DISSERTAÇÃO DE MESTRADO Nº 1025

ESTUDO DE FUNÇÕES DE CUSTO PARA REDES NEURAIS COM DADOS DESBALANCEADOS

Yuri Sousa Aurelio

DATA DA DEFESA: 18/12/2017

Universidade Federal de Minas Gerais

Escola de Engenharia

Programa de Pós-Graduação em Engenharia Elétrica

ESTUDO DE FUNÇÕES DE CUSTO PARA REDES NEURAIS COM DADOS DESBALANCEADOS

Yuri Sousa Aurelio

Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do Título de Mestre em Engenharia Elétrica.

Orientador: Prof. Antônio de Pádua Braga

Belo Horizonte - MG

Dezembro de 2017

Resumo

Este trabalho apresenta uma análise comparativa das técnicas utilizadas em redes neurais para problemas de classes desbalanceadas. A partir de uma comparação inicial das técnicas clássicas, um estudo mais aprofundado é feito com a utilização de funções de custo para lidar com o problema de treinamento das redes neurais em que se tem desbalanceamento das classes de saída. É apresentada uma abordagem da inclusão da informação *a priori* na função de custo de entropia cruzada (do inglês, *cross-entropy*), junto a uma modificação do algoritmo do *resilient backpropagation*, bem como seu impacto no aprendizado de algoritmo em problemas de classes desbalanceadas. Devido à diferença no número de observações entre as classes nos problemas de classes desbalanceadas, medir o desempenho do algoritmo de aprendizado requer métricas apropriadas, como *Area Under the ROC Curve* (AUC), F1-score, Kubat's *Geometric-mean* (G-mean), *Adjusted Geometric-mean* (AGm) e outras. Todavia, a grande maioria dos problemas dessa área é treinada usando o erro médio quadrático ou a entropia cruzada (também conhecida como função de erro logística). Isso faz com que o algoritmo de otimização da rede neural busque otimizar uma função de custo diferente daquela que será utilizada para validação do seu desempenho. É então proposta uma abordagem de como extrair métricas adequadas para problemas de desbalanceamento da matriz de confusão e transformá-las em funções de custo a serem utilizadas durante a etapa de treinamento. Um estudo comparativo entre a abordagem tradicional de treinamento e as funções de custo propostas é realizado, realçando-se os pontos positivos e negativos de cada abordagem. Experimentos numéricos para diferentes bases de treinamento com diferentes tipos de desbalanceamento são apresentados.

Abstract

The work presented here makes a comparative approach of the techniques used in neural networks in problems of unbalanced classes. Based on an initial comparison of classical techniques, a more in-depth study is done under the use of cost functions to deal with the problem during the training phase in neural networks with unbalanced data. An approach about the inclusion of a priori information in the cross-entropy cost function is presented together with a modification of the resilient backpropagation algorithm and the impacts on the learning algorithm.Because of the difference in the number of observations between classes in unbalanced class problems, measuring the performance of the learning algorithm requires more appropriate metrics such as AUC, F1-score, Kubat's G-mean (Geometric-mean), AGm (Adjusted Geometric-mean) and others. However, the vast majority of problems in this area are trained using the mean square error or cross-entropy (also known as logistic error function). This makes the neural network learning algorithm to optimize a cost function different from the one that will be used to validate its performance. An approach is then presented on how to extract appropriate metrics for this kind of problem from the confusion matrix and transform them into cost functions to be used during the training phase. A comparative study between the traditional training approach and the presented cost functions is carried out, presenting the positive and negative points of each approach. Numerical experiments for different training databases with different unbalanced rates are presented.

Dedico este trabalho a Deus, pela oportunidade que me deu, e à minha família, pelo amor incondicional. Dedico à minha noiva pelos cuidados e amor. Dedico aos professores que sempre me apoiaram e aos amigos que sempre me sustentaram. Dedico à família do doador que me deu a chance de um novo coração poder bater dentro de mim e me trazer até aqui.

"The sad thing about artificial intelligence is that it lacks artifice and therefore intelligence."
Jean Baudrillard

Agradecimentos

Agradeço a Deus pelo dom da vida e as oportunidades que me deu.

À minha família, especialmente aos meus pais, por todo o amor e cuidado.

À minha noiva, Nathália, pelo amor e carinho na caminhada.

Ao meu orientador, Antônio de Pádua Braga, por ter me acolhido em um momento difícil, por sempre acreditar em mim, me apoiar e me fazer melhor.

Ao meu coorientador, Cristiano Leite Castro, pelos ensinamentos, paciência e direção.

Aos professores e colegas do LITC: por todos os ensinamentos, companheirismo e ajuda prestada durante o meu mestrado.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

Vivemos um momento de crescimento massivo da quantidade de informação gerada e armazenada. Essa produção em grande escala dos dados, ainda que de forma muitas vezes desorganizada, gera múltiplas oportunidades de abordagem e exploração de técnicas de aprendizado de máquinas. Todavia, a quantidade de dados de qualidade produzidos não consegue acompanhar esse crescimento. Isso faz com que, muitas vezes, a informação de qualidade seja limitada ou tenha custo elevado. Com isso, em muitos casos, existe um excesso de dados que leva a um desbalanceamento massivo entre a informação buscada e a disponível.

Esse crescimento acarretou um aumento do número de problemas conhecidos na área de aprendizado de máquinas, no que tange à classificação, como problemas de classes desbalanceadas. Tal tema passou a atrair uma parcela de interesse significativo de organizações industriais e acadêmicas. Diversos trabalhos sobre o tema começaram, então, a surgir em diferentes áreas do conhecimento (Bacha et al., 2008; Zheng, 2010; Zhao et al., 2008; Arar and Ayan, 2015; Cho et al., 2008; Fallahi and Jafari, 2011; Song et al., 2014).

Com o aumento da capacidade de processamento e armazenamento de dados, as redes neurais começaram a ganhar ainda mais destaque no cenário acadêmico e industrial. Esse crescimento, em conjunto, das redes neurais e do problema de classes desbalanceadas atraiu o interesse do autor e seus orientadores para o estudo aqui apresentado, sobre diferentes técnicas utilizadas para lidar com o problema das classes desbalanceadas em redes neurais.

A maioria dos estudos nesta área lida com o problema por meio de três estratégias distintas: pré-processamento de dados, *ensembles* e manipulação algorítmica. Algumas dessas técnicas são resumidas em Michie et al. (1994), Chawla et al. (2002), Chawla

et al. (2004b), Chawla et al. (2004a), He et al. (2009), Lan et al. (2010), Thai-Nghe et al. (2010) e Batista et al. (2004). Todavia, técnicas de pré-processamento como *Synthetic Minority Oversampling Technique* (SMOTE)(Chawla et al., 2002), *Weighted Wilson's Editing* (WWE)(Barandela et al., 2004) e *Adaptive Synthetic Sampling* (ADASYN)(He et al., 2008), conforme apresentado em Castro and Braga (2013), não apresentam uma melhoria significativa nas redes neurais de *perceptron* multicamadas (MLP - do inglês, *multilayer perceptron*).

Diferentemente do primeiro grupo, as técnicas baseadas em *ensembles* têm mostrado algumas melhorias na arquitetura de MLPs. Essas técnicas vêm atraindo muita atenção, uma vez que são capazes de lidar com o problema do desbalanceamento, obtendo bons resultados, como observado em Chen et al. (2010), Chawla et al. (2003), Guo and Viktor (2004) e Sun et al. (2007). No entanto, essa abordagem tem um sério inconveniente quando se trata de MLP: o tempo computacional elevado. Por outro lado, as técnicas baseadas em manipulação algorítmica podem lidar com problemas de dados desequilibrados na arquitetura dos MLPs e não alteram significativamente o tempo computacional, especialmente aquelas que consideram funções de penalidade distintas para distinguir a importância da classe (abordagem sensível ao custo) (Castro and Braga, 2013; Castro and de Pádua Braga, 2009; Alejo et al., 2007; Oh, 2011).

O trabalho aqui apresentado analisa o efeito da manipulação algorítmica sobre as funções de custo das redes neurais a serem otimizadas e como essa alteração influencia o resultado e a curva de aprendizado da rede neural. Em um primeiro momento, é apresentado o efeito da incorporação da informação *a priori* de cada classe a uma função de entropia cruzada, e é proposta uma nova abordagem usando o algoritmo de *backpropagation* com uma modificação no *Resilient Backpropagation*(Rprop) (Riedmiller and Braun, 1993), de forma que a otimização da função ocorra de forma suave e com menos parâmetros a serem otimizados.

Todavia, existem situações em que o modelo de desempenho e avaliação de sucesso de um algoritmo é medido de forma específica, seja por métricas próprias para análise de problemas de classes desbalanceadas, como *Geometric-mean* (G-mean), F1-score, Área sob a curva (AUC, do inglês *Area Under the ROC Curve*), seja por funções focadas no problema, como *Adjusted Geometric-mean* (AGm) e outras. Diante dessa questão, é apresentada uma abordagem para criação de funções de custo específicas, as quais podem ser construídas com os termos da matriz de confusão. Dessa forma, é possível utilizar diferentes funções/métricas de problemas de balanceamento como funções de custo, de tal forma que a métrica a ser utilizada para avaliação do modelo seja a mesma utilizada pela rede neural como função de custo a ser otimizada. Essa abordagem pode ser utilizada e generalizada para problemas e gama de algoritmos além dos apresentados neste trabalho.

1.2 Principais contribuições deste trabalho

Dado o crescimento constante das redes neurais, acompanhado por uma predominância cada vez maior de problemas de dados desbalanceados, este trabalho apresenta as seguintes contribuições:

- Um estudo aprofundado da utilização da informação *a priori* na função de custo de entropia cruzada (*cross-entropy*) em redes neurais para problemas de classes desbalanceadas.
- Uma nova abordagem da implementação de função de custo sensível na função de erro de entropia cruzada com Rprop modificado.
- Uma abordagem para extração dos termos da matriz de confusão para transformação das métricas de análise em funções de custo.
- Uma análise de diferentes funções de custo baseada em métricas de análise de desempenho do treinamento de redes neurais com classes desbalanceadas.
- O artigo: Aurelio, Yuri, Braga, A.P., and Castro, C.L (2016). Função de custo logística sensível para melhorar performance da rede perceptron de múltiplas camadas. In *XII Simpósio de Mecânica Computacional*.
- O artigo: Aurelio, Yuri, Braga, A.P., and Castro, C.L. Learning from Imbalanced Data Sets with Weighted Cross-Entropy Cost-Sensitive Function. (Em submissão - *Neural Processing Letters*).
- O artigo: Aurelio, Yuri, Braga, A.P., and Castro, C.L. Multilayer Perceptron functions based in metrics to unbalanced learning problems. (Em fase de escrita - resultados já gerados).
- A disponibilização das funções de custo implementadas para a comunidade em https://github.com/yurisousa/cost-functions_MLP_MATLAB.

