. data anal.* 6(5): 429–449. 2002.
- Jo, T.; Japkowicz, N. Class imbalances versus small disjuncts. *SIGKDD Explor. Newsl.*, v. 6, n. 1, p. 40–49, 2004.
Disponvel em <http://doi.acm.org/10.1145/1007730.1007737>
- Joachims, T. Learning to classify text using support vector machines: Methods, theory and algorithms, kluwer academic publishers, norwell, ma, usa. 2002.
- Karakoulas, G.; Shawe-Taylor, J. Optimizing classifiers for imbalanced training sets, proceedings of conference on advances in neural information processing systems ii, mit press, cambridge, ma, usa, pp. 253–259. 1999.
- Kiritchenko, S.; Matwin, S.; Nock, R.; Famili, A. F. Learning and evaluation in the presence of class hierarchies: Application to text categorization. In: *Proceedings of the 19th International Conference on Advances in Artificial Intelligence: Canadian Society for Computational Studies of Intelligence*, AI'06, Berlin, Heidelberg: Springer-Verlag, 2006, p. 395–406 (AI'06,).
Disponvel em http://dx.doi.org/10.1007/11766247_34
- Kubat, M.; Matwin, S. Addressing the curse of imbalanced training sets: one-sided selection, *proc. 14th international conference on machine learning*, morgan kaufmann, pp. 179–186. 1997.
- Kukar, M.; Kononenko, I. Cost-sensitive learning with neural networks, proceedings of the 13th european conference on artificial intelligence (ecai-98), john wiley and sons, pp. 445–449. 1998.

- Labrou, Y.; Finin, T. Yahoo! as an ontology—using yahoo! categories to describe documents. in: Proceedings of the acm conference on information and knowledge management, pp 180–187. 1999.
- Li, C. Classifying imbalanced data using a bagging ensemble variation (bev). in: the acm southeast conference, pp. 203–208. 2007.
- Ling, C. X.; Li, C. Data mining for direct marketing: problems and solutions, proc. fourth acm sigkdd int. conf. knowledge discovery and data min., new york, ny, pp. 73–79. 1998.
- Liu, X.Y.; Wu, J.; Zhou, Z. Exploratory under sampling for class imbalance learning,” proc. int’l conf. data mining, pp. 965–969. 2006.
- Morik, K.; Brockhausen, P.; Joachims, T. Combining statistical learning with a knowledge-based approach - a case study in intensive care monitoring, proceedings of the sixteenth international conference on machine learning, morgan kaufmann publishers inc., san francisco, ca, usa, pp. 268–277. 1999.
- Moturu, S. T; Johnson, W. G.; Liu, H. Predictive risk modelling for forecasting high-cost patients: a real-world application using medicaid data, international journal of biomedical engineering and technology 2(1): 114–132. 2010.
- Natowicz, R; Incitti, R. H. E. G. C. B. G. P. Y. K. C. C. A. F. P. L.; Rouzier, R. Prediction of the outcome of preoperative chemotherapy in breast cancer by dna probes that convey information on both complete and non complete responses, bmc bioinformatics 9: 149–166. 2008.
- Pazzani, M.; Merz, C. M. P. A. K. H. T.; Brunk, C. Reducing misclassification costs, proceedings of the 11th international conference on machine learning, icml, morgan kaufmann, pp. 217–225. 1994.
- Prati, R. C; Batista, G. E. A. P. A.; Monard, M. C. Class imbalances versus class overlapping: An analysis of a learning system behavior, micai 2004: Advances in artificial intelligence, third mexican international conference on artificial intelligence, vol. 2972 of lecture notes in computer science, springer, pp. 312–321. 2004b.
- Rocchio, J. The smart retrieval system: experiments in automatic document processing, chap: relevance feedback in information retrieval, prentice hall, pp 313–323. 1971.
- Silla, Jr., C. N.; Freitas, A. A. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, v. 22, n. 1-2, p. 31–72, 2011.
Disponvel em <http://dx.doi.org/10.1007/s10618-010-0175-9>