1.3 Organização do texto

Uma revisão teórica é apresentada para tratar das principais estratégias adotadas na última década para lidar com o problema de dados desbalanceados, com ênfase no aprendizado de redes neurais.

A primeira parte do trabalho trata da introdução de informação *a priori* em funções de custo, e apresenta uma nova abordagem para a agregação desta à função de erro de entropia cruzada com Rprop modificado.

A segunda parte expõe uma abordagem para extração dos termos da matriz de confusão para transformação das métricas de análise em funções de custo, fazendo-se uma análise exploratória dessas métricas em problemas de redes neurais com dados desbalanceados.

Capítulo 2

Revisão teórica

O interesse no campo da inteligência artificial tem crescido de forma exponencial nos últimos anos, principalmente em relação ao aprendizado de máquinas. Dessa forma, diversos novos problemas, que antes não eram tratados com aprendizado de máquinas, começaram a utilizar essa técnica. Soma-se a isso o crescimento massivo do número de informações disponíveis, devido, em grande parte, ao aumento do número de sensores e de algoritmos de captação de dados.

Muitos dos problemas que antes não eram tratados com aprendizado de máquinas possuem uma natureza de dados desbalanceada, como por exemplo: detecção de doenças, genes defeituosos, fraude fiscal, transação de cartão duvidosa etc; ou seja, problemas onde um evento (classe) possui um número relativamente menor de ocorrências (amostras) do que os demais eventos (classes) sob observação. Dessa forma, algoritmos que sejam insensíveis ao desbalanceamento dos dados, ou que sejam específicos para dados desbalanceados, têm chamado a atenção da indústria e da academia.

Os algoritmos que tratam de problemas de classes desbalanceadas podem ser divididos em três categorias: pré-processamento de dados, *ensemble* e manipulação algorítmica. Há também abordagens que fazem uso da combinação de duas ou mais dessas. A seguir será exposta, brevemente, cada uma dessas três abordagens.

2.1 Pré-processamento de dados

A expressão "pré-processamento de dados"pode remeter a diferentes tipos de abordagem, por exemplo: eliminação de dados faltantes, normalização dos dados, criação e eliminação de variáveis etc. Todavia, neste trabalho, "pré-processamento"refere-se à criação de abordagens que tenham como objetivo mitigar o efeito que o desbalanceamento dos dados possa causar no treinamento de um modelo de aprendizado de máquinas. Dentre as abordagens deste tema, destacam-se as técnicas de *undersam-*

pling (eliminação de amostras) e *oversampling* (criação de amostras aleatórias).

Dentre as técnicas de *undersampling*, a mais utilizada consiste simplesmente em eliminar aleatoriamente amostras da classe majoritária, até que o número de amostras dessa classe se iguale ou se aproxime do número de amostras da classe minoritária. Contudo, um dos grandes problemas nesse tipo de abordagem é que a eliminação de amostras causa uma grande perda de informação e valor dos dados. Tal perda pode fazer com que o classificador tenha desempenho abaixo do esperado, por apresentar uma variabilidade de dados menor do que poderia, principalmente se as amostras eliminadas estiverem próximas à fronteira de separação ou em uma região de baixa densidade de dados.

Existem algumas técnicas de *undersampling* mais sofisticadas, que tendem a diminuir os efeitos negativos da eliminação de amostras da classe majoritária, por exemplo: WWE (Barandela et al., 2003, 2004), *NearMiss-1*, *NearMiss-2*, *NearMiss-3* (Mani and Zhang, 2003), *One-sided selection (OSS)* (Kubat et al., 1997) e *Condensed nearest neighbor rule (CNN)* (Hart, 1968). Apesar de promissoras, principalmente quando se tem um número massivo de dados, essas técnicas de abordagem mais sofisticada de *undersampling* em pré-processamento não são muito utilizadas pela academia nem pela indústria, devido ao custo computacional e o fato de poderem gerar perda de informação.

Diferentemente do conceito de *undersampling*, as técnicas de *oversampling* consistem na criação de novas amostras da classe minoritária, a fim de igualar ou aumentar o número de amostras da classe majoritária. De forma sucinta, pode-se dizer que, ao criar novas amostras, parecidas com as existentes da classe majoritária, tende-se a aumentar o efeito que a classe minoritária terá durante a fase de treinamento do modelo de aprendizado de máquinas. Dentre as técnicas de criação de novas amostras, baseadas no conjunto de dados da classe minoritária, destacam-se os algoritmos conhecidos como SMOTE (Chawla et al., 2002), SMOTE + Tomek *links* (Tomek, 1976; Batista et al., 2003) e ADASYN (He et al., 2008). Um dos maiores questionamentos quanto à utilização de técnicas de criação de amostras aleatórias advém do fato de que, em algumas situações, a criação desse tipo de amostra não representa a realidade, e pode gerar uma variabilidade muito grande nos dados, podendo causar um efeito de *underfitting*. Apesar desse questionamento, esse tipo de abordagem foi muito utilizado pela academia e pela indústria na última década, e apresenta resultados interessantes.

A seguir é apresentada, de forma resumida, a técnica de *undersampling* WWE, bem como as técnicas de *oversampling* SMOTE, SMOTE + Tomek *links* e ADASYN, as quais serão consideradas direta ou indiretamente para efeito de comparação neste trabalho.

2.1.1 WWE: *Weighted Wilson's Editing*

Um dos grandes problemas das técnicas de *undersampling* consiste em eliminar amostras da classe majoritária que possam conter informação importante para o treinamento do classificador, gerando perda de informação significativa. A fim de reduzir esse efeito adverso, Barandela et al. (2003) introduziram um algoritmo, baseado na ideia de Wilson (1972), cuja abordagem consiste em utilizar a regra do vizinho mais próximo (*nearest neighbor rule*) para eliminação de ruído.

A abordagem de Wilson, conhecida como *Wilson's Editing*, consistia basicamente em utilizar k vizinhos mais próximos para estimar a classe de cada uma das amostras de todo o conjunto de treinamento e descartar aquelas cuja classe prevista era erroneamente classificada, de acordo com a regra do vizinho mais próximo. Todavia, a ideia de Wilson era utilizada com o objetivo principal de eliminar ruído, sendo empregada em todo o conjunto de treinamento e não apenas sobre a classe majoritária, não tendo como objetivo principal a redução no número de amostras da classe majoritária.

Apesar de não ser uma abordagem específica de *undersampling*, o algoritmo *Wilson's Editing* inspirou vários trabalhos no campo de pré-processamento, como o *Weighted Wilson's Editing* (Barandela et al., 2003, 2004), a trabalharem com a abordagem do vizinho mais próximo.

Diferente do algoritmo que o inspirou, o WWE foi proposto com o objetivo de combater problemas de treinamento de dados desbalanceados. Muito parecido com seu antecessor, o WWE consiste, sucintamente, em aplicar o algoritmo *Wilson's Editting* com uma métrica de distância diferente da euclidiana, a qual pode ser verificada na equação 2.1.

$$d_W(Y, x_0) = (N_i/N)^{1/m} \cdot d_E(Y, x_0) \tag{2.1}$$

em que:

Y : novo padrão a ser classificado

x_0 : amostra de treinamento da classe i

N_i : número de amostras da classe i

N : número de amostras do conjunto de treinamento

m : dimensionalidade do espaço de características

d_E : distância euclidiana.

Dessa forma, a nova distância cria uma tendência para que as amostras a serem classificadas pelos k vizinhos mais próximos encontrem seus vizinhos mais próximos entre as amostras da classe minoritária. Logo, existe uma tendência a diminuir o número de amostras da classe majoritária muito maior do que no algoritmo *Wilson's Editing*. Apesar de apresentar resultados interessantes, um dos problemas dessa abordagem é que ela não garante que ocorrerá um balanceamento das classes.

2.1.2 SMOTE: *Synthetic Minority Oversampling Technique*

Antes da introdução do algoritmo conhecido como SMOTE (Chawla et al., 2002), a abordagem de *oversampling* mais comumente utilizada era baseada na amostragem da classe minoritária com repetição. Todavia, trabalhos como o de Japkowicz et al. (2000) haviam mostrado que esse tipo de estratégia não apresentava uma melhoria significativa no reconhecimento da classe minoritária, principalmente quando utilizado como modelo de pré-processamento para algoritmos de árvores, em que a replicação das amostras não causava um aumento na fronteira de decisão da classe com menor número de amostras (Chawla et al., 2002).

O algoritmo SMOTE foi criado, então, com o objetivo de gerar um aumento no número de amostras da classe minoritária por meio da criação de amostras sintéticas, as quais tivessem um impacto positivo na fronteira de decisão após o treinamento dos dados. Com esse objetivo, para cada amostra da classe minoritária, o algoritmo busca os k vizinhos mais próximos e seleciona n dentre estes aleatoriamente, criando uma amostra sintética entre o segmento que liga a amostra selecionada a cada uma das n amostras aleatórias dentre os k vizinhos mais próximos. Por exemplo, se o número de amostras sintéticas totais a serem criadas for de 200% do número de elementos da classe minoritária, somente dois dos k vizinhos mais próximos são selecionados, e as amostras sintéticas são criadas da seguinte maneira:

1. calcula-se a diferença da amostra sob consideração em relação ao seu vizinho mais próximo;

2. multiplica-se este valor por um número aleatório entre 0 e 1;

3. adiciona-se este valor ao vetor de características da amostra sob consideração.

No trabalho aqui apresentado, o número de k vizinhos mais próximos adotado, quando o SMOTE for utilizado, será de $k = 5$ (valor padrão proposto), e o número de amostras sintéticas utilizadas para cada uma das bases de dados está representado na tabela 2.1.

Tabela 2.1: Número de amostras sintéticas por base de dados

Base de dados	Alias	%. de amostras sintéticas
Ionosphere	iono	100%
Pima Indians Diabetes	pid	100%
German Credit	gmn	100%
WP Breast Cancer	wpbc	200%
Vehicle (4 versus. all)	veh	200%
SPECTF Heart	hrt	300%
Segmentation (1 versus. all)	seg	500%
Glass (7 versus. all)	gls7	500%
Euthyroid (1 versus. all)	euth	600%
Satimage (4 versus. all)	sat	800%
Vowel (1 versus. all)	vow	900%
Abalone (18 versus. 9)	a18-9	1000%
Yeast (9 versus. 1)	y9-1	2000%
Car (3 versus. all)	car	2500%
Yeast (5 versus. all)	y5	2000%
Abalone (19 versus. all)	a19	10000%

O percentual de amostras sintéticas presentes na tabela 2.1 foi definido de acordo com critérios de *benchmarket*, também presente em (Castro and Braga, 2013).

2.1.3 SMOTE + Tomek *links*

A abordagem compreendida como SMOTE + Tomek *links* consiste em implementar a sobreamostragem da classe minoritária utilizando dados sintéticos e, assim, realizar um método de "limpeza"dos dados por meio do algoritmo Tomek *links*.

De forma simplificada, Tomek *links* podem ser definidos como um par de vizinhos mais próximos minimamente distanciados das classes opostas (He et al., 2009). Isto é, se a distância entre duas amostras (x_i, x_j) em que $x_i \in S_{minoritrio}, x_j \in S_{majoritrio}$ for menor que a distância entre (x_i, x_k) e (x_j, x_k), em que $x_k \in S$, logo, x_i e x_j são considerados um par de Tomek *links*. Em Tomek (1976), se dois pares de amostras representam um par de Tomek *links*, então, ou uma dessas amostras é ruído, ou as duas estão próximas à borda.

Na abordagem combinada de SMOTE + Tomek *links* introduzida em Batista et al. (2003), os seguintes passos são implementados:

1. aplica-se o algoritmo de SMOTE;
2. identificam-se os pares de Tomek *links*;
3. eliminam-se as amostras identificadas como pares de Tomek *links*.

Esse tipo de abordagem foi utilizado pela primeira vez a fim de melhorar o desempenho de classificadores para o problema de anotação em proteínas em Bioinformática (Batista et al., 2003).

2.1.4 ADASYN: *Adaptative Synthetic*

Segundo os autores desta abordagem, o ADASYN tem como ideia utilizar a distribuição ponderada para diferentes exemplos da classe minoritária, de acordo com a dificuldade do aprendizado. Quanto maior a dificuldade do modelo em aprender uma amostra, maior será o número de dados sintéticos gerados a partir desta.

Na implementação do ADASYN, primeiramente calcula-se o número de amostras sintéticas que precisa ser gerado para a classe minoritária:

$$G = (|S_{maj}| - |S_{min}|) \cdot \beta \tag{2.2}$$

em que $\beta \in [0, 1]$ é um parâmetro usado para especificar o nível de balanceamento desejado após a criação das amostras sintéticas. Quando $\beta = 1$, o balanceamento total do conjunto de dados é criado após o processo de geração de amostras sintéticas. O próximo passo consiste em encontrar os k vizinhos mais próximos de cada uma das amostras $\mathbf{x_i} \in S_{min}$, de acordo com a distância euclidiana, e calcular a taxa ψ, como definido:

$$\psi_i = \frac{\Delta_i / K}{Z}, \quad \mathtt{i} = 1, ..., |S_{min}| \tag{2.3}$$

em que Δ_i é o número de exemplos da classe majoritária entre os k vizinhos mais próximos de x_i, e Z é a constante de normalização tal que ψ seja uma função de distribuição ($\sum \psi_i = 1$). Então, determina-se o número de amostras sintéticas que precisa ser gerado para cada $\mathbf{x_i} \in S_{min}$:

$$g_i = \psi_i \cdot G \tag{2.4}$$

Por último, para cada $\mathbf{x_i} \in S_{min}$, criam-se g_i amostras sintéticas de forma semelhante ao SMOTE.

Como visto em He et al. (2009), a ideia principal do ADASYN é utilizar a distribuição de densidade ψ como um critério de decisão automático para o número de amostras sintéticas que precisam ser geradas para cada amostra da classe minoritária, mudando de forma adaptativa o peso das diferentes amostras da classe minoritária, de acordo com a distorção das distribuições.

2.2 *Ensemble*

Ensemble consiste em uma estratégia de combinar diferentes[1] tipos de algoritmos e modelos com o objetivo de formar um classificador mais robusto. Os *ensembles* podem ser separados em dois grupos: combinação paralela e combinação sequencial de modelos.

2.2.1 Combinação paralela

Na combinação paralela de modelos, os algoritmos podem ser treinados com um mesmo conjunto de dados ou com um subconjunto do total dos dados. Após serem treinados, os algoritmos são então combinados, em geral, de duas formas: votação/média ou combinação com aprendizado.

No modelo de votação, cada um dos modelos opina (realiza um voto) sobre a classe de cada uma das amostras a serem preditas. Na votação majoritária, aquela classe que receber o maior número de votos é considerada a classe prevista. Todavia, caso se queira estabelecer um peso distinto para o voto de cada um dos modelos, a votação passa a ser chamada de "votação ponderada". De forma análoga ao modelo de votação, a média simples consiste na média do resultado dos classificadores, e na média ponderada há uma ponderação da saída de cada um dos modelos antes de tirar a média final.

Diferentemente do modelo de votação/média, alguns *ensembles* utilizam outros tipos de modelo de aprendizado de máquinas, para combinar as saídas dos modelos treinados em paralelo.

Essa abordagem de combinação paralela é muito utilizada para modelos de árvores como *bagging* (Breiman, 1996) e *random forests* (Breiman, 2001). Todavia, essa abordagem não é muito popular nos modelos de redes neurais[2], que são o foco do trabalho aqui apresentado.

2.2.2 Combinação sequencial

Os *ensembles* que fazem combinação sequencial de algoritmos e modelos são conhecidos como *boosting*. Essa abordagem baseia-se na ideia de combinar sequencialmente fracos modelos de aprendizado, os quais, a cada nova tentativa, focam em prever as amostras nas quais o classificador anterior apresentou uma dificuldade de predição.

Diferentes modelos utilizando *boosting* surgiram na última década, muitos combinando abordagens de pré-processamento com *boosting*, como aqueles apresentados por

[1] Pode ser o mesmo modelo treinado diversas vezes.

[2] *Ensembles* paralelos estão começando a ser utilizados em redes neurais profundas, mas de forma diferente da discutida nesta seção. Todavia, esse tipo de estrutura de redes complexas não está em foco nesta dissertação.

Chawla et al. (2003), Guo and Viktor (2004) e Chen et al. (2010). Grande parte dessas abordagens foi baseada no modelo conhecido como AdaBoost (Freund and Schapire, 1997).

A ideia do AdaBoost consiste na modificação dos dados a cada chamada, aplicando-se pesos para cada uma das amostras de treino. Inicialmente, esses pesos w são definidos para cada uma das amostras como sendo $w_i = 1/N$, em que N é o número total de amostras da base de treinamento. Em cada uma das iterações sucessivas, o peso de cada uma das amostas é modificado e o algoritmo é reaplicado sobre os dados com os novos pesos. Em um determinado passo, as amostras de treinamento que foram preditas incorretamente pelo modelo no passo anterior aumentaram seus pesos, enquanto os pesos foram diminuídos para aquelas que foram classificadas corretamente. À medida que novas iterações são realizadas, as amostras que são difíceis de prever recebem uma ênfase maior, de forma que o classificador é forçado a focar nas amostras que foram erroneamente previstas no classificador anterior.

Dentre os modelos de *ensemble*, um que apresentou resultados interessantes utilizando redes neurais simples foi o RAMOBoost, o qual consiste na combinação de um modelo de pré-processamento semelhante ao ADASYN (desenvolvido pelos mesmos autores) junto a uma versão modificada do AdaBoost, conhecida como AdaBoost.M2 (Freund et al., 1996).

2.3 Manipulação algorítmica

Outro tipo de estratégia que vem crescendo nos últimos anos, quando se trata de dados desbalanceados, é a manipulação algorítmica. Em resumo, trata-se de criar modificações nos algoritmos de aprendizado de máquinas para que eles possam compensar o desbalanceamento das classes. Essas modificações podem ser feitas de quatro maneiras distintas (Kukar et al., 1998): modificações sensíveis ao custo aplicadas à estimativa probabilística; modificação da saída da rede neural com função de custo sensível; funções de custo sensível implementadas sobre a taxa de aprendizado λ; e, por último, modificação da função de custo a ser minimizada pelo otimizador da rede neural.

Esta dissertação concentra-se na quarta abordagem de manipulação algorítmica para compensar o desbalanceamento das classes. Tal abordagem tem como foco a criação e alteração das funções de custo a serem minimizadas pela rede neural, como visto em Castro and Braga (2013), Fan et al. (1999), Domingos (1999), Zhou and Liu (2006) e Sordoni et al. (2015).

A escolha de trabalhar com apenas este tipo de abordagem é devido à flexibilidade que a mesma apresenta e a simplicidade de adaptação aos algoritmos existentes.

Capítulo 3

Uso da informação a priori em funções de custo em redes neurais

Este capítulo apresenta uma abordagem alternativa para tratar o problema do conjunto de dados desequilibrados nas redes neurais. O modelo leva em consideração a probabilidade anterior de cada classe, a fim de criar uma função sensível ao custo com base na função de erro de entropia cruzada com uma versão modificada do Rprop. O *benchmarking* com vários conjuntos de dados mostrou a eficácia e robustez do método ante diferentes proporções de desequilíbrio. Os ganhos de desempenho foram avaliados com métricas diferentes, como G-mean, Agm e a AUC característica de operação do receptor.

3.1 Introdução

Vários estudos surgiram para abordar o problema de classes desbalanceadas nos últimos anos. A maioria dos algoritmos de aprendizagem considera o erro de cada amostra igualmente importante, de modo que o problema de aprendizagem se reduz a minimizar o erro geral de classificação, o que leva ao detrimento da taxa de sucesso da classe minoritária ao lidar com um conjunto altamente desequilibrado. Apesar de esses algoritmos clássicos terem uma alta acurácia, eles têm uma taxa de sucesso baixa para a classe minoritária, que na maior parte do tempo é a classe de maior interesse, por exemplo: detecção de falhas, fraude, doenças etc.

Assim, a fim de lidar com dados desequilibrados em redes neurais MLP levando em consideração as características de cada um, foi levantado neste trabalho o efeito da incorporação da informação *a priori* de cada classe a uma função de entropia cruzada, e foi proposta uma nova abordagem usando o algoritmo de *backpropagation* com uma modificação no Rprop (Riedmiller and Braun, 1993).