- Silva, C.; Silva, A. N. S. P. A. J. G.; Nunes, R. Lung nodules classification in ct images using simpsons index, geometrical measures and oneclass svm, machine learning and data mining in patternrecognition, vol. 5632 of lecture notes in computer science, springer berlin / heidelberg, pp. 810–822. 2009.
- Soumen Chakrabarti, B. E. D.; Indyk., P. "enhanced hypertext categorization using hyperlinks" in proceedings of acm sigmod '98, [online: <http://www.cs.berkeley.edu/~soumen/sigmod98.ps>]. 1998.
- Souza, M. R. P.; Cavalcanti, G. C.; Tsang, I. R. Off-line signature verification: An approach based on combining distances and one-class classifiers, proceedings of the 22nd ieee international conference on tools with artificial intelligence, ictai 2010, arras, france, ieee computer society, pp. 7–11. 2010.
- Stolfo, S. J.; Prodromidis, A. L. T. S. L. W. F. D. W.; Chan, P. K. Jam: Java agents for meta-learning over distributed databases. in proceedings of the 3rd international conference on knowledge discovery and data mining, 74–81. new- port beach, ca: Aaai press. 1997.
- Sun, A.; Lim, E.-P.; Ng, W.-K.; Srivastava, J. Blocking reduction strategies in hierarchical text classification. *IEEE Trans. on Knowl. and Data Eng.*, v. 16, n. 10, p. 1305–1308, 2004.
Disponvel em <http://dx.doi.org/10.1109/TKDE.2004.50>
- Sun, Y; Wong, A. K. C.; Kamel, M. S. Classification of imbalanced data: A review, international journal of pattern recognition and artificial intelligence 23(4): 687-719. 2009.
- Sun, A.; Lim, E. Hierarchical text classification and evaluation. in: Proceedings of the ieee international conference on data mining, pp 521–528. 2001.
- Sun, Y; Kamel, M. S. W. A. K. C.; Wang, Y. Cost-sensitive boosting for classification of imbalanced data, pattern recognition 40(12): 3358–3378. 2007.
- Tomek, I. Two modifications of cnn. ieee transactions on systems, man and communications, smc – 6, 769-772. 1976.
- Valentini, G. True path rule hierarchical ensembles. In: *Proceedings of the 8th International Workshop on Multiple Classifier Systems*, MCS '09, Berlin, Heidelberg: Springer-Verlag, 2009, p. 232–241 (*MCS '09*,).
Disponvel em http://dx.doi.org/10.1007/978-3-642-02326-2_24

- Vens, C.; Struyf, J.; Schietgat, L.; Džeroski, S.; Blockeel, H. Decision trees for hierarchical multi-label classification. *Mach. Learn.*, v. 73, n. 2, p. 185–214, 2008.
Disponvel em <http://dx.doi.org/10.1007/s10994-008-5077-3>
- Veropoulos, K.; Campbell, C.; Cristianini, N. Controlling the sensitivity of support vector machines, proceedings of the international joint conference on artificial intelligence, pp. 55–60. 1999.
- Wang, J.; Lee, M. C. Reconstructing ddc for interactive classification. In: *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management*, CIKM '07, New York, NY, USA: ACM, 2007, p. 137–146 (*CIKM '07*,).
Disponvel em <http://doi.acm.org/10.1145/1321440.1321462>
- Wang, Y.; Gong, Z. Hierarchical classification of web pages using support vector machine. In: *Proceedings of the 11th International Conference on Asian Digital Libraries: Universal and Ubiquitous Access to Information*, ICADL 08, Berlin, Heidelberg: Springer-Verlag, 2008, p. 12–21 (*ICADL 08*,).
Disponvel em http://dx.doi.org/10.1007/978-3-540-89533-6_2
- Weiss, G. M. Mining with rarity: a unifying framework, *sigkdd explor. newsl.* 6(1): 7–19. 2004.
- Wu, G.; Chang, E. Y. Kba: Kernel boundary alignment considering imbalanced data distribution, *ieee transactions on knowledge and data engineering* 17(6): 786–795. 2005.