3.2 Aprendizado em classes desbalanceadas

Em problemas de classificação, os marcadores de saída $\mathbf{y}_i$ no conjunto de treinamento $\mathbf{S} = \{(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{X} \times \mathcal{Y} \,|\, i = 1, \ldots, N\}$ são obtidos de acordo com uma função geradora desconhecida $f(\mathbf{x})$. O objetivo é, então, estimar o modelo $f(\mathbf{x} \,|\, \theta)$ que seja o mais próximo possível de $f(\mathbf{x})$. Ao invés de adotar uma função de custo empírica baseada no erro médio quadrático (MSE, do inglês *Mean Square Error*), uma maneira de encontrar um conjunto ótimo de parâmetros θ que aproxime o modelo $f(\mathbf{x} \,|\, \theta)$ de $f(\mathbf{x})$ consiste em minimizar a função de erro logística (também conhecida como entropia cruzada ou *cross-entropy*) baseada no erro entre a saída alvo e o resultado da saída gerada pelo modelo dado por θ:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [-y_i \log(\hat{y}_i) - (1 - y_i) \log(1 - \hat{y}_i)] \tag{3.1}$$

em que $\hat{y} = f(\mathbf{x} \,|\, \theta)$ é a saída do modelo dado pelo aprendizado do mapeamento da função de entrada $\mathcal{X}$ no espaço de saída $\mathcal{Y}$.

A função de erro de entropia cruzada (Equação 3.1) é escolhida no lugar da função de erro médio quadrático por ser uma função convexa e por ser mais apropriada para cálculo da probabilidade *a posteriori* em redes neurais. Observa-se que, quando $y_i = 0$, a função de custo depende somente de $-\log(1-\hat{y}_i)$, mas se $y_i = 1$, a expressão torna-se $-\log(\hat{y}_i)$. Então, em ambas as situações, o erro decai de forma logarítmica à medida que $\hat{y}_i$ tende a y_i. Como se pode observar na Figura 3.1, dado que as duas curvas são simétricas, o erro tende a diminuir em uma taxa logarítmica igual em ambas as situações ($y_i = 0$ or $y_i = 1$). Dessa forma, nota-se que, quando as classes são balanceadas, o erro dado pelo termo $-\log(\hat{y}_i)$ tende a ser proporcional ao erro gerado por $-\log(1 - \hat{y}_i)$, assumindo-se que, para uma dada saída $\hat{y}$, ambos os termos serão responsáveis por 50% do erro total $J(\theta)$.

Entretanto, quando se trata de problemas de classes desbalanceadas, o termo responsável pela classe majoritária terá uma influência maior na função de custo total $J(\theta)$, em comparação com o termo responsável pela classe minoritária. Isso acontece porque o erro total, obtido como uma soma dos erros dados por ambos os termos, é minimizado independentemente de qual classe foi responsável pela sua composição.

Como exemplo, tome-se em consideração o problema de classificação binário da Figura 3.2, constituído de 200 amostras em cada uma das classes. Para este problema será utilizada uma rede neural *perceptron* de múltiplas camadas (MLP) com dois nós de entrada, dois neurônios na cama escondida e um na saída, treinada com o algoritmo clássico do *backpropagation* e função de erro de entropia cruzada da Equação 3.1. Esta rede neural é inicializada com os pesos $\theta \in [-0.001, 0.001]$ e taxa de aprendizado

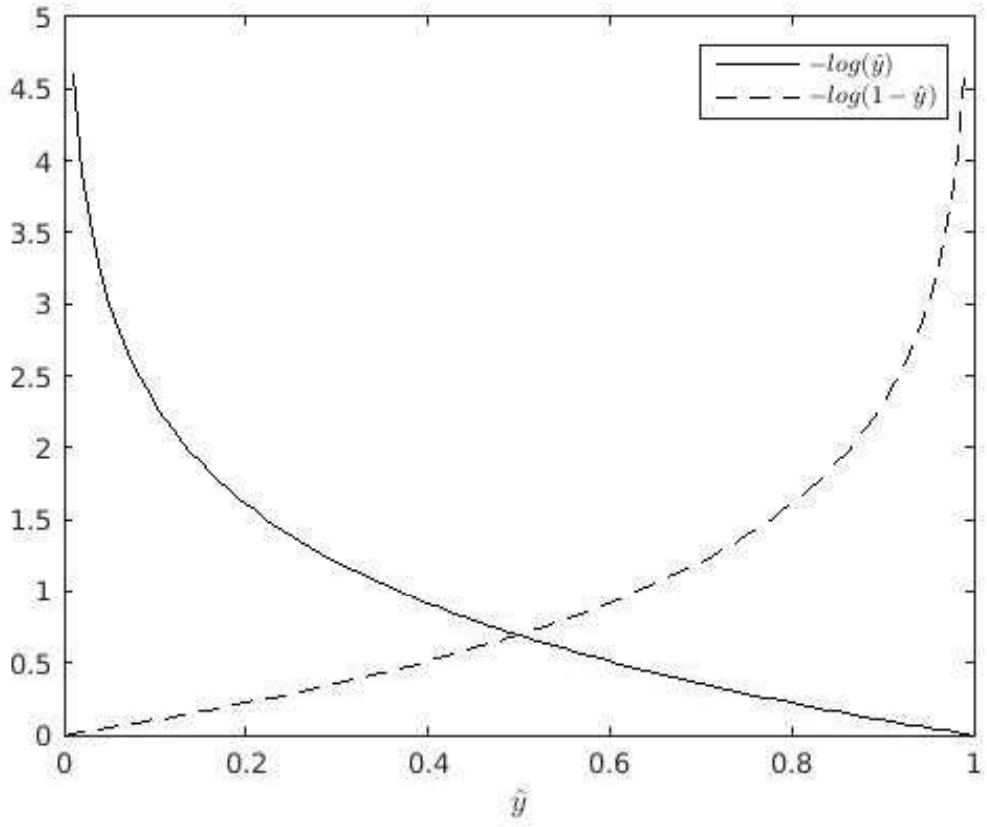

Figura 3.1: Ilustração da função de erro de entropia cruzada $J(\theta)$ para diferentes valores de saída $\hat{y}$ para y_i igual a um (linha sólida) e y_i igual a zero (linha pontilhada)

$\eta = 0.5$. Considerem-se também os casos de desbalanceamento, em que a Classe A possui $5, 50, 100$ amostras aleatórias do conjunto original e mesmas condições iniciais. Para avaliar a contribuição de cada classe para a função de erro de entropia cruzada, a proporção do erro (obtida através da divisão do primeiro pelo segundo termo da Equação 3.1) é dada pela Equação 3.2 ao longo de 1000 iterações, como apresentado na Figura 3.3.

$$R = \frac{-\mathbf{y}\log(\hat{\mathbf{y}})}{-(1-\mathbf{y})\log(1-\hat{\mathbf{y}})} \tag{3.2}$$

Observa-se que, em condições de desbalanceamento, a taxa de erro de entropia cruzada no começo do processo de aprendizado será menor que uma unidade e proporcional à taxa de desbalanceamento da informação *a priori* (número de amostras da classe positiva sob o número de amostras da classe negativa). Esse acontecimento deve-se ao fato de que, no começo do processo de aprendizado, o algoritmo cometerá erros para ambas as classes. Entretanto, como a classe majoritária possui um maior número de amostras, o termo responsável por esta assumirá uma maior parcela do erro

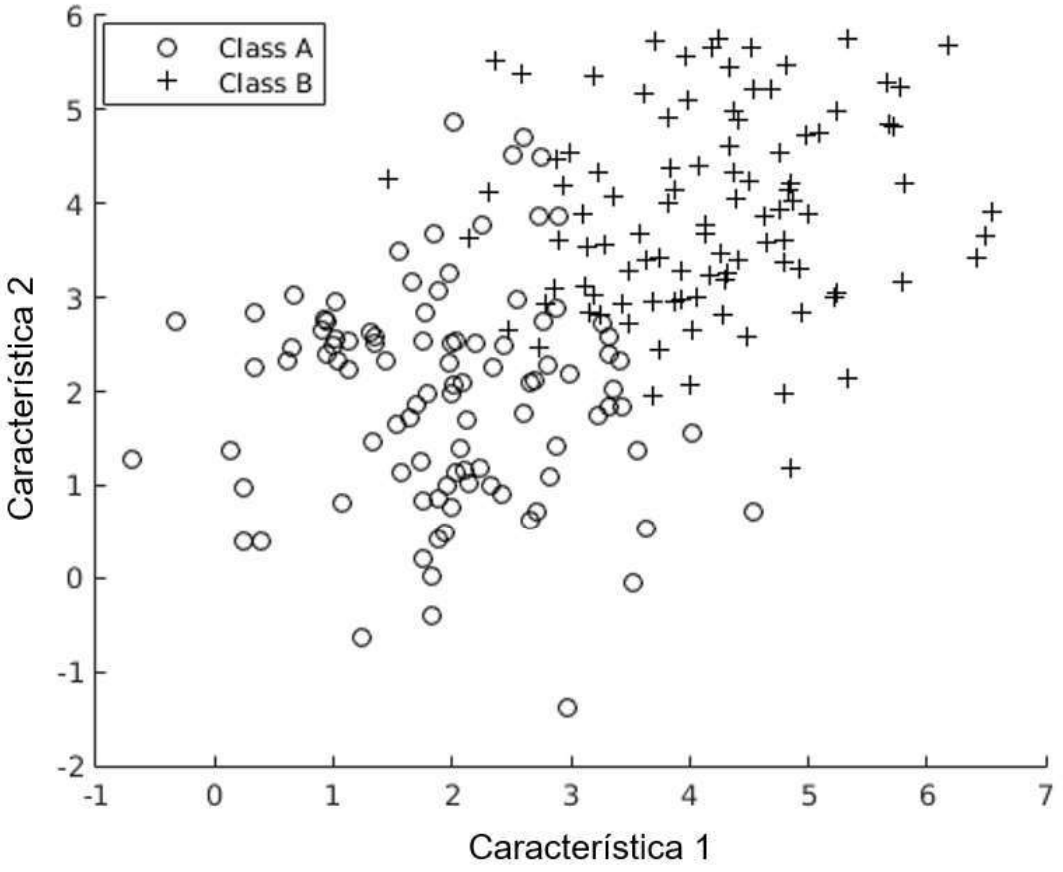

Figura 3.2: Problema de duas gaussianas

total. Isso fará com que o gradiente descendente aponte para um sentido que favoreça a classe de maior predominância, o que explica o aumento na proporção do erro da entropia cruzada.

3.3 Abordagem da entropia cruzada ponderada

A diferença entre as proporções de erro no conjunto de dados balanceado e desbalanceado pode ser contornada considerando a regra de decisão ótima (Equação 3.3). Em resumo, a equação diz que quando uma predição é feita, a saída deve ser balanceada de acordo com a informação *a priori* do número de amostras de cada classe. De acordo com a Equação 3.3 é esperado, ao final do processo de aprendizado, que a taxa $-\mathbf{y}\log(\hat{\mathbf{y}})$ sob $-(1-\mathbf{y})\log(1-\hat{\mathbf{y}})$ assuma um valor próximo da unidade no caso em que há balanceamento das classes, como é possível observar no gráfico 3.3. Entretanto, voltando aos casos do exemplo anterior, em que há desbalanceamento no conjunto de dados, a proporção do erro de entropia cruzada deveria assumir valores

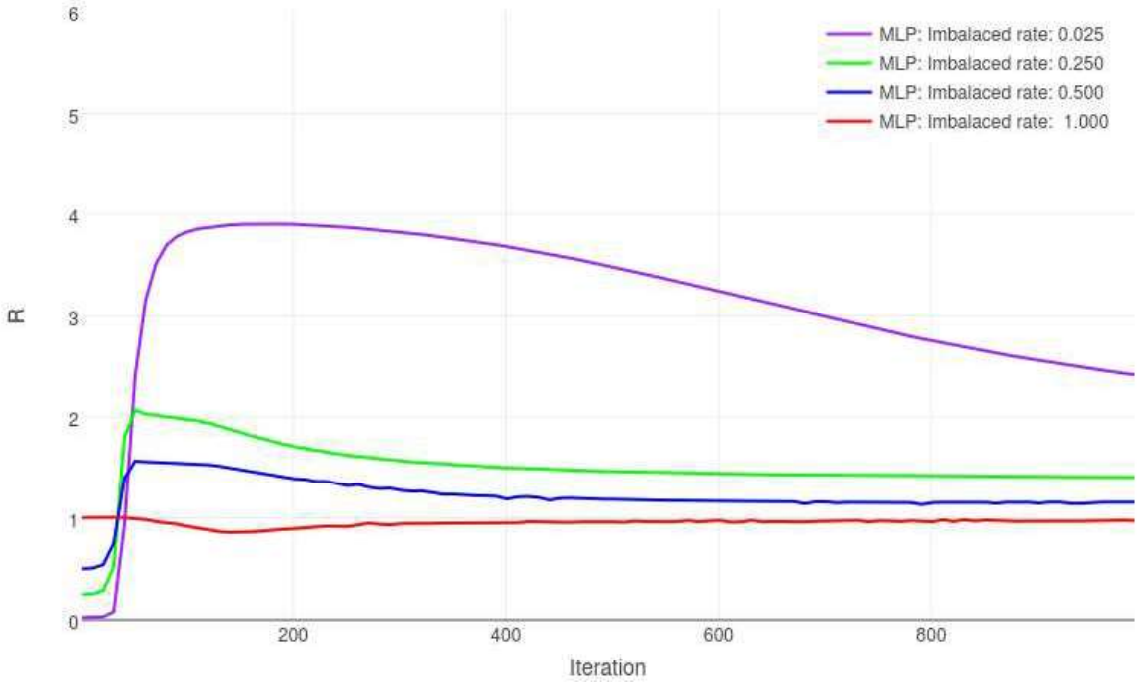

Figura 3.3: Taxa de erro de entropia cruzada para MLP

próximos a $0.025, 0.25, 0.5$ durante o processo de aprendizado, considerando que o algoritmo estabelecesse um valor de erro igual para todas as observações . De acordo com a Equação 3.3, todavia, esse valor ocorreu somente no início do processo. Assim, fica claro que a taxa de erro medida tende à unidade ao longo das iterações. Isso acontece porque, em algum ponto ao longo das iterações, o termo com menor participação terá um grande impacto no todo, tornando-se mais influente. Entretanto, ao atingir esse ponto, o algoritmo pode convergir devagar ou tornar-se estacionário.

$$f_0(x) = \begin{cases} 1, & \text{se } \frac{p(\mathbf{x}|\mathbf{y}=1)}{p(\mathbf{x}|\mathbf{y}=0)} \geq \frac{p(\mathbf{y}=0)}{p(\mathbf{y}=1)} \\ 0, & \text{caso contrário} \end{cases} \tag{3.3}$$

Analisando a Equação 3.3 e o trabalho presente em Castro and Braga (2013), fica claro que a introdução da informação *a priori* pode levar o algoritmo a uma solução mais equilibrada e benéfica à classe minoritária. Uma das maneiras de fazê-lo é incorporando a informação do número de amostras por classe na função de erro de entropia cruzada, como mostrado a seguir:

$$J(\theta) = \frac{1}{m} \sum_{i=1}^{m} [-y_i \log(\hat{y}_i)\lambda - (1 - y_i) \log(1 - \hat{y}_i)(1 - \lambda)] \tag{3.4}$$

$$\lambda^{(j)} = \left(\frac{N}{M}\right)^{-1} \tag{3.5}$$

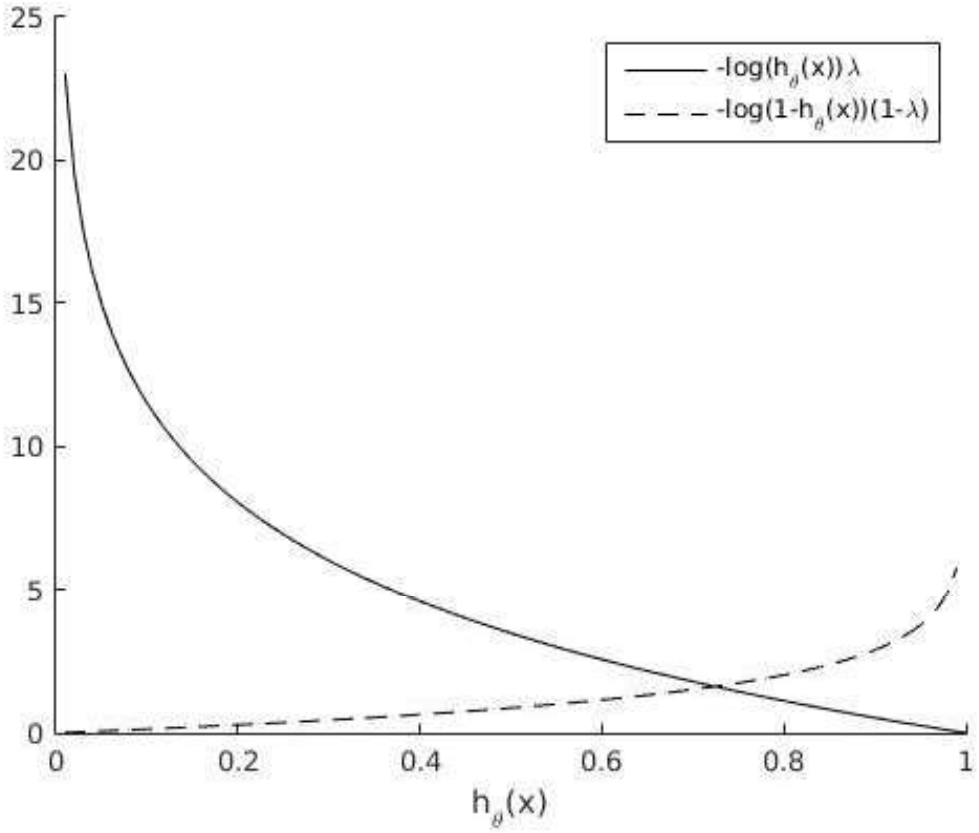

Figura 3.4: Saída da classe minoritária para a função de erro $J(\theta)$ de entropia cruzada com informação *a priori* para diferentes valores de saída do modelo $h_\theta(\mathbf{x}) = \hat{y}_i$ para $\mathbf{y_i} = 1$ (linha sólida) e $\mathbf{y_i} = 0$ (linha pontilhada)

em que N é o número de observações da classe minoritária, determinada pelo marcado de saída com o valor unitário, e M é o número total de observações presentes no conjunto de dados utilizado no treinamento.

Adotando-se a abordagem explicada e sintetizada pela Equação 3.4 e considerando-se a razão entre o número de observações da classe A sob o número de observações da classe B igual a 0.20, pode-se observar na Figura 3.4 que o erro oriundo do termo $\sum_{i=1}^{m} -y_i \log(\hat{y}_i)\lambda$ decairá mais acentuadamente do que aquele originado da parcela $\sum_{i=1}^{m} -(1 - y_i) \log(1 - \hat{y}_i)(1 - \lambda)$. Além disso, para qualquer $h_\theta(x)$ equidistante, $\sum_{i=1}^{m} -y_i \log(\hat{y}_i)\lambda$ será cinco vezes maior que $\sum_{i=1}^{m} -(1 - y_i) \log(1 - \hat{y}_i)(1 - \lambda)$.

Tomando-se a demonstração acima, aplica-se o gradiente à equação 3.4 a fim de obter $\partial J(\theta)/\partial \theta^{(n-1)}$ e $\partial J(\theta)/\partial z^{(n-1)}$.

$$\frac{\partial J(\theta)}{\partial \theta^{(n-1)}} = \left[\left(qg(z^{(n)}) - \lambda y\right) + yg(z^{(n)})(\lambda - q)\right] a^{(n-1)} \tag{3.6}$$

$$\frac{\partial J(\theta)}{\partial z^{(n-1)}} = \delta^{(n)} \theta^{(n-1)} g(z^{(n-1)})(1 - g(z^{(n-1)})) \tag{3.7}$$

$$\frac{\partial J(\theta)}{\partial z^{(n-1)}} = \delta^{(n-1)} \tag{3.8}$$

$$q = (1 - (N/M))^{-1} \tag{3.9}$$

$$qg(z^{(n)}) - \lambda y = \gamma \tag{3.10}$$

$$yg(z^{(n)})(\lambda - q) = \beta \tag{3.11}$$

$$\gamma + \beta = \delta \tag{3.12}$$

em que n é a última camada de neurônios.

Observando-se as equações do gradiente descendente aplicado à função de custo tradicional do erro de entropia cruzada, nota-se que para a implementação da abordagem aqui presente basta alterar o termo de erro (*error term*) da camada de saída $\delta^{(n)} = g(z^{(n)}) - y$ por $\delta(n) = \left(qg(z^{(n)}) - \lambda y\right) + yg(z^{(n)})(\lambda - q)$ para garantir que todas as outras condições do processo de *backpropagation* sejam garantidas, não havendo necessidade de alterações nas outras funções.

Aplicando-se a abordagem aqui desmonstrada ao problema das duas gaussianas, conforme o exemplo descrito anteriormente (ver Figura 3.2), nota-se pela Figura 3.5 que a taxa do erro de entropia cruzada (representada pela linha tracejada) permanece constante ao longo das iterações e, quando a informação *a priori* é adicionada à taxa do erro de entropia cruzada, a taxa de erro torna-se proporcional (assim como representado pela linha pontilhada do gráfico). Dessa forma, o algoritmo tende a ir de forma constante na direção de um equilíbrio entre as classes majoritária e minoritária.

Além dos resultados apresentados neste trabalho, pode-se inferir que a abordagem de inclusão da informação *a priori* na função de custo de entropia cruzada para problemas binários pode ser considerada uma alternativa para problemas de classes desbalanceadas, visto que essa abordagem faz com que o gradiente descendente vá em uma direção de maior equilíbrio.

Com o intuito de reduzir o número de variáveis, melhorar o desempenho do processo de aprendizado, garantir maior estabilidade e melhor compreensão acadêmica, uma estratégia dinâmica para atualização da taxa de aprendizado baseado no Rprop foi aqui introduzida e adotada para essa abordagem. O algoritmo Rprop não leva em consideração o valor absoluto fornecido pelo gradiente do erro, somente o sinal, para fazer a atualização dos pesos da rede neural. Entretanto, para a abordagem aqui apresentada, o módulo do gradiente do erro é muito importante e deve ser levado em consideração.

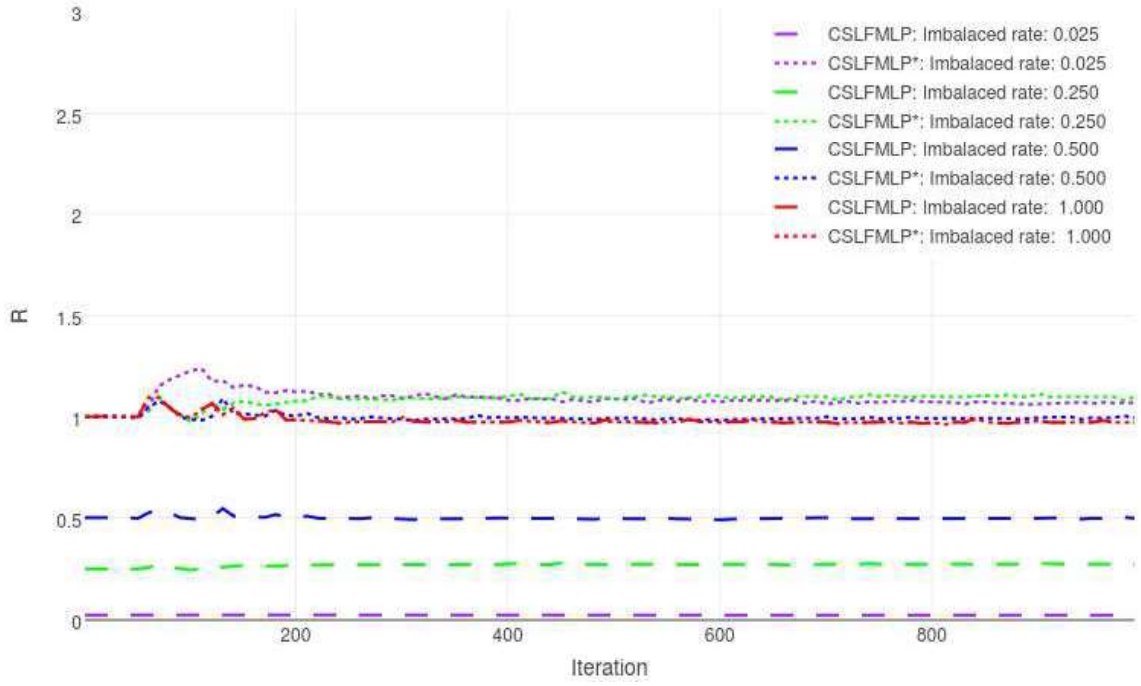

Figura 3.5: Taxa de erro de validação cruzada com informação *a priori* agregada em redes MLP

Logo, a estratégia adotada leva em consideração tanto a mudança do sinal do gradiente quanto o seu módulo para a atualização dos pesos da rede neural. Em resumo, essa modificação faz com que o valor do passo seja modificado de acordo com o erro ponderado da função de custo da entropia cruzada apresentada e suas mudanças de direções e sentido. Para sua implementação, a função sinal (*sign*) presente no algoritmo original do Rprop proposto em Riedmiller and Braun (1992) foi removida, assim como apresentado na Equação 3.13. Essa alteração permite que a taxa de aprendizado aumente ou diminua de acordo com a direção do gradiente e o módulo, o qual é impactado pela inclusão da informação *a priori*.

$$\begin{aligned} \Delta w_{ij}(t) &= -\mathbf{sign}\left(\frac{\delta E}{\delta w_{ij}}(t)\right) * \Delta_{ij}(t) \\ &\downarrow \\ \Delta w_{ij}(t) &= -\frac{\delta E}{\delta w_{ij}}(t) * \Delta_{ij}(t) \end{aligned} \tag{3.13}$$

Alguns benefícios da abordagem modificada do Rprop aqui apresentada, como a instabilidade no processo de aprendizado, também podem impactar a rede MLP padrão. As Figuras 3.3 e 3.5 foram geradas utilizando a abordagem do Rprop modificado. Observa-se que, utilizando-se a abordagem padrão, como demonstrado nas Figuras 3.6

e 3.7, há uma maior instabilidade da taxa de erro de entropia.

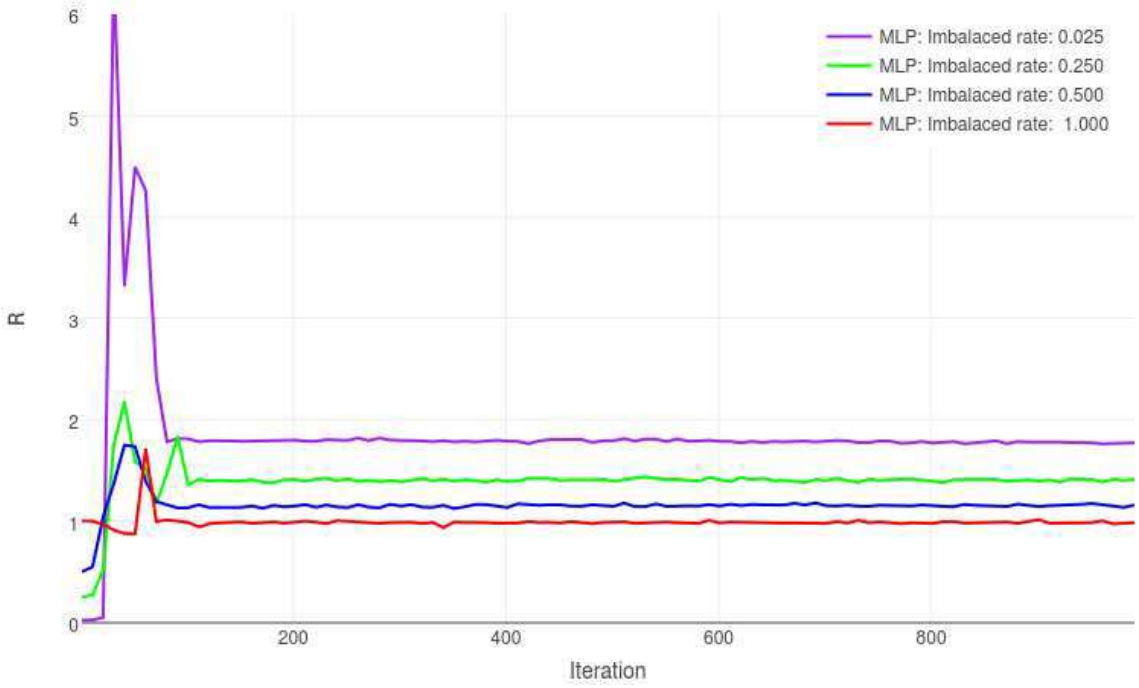

Figura 3.6: Taxa de erro de entropia cruzada para MLP com Rprop padrão

3.4 Metodologia do experimento e resultados

3.4.1 Metodologia do experimento

Um estudo empírico foi conduzido com 16 bases de dados diferentes do repositório da Universidade da Califórnia de Irvine (UCI), com seis métodos distintos: Rprop (Riedmiller and Braun, 1993), SMOTE (Chawla et al., 2002), SMOTE + Tomek *Links* (SMTTL) (Tomek, 1976), WWE (Barandela et al., 2004), RAMOBoost (Provost and Fawcett, 2001) e função de custo sensível com informação *a priori* acrescentada ao erro de entropia cruzada junto com Rprop modificado, aqui proposta e nomeada para fins práticos de *Cost-sensitive Logistic Function MLP* (CSLFMLP). As dezesseis bases de dados foram tratadas conforme apresentado em Castro and Braga (2013). As características de desbalanceamento das bases de dados podem ser encontradas na Tabela 3.1.

Algumas métricas usualmente utilizadas para avaliação de problemas de desbalanceamento foram selecionadas para avaliar o desempenho do comparativo aqui presente. São elas:

- Kubat's G-mean:
  Uma das melhores métricas para mensurar o balanceamento entre verdadeiros

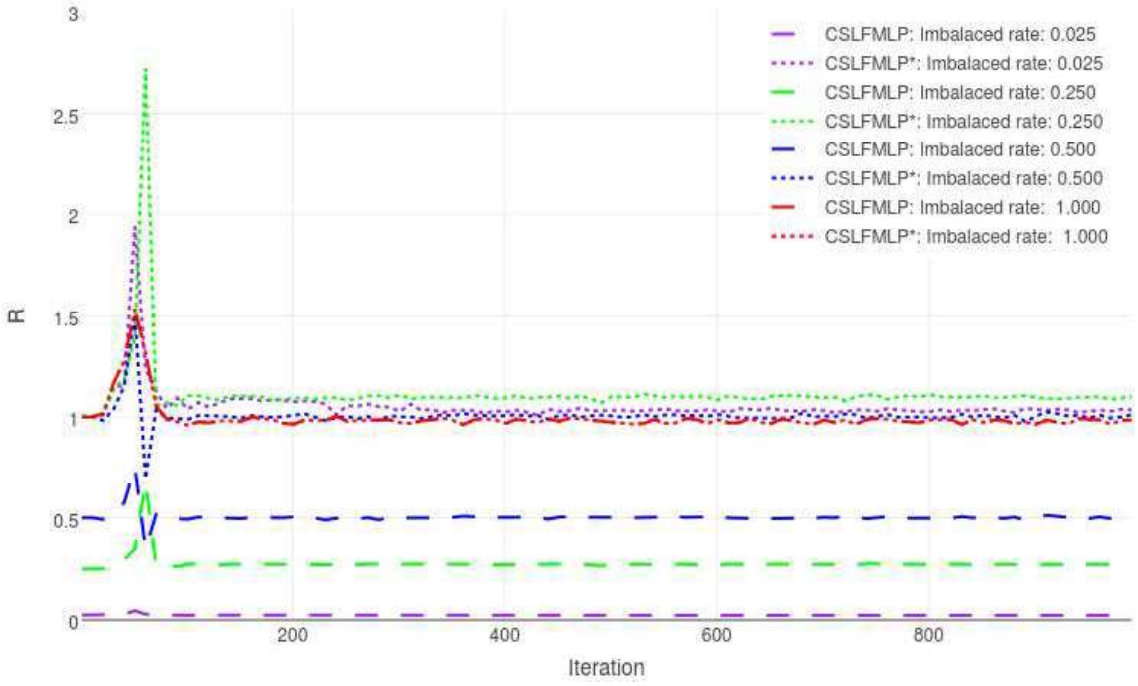

Figura 3.7: Taxa de erro de entropia cruzada com informação *a priori* agregada em redes MLP com Rprop padrão

positivos (TP, do inglês *True Positives*) e verdadeiros negativos (TN, do inglês *True Negatives*) devido ao fato de ser uma média geométrica entre ambas. É definida por $\sqrt{TPr \cdot TNr}$ (Kubat et al., 1997).

- Área sob a curva ROC (AUC, do inglês *Area Under the ROC Curve*):
  Uma das mais importantes da área, pois fornece uma estimativa de quão bem está sendo ranqueada a classe positiva (Fawcett, 2006).

- *Adjusted Geometric-Mean* (AGm):
  Métrica recente, proposta em Batuwita and Palade (2012), analisa a relação entre especificidade (SP, do inglês *Specificity*) e sensitividade (SE, do inglês *Sensitivity*), favorecendo esta.

Vinte casos distintos foram gerados para cada conjunto de dados, embaralhando-se os índices iniciais. Cada caso foi dividido em duas partes: 70% para treinamento e 30% para teste. A metodologia *k-fold* foi utilizada com $k = 7$ (assim com realizado em (Castro and Braga, 2013))para achar o melhor número de neurônios da camada escondida ($h^o = 1 : 3 : 13$). Após o processamento de cada *k-fold*, a média da métrica analisada foi selecionada. Logo, para cada caso, cinco valores distintos foram encontrados, um para cada h^0. Após todo o processo do *k-fold* para um conjunto de dados, uma matriz $20x5$ foi obtida. O melhor h^0 para cada caso (linha da matriz) foi selecionado, e sua

Tabela 3.1: Características das bases de dados

Base de dados	Alias	No. de atributos	n_1	n_2	$n_1/(n_1+n_2)$
Ionosphere	iono	34	126	225	0,359
Pima Indians Diabetes	pid	08	268	500	0,349
German Credit	gmn	24	300	700	0,3
WP Breast Cancer	wpbc	33	47	151	0,237
Vehicle (4 versus. all)	veh	18	199	647	0,235
SPECTF Heart	hrt	44	55	212	0,206
Segmentation (1 versus. all)	seg	19	30	180	0,143
Glass (7 versus. all)	gls7	10	29	185	0,136
Euthyroid (1 versus. all)	euth	24	238	1762	0,119
Satimage (4 versus. all)	sat	36	626	5809	0,097
Vowel (1 versus. all)	vow	10	90	900	0,091
Abalone (18 versus. 9)	a18-9	08	42	689	0,057
Yeast (9 versus. 1)	y9-1	08	20	463	0,041
Car (3 versus. all)	car	06	69	1659	0,04
Yeast (5 versus. all)	y5	08	51	1433	0,034
Abalone (19 versus. all)	a19	08	32	4145	0,008

moda foi escolhida como o melhor número de neurônios para a camada escondida da rede neural para aquele conjunto de dados. Em um passo subsequente, todo o conjunto de treinamento para cada um dos casos foi utilizado para o treinamento do modelo, e o conjunto de teste foi utilizado para avaliação do desempenho. A média dos 20 testes para cada um dos conjuntos de dados será apresentada no resultado final.

3.4.2 Testes não paramétricos

Durante muitos anos, os testes paramétricos foram referência na análise comparativa de classificadores. Contudo, Demšar (2006) recomenda o teste não paramétrico e robusto de Friedman (Friedman, 1937) para comparação estatística de mais de dois classificadores ao longo de múltiplos conjuntos de dados. O teste de Friedman faz um ranqueamento de L algoritmos ao longo de M base de dados em que o melhor desempenho recebe o valor 1, o segundo 2, e assim por diante; o valor médio desse ranqueamento pode ser visualizado na última linha das Tabelas 3.2 e 3.3. Assumindo-se a hipótese nula, segundo a qual todos os algoritmos são equivalentes, então o valor médio dos *rankings* deve ser equivalente, de forma que a estatística dada pela Equação (3.15) é distribuída de acordo com a distribuição-F, com $L-1$ e $(L-1)(M-1)$ graus de

liberdade.

$$F_F = \frac{(M-1)\chi_F^2}{M(L-1)-\chi_F^2} \tag{3.14}$$

$$\chi_F^2 = \frac{12M}{L(L+1)}\left(\sum_t R_l^2 - \frac{L(L+1)^2}{4}\right) \tag{3.15}$$

Quando a hipótese nula é rejeitada, Demšar (2006) sugere que outro teste seja feito a fim de quantificar a diferença entre os algoritmos. O modelo mais utilizado para analisar o comportamento de um algoritmo em relação aos demais é o teste Bonferroni-Dunn *post hoc* (Dunn, 1961), em que o classificador em foco é considerado estatisticamente diferente de forma significativa se as médias dos *rankings* apresentarem uma diferença crítica (CD, do inglês *critical difference*) mínima (Demšar, 2006), conforme a Equação 3.16.

$$CD = q_a\sqrt{\frac{L(L+1)}{6M}} \tag{3.16}$$

em que q_a é o valor crítico no nível de confiança de $1-\alpha$.

3.4.3 Resultados

O experimento foi conduzido para cada uma das métricas analisadas: G-mean, AUC e AGm. Os resultados estão apresentados nas Tabelas 3.2, 3.3 e 3.4.

Uma vez obtidos esses resultados, o teste de significância foi conduzido para analisar a abordagem apresentada. Primeiramente, o teste de Friedman (Friedman, 1937) foi conduzido com o valor do ranque médio de cada uma das métricas (presente na última linha de cada uma das tabelas de resultados 3.2, 3.3 e 3.4). Conforme a tabela apresentada em Sheskin (2007), o valor crítico mínimo quando $\alpha = 0.1$, $M = 16$ (número de base de dados) e $L = 6$ (número de algoritmos sob comparação), o qual garante que a hipótese nula seja rejeitada,(H_0) deve ser $F_F = 1.9256$. Para os ranques médios apresentados nas tabelas de resultados 3.2, 3.3 e 3.4, o valor da estatística F_F encontrada foi: 3.3206, 2.4818 e 6.8324, respectivamente.

Descartada a hipótese nula para todas as métricas analisadas, o teste de Bonferroni-Dunn *post hoc* foi aplicado para checar se o desempenho do algoritmo proposto é diferente dos outros (metodologia "um contra todos") no nível de confiança de $1-\alpha$. Para $\alpha = 0.1$, a mínima CD que deve ser alcançada para considerar dois algoritmos estatisticamente diferentes em relação ao desempenho deve ser de 1.7125. A diferença entre os ranques médios, de acordo com a metodologia de Bonferroni-Dunn *post hoc*,

Tabela 3.2: Valores médios de G-Mean

Base	Rprop	SMOTE	SMTTL	WWE	RAMOBoost	CSLFMLP
iono	85.64	87.49	88.65	85.55	**89.75**	85.56
pid	70.73	74.30	74.32	74.46	73.08	**74.57**
gmn	69.20	70.64	70.22	70.64	67.87	**71.08**
wpbc	64.44	66.76	64.37	63.25	**68.53**	67.34
veh	96.87	96.66	96.60	96.74	**97.72**	97.41
hrt	66.41	68.36	67.57	**73.39**	67.43	68.66
seg	99.57	99.44	99.68	99.51	**99.76**	99.37
gls7	91.00	89.92	90.14	**92.27**	90.45	91.61
euth	89.99	91.21	91.23	91.32	89.42	**91.73**
sat	74.60	77.40	77.77	80.21	76.27	**87.58**
vow	97.82	97.24	98.27	98.04	**99.30**	98.65
a18-9	74.17	84.37	**84.58**	76.64	74.61	83.77
y9-1	**74.27**	70.96	70.08	73.57	74.10	73.40
car	94.87	93.04	91.78	96.75	95.85	**99.19**
y5	51.36	77.93	78.52	66.46	63.94	**79.72**
a19	14.18	75.89	75.90	25.08	41.13	**76.96**
av. Rank	4.56	4.00	3.56	3.25	3.44	2.19

pode ser visualizada na Tabela 3.5. Os valores que superam o valor de CD mínima encontram-se em negrito.

Analisando-se os resultados presentes na tabela 3.5, considerando-se a métrica G-mean, pode-se inferir que a abordagem aqui apresentada demonstra ser significativamente melhor que o modelo Rprop e SMOTE, sendo ainda seu rank melhor do que os algoritmos SMTLL, WWE e RAMOBoost, mas não estatisticamente diferente destes últimos. Em relação à métrica AUC, o CSLFMLP demonstrou-se melhor que o SMOTE, além disso, com rank melhor e estatisticamente igual ao RAMOBoost, SMTTL e WWE. A última métrica avaliada pelo teste foi a AGm, em que o CSLFMLP mostrou-se significativamente melhor que o Rprop, além de estatisticamente igual ao RAMOBoost, SMTTL e WWE. Os resultados demonstram que a abordagem adotada para inclusão da informação *a priori* junto à função de erro de entropia cruzada com Rprop modificado, revelou-se uma boa alternativa a métodos nos quais essas métricas são utilizadas para avaliação de modelos de classes desbalanceadas.

Tabela 3.3: Valores médios da AUC

Base	Rprop	SMOTE	SMTTL	WWE	RAMOBoost	CSLFMLP
iono	89.90	91.86	92.03	93.02	**93.50**	91.58
pid	**82.88**	82.63	82.65	82.70	80.54	82.79
gmn	**78.49**	77.71	77.56	78.20	74.27	78.43
wpbc	73.30	70.82	71.19	72.54	**74.13**	74.02
veh	99.43	98.87	99.36	98.75	99.25	**99.76**
hrt	**81.58**	77.96	77.54	80.65	79.32	78.70
seg	99.98	99.97	**99.99**	99.91	99.90	99.98
gls7	**95.90**	95.61	95.28	95.09	93.81	95.08
euth	95.53	95.67	95.58	95.35	96.32	**96.81**
sat	92.50	91.98	92.27	92.85	93.18	**94.52**
vow	99.87	99.85	99.80	99.73	99.75	**99.87**
a18-9	**94.56**	93.76	93.69	94.18	89.16	93.33
y9-1	79.90	77.30	81.18	82.05	**84.09**	83.63
car	**99.71**	99.44	99.62	98.67	98.32	99.64
y5	85.65	85.76	85.78	**86.50**	85.79	86.25
a19	83.33	84.19	84.14	83.29	75.76	**85.13**
av. Rank	2.81	4.19	3.81	3.75	4.00	2.44

3.5 Conclusão do capítulo

Os testes significativos confirmam que o algoritmo CSLFMLP, aqui proposto, é uma boa abordagem quando se trata de dados não balanceados. Em geral, o algoritmo apresentou o melhor índice médio para todas as métricas usadas e provou ser significativamente ou ligeiramente melhor que os outros normalmente utilizados para esta tarefa em redes neurais.

O principal objetivo do trabalho apresentado neste capítulo foi desenvolver e apresentar uma nova metodologia de implementação simples, com base no *Resilient Backpropagation* padrão para a função de erro de entropia cruzada com informação *a priori*, que fosse estatisticamente melhor ou igual aos métodos tradicionais que tratam de dados não balanceados por meio de redes neurais *perceptron* de múltiplas camadas.

Além da robustez e eficiência da abordagem mostrada, sua implementação e compreensão simples tornam esse algoritmo uma ferramenta didática e prática no campo das redes neurais MLP. É importante enfatizar que o algoritmo não está restrito a problemas binários, sendo capaz de tratar dados de várias classes com diferentes níveis de desequilíbrio, usando a abordagem "um contra todos"em cada neurônio de saída.

A abordagem utilizada para o Rprop modificado pode ser entendida como uma nova

Tabela 3.4: Valores médios de AGm

Base	Rprop	SMOTE	SMTTL	WWE	RAMOBoost	CSLFMLP
iono	83.21	86.02	85.17	83.46	**86.32**	83.92
pid	65.95	75.39	**75.46**	73.72	71.14	74.29
gmn	63.99	**72.61**	71.75	68.63	67.09	72.37
wpbc	59.23	62.84	61.32	63.23	61.60	**64.77**
veh	96.58	96.52	95.56	96.69	97.08	**97.55**
hrt	64.71	65.40	54.93	**75.27**	65.10	64.29
seg	99.39	99.56	99.51	**99.59**	99.56	99.38
gls7	86.93	89.41	**89.72**	89.38	86.65	87.54
euth	87.38	90.30	90.28	89.55	86.76	**91.00**
sat	67.27	81.51	80.81	74.46	74.10	**87.10**
vow	97.33	96.49	97.69	97.59	**99.99**	98.60
a18-9	65.70	79.89	80.09	69.61	60.82	**81.16**
y9-1	66.03	68.34	**68.74**	65.28	67.28	65.74
car	92.66	93.65	93.82	98.05	93.55	**99.41**
y5	42.18	75.26	75.45	53.58	50.27	**77.28**
a19	8.30	74.94	75.30	10.48	40.81	**78.36**
av. Rank	5.31	2.75	3.00	3.75	4.00	2.44

Tabela 3.5: Teste Bonferroni-Dunn *post hoc* (CSLFMLP x todos)

CSLFMLP versus					
Métrica	Rprop	SMOTE	SMTTL	WWE	RAMOBoost
G-Mean	**2.3750**	**1.8125**	1.3750	1.0625	1.25
AUC	0.3750	**1.7500**	1.3750	1.3125	1.5625
AGm	**2.8750**	0.3125	0.5625	1.0625	1.5625

implementação de decaimento de peso, podendo ser implementada separadamente e utilizada como taxa de aprendizado dinâmico para diferentes tipos de algoritmos de otimização da função de custo em redes neurais. Tal abordagem será tratada em trabalhos futuros, cujo foco não seja classes desbalanceadas.

Capítulo 4

Desenvolvimento de funções de custo extraídas da matriz de confusão para redes neurais com classes desbalanceadas

Este capítulo tem como objetivo principal apresentar uma abordagem para criação de funções de custo, as quais podem ser construídas utilizando-se os termos da matriz de confusão. Dessa forma, é possível utilizar diferentes funções/métricas de problemas de balanceamento como funções de custo, de tal forma que a métrica a ser utilizada para avaliação do modelo seja a mesma utilizada pela rede neural como função de custo a ser otimizada.

Ao longo do capítulo, será apresentado como extrair métricas da matriz de confusão e usá-las como função de custo para treinar redes neurais. Testes empíricos, considerando a abordagem discutida, serão apresentados ao final do capítulo para problemas de classes desbalanceadas.

4.1 Introdução

A inteligência computacional vem sendo utilizada para a solução de um número cada vez maior de questões, relacionadas a diversos problemas que envolvem dados. Essa gama variada de questões e problemas a serem resolvidos cria uma diversificação muito grande do que é preciso em cada tipo de solução. Dessa forma, há uma necessidade crescente de métricas cada vez mais específicas para cada uma das questões a serem abordadas, principalmente quando se trabalha com problemas desbalanceados, para os quais métricas tradicionais, como acurácia e precisão, não são aconselháveis.

Ao lidar com dados não balanceados, a medição do desempenho de um algoritmo de aprendizagem requer métricas mais apropriadas (Weiss, 2004; Caruana and Niculescu-Mizil, 2004), como AUC (Hanley and McNeil, 1982), G-mean de Kubat (Kubat et al., 1997), AGm (Batuwita and Palade, 2012), F1-score (Pazzani and Billsus, 1997) e outras. Alguns exemplos podem ser vistos em problemas médicos, quando é necessário detectar alguma doença em um conjunto de dados desequilibrado; nesse caso, a AUC em conjunto com G-mean é uma métrica bastante utilizada. Por outro lado, na classificação de texto, o F1-score é o mais usado para medir o desempenho.

Um ponto comum dessas métricas é a capacidade de avaliar o desempenho do algoritmo de acordo com a capacidade de detectar ambas as classes, dando igual ou maior importância à classe minoritária. A fim de alcançar um bom desempenho nessas métricas, muitas abordagens emergiram nos últimos anos para lidar com dados não balanceados em redes neurais *perceptron* de múltiplas camadas (MLP). A maioria das abordagens atuais se enquadra em três categorias: pré-processamento de dados, *ensemble* e manipulação algorítmica. Algumas dessas técnicas estão resumidas em Michie et al. (1994), Chawla et al. (2002), Chawla et al. (2004b), Chawla et al. (2004a), He et al. (2009), Lan et al. (2010), Thai-Nghe et al. (2010) e Batista et al. (2004).

Este capítulo apresenta uma abordagem para construir funções sensíveis ao custo, baseadas na métrica utilizada para avaliar o modelo de aprendizagem. A metodologia apresentada foi utilizada para criar funções baseadas nas métricas: G-mean, F1-score e AGm. Uma função sensível ao custo para a métrica AUC também é estudada com base na função de cálculo aproximada (Hong et al., 2007).

A principal contribuição acadêmica deste capítulo consiste em apresentar como as métricas podem ser extraídas da matriz de confusão e como elas podem ser usadas para lidar com problemas de dados desbalanceados. Uma comparação das métricas como funções de custo é construída para avaliar seu desempenho e impacto.

4.2 Desenvolvimento da abordagem

Nesta seção descreve-se a ideia principal da abordagem aqui apresentada e como ela foi usada para criar funções sensíveis ao custo com base na métrica de avaliação de um modelo. A abordagem apresentada pode ir além das métricas demonstradas aqui, tornando possível criar qualquer função de custo sensível que possa ser representada em termos dos elementos da matriz de confusão.

Algumas das métricas mais utilizadas para medir o desempenho em MLP, ao lidar com dados não balanceados, podem ser representadas em termos da matriz de confusão (Tabela 4.1).

Tabela 4.1: Matriz de confusão

	Predicted Positive	**Predicted Negative**
Positive Label	True positive (TP)	False Negative (FN)
Negative Label	False Positive (FP)	True negative (TN)

A matriz de confusão é aqui apresentada em inglês para facilitar a visualização e compreensão da abordagem apresentada.

Considerando um problema binário em que $y \in \mathcal{Y} = \{0, 1\}$ e $\hat{y} \in \mathcal{R} = [0, 1]$, é possível aproximar a matriz de confusão em termos da saída da rede MLP ($\hat{\mathbf{y}}$) e a categoria desejada ($\mathbf{y}$), como pode ser visualizado na Tabela 4.2.

Tabela 4.2: Termos da matriz de confusão $\hat{\mathbf{y}}$ e $\mathbf{y}$

	Predicted Positive	**Predicted Negative**
Positive Label	$\sum_{i=1}^{n} y_i \cdot \hat{y}_i$	$\sum_{i=1}^{n} y_i \cdot (1 - \hat{y}_i)$
Negative Label	$\sum_{i=1}^{n} (1 - y_i) \cdot \hat{y}_i$	$\sum_{i=1}^{n} (1 - y_i) \cdot (1 - \hat{y}_i)$

$y = 1$: Representa o rótulo da classe minoritária.

$y = 0$: Representa o rótulo da classe majoritária.

$\hat{y}$: Saída da rede neural MLP.

Com base nisso, decidiu-se converter a representação das métricas AUC (função de cálculo aproximado) (Hong et al., 2007), Kubat's G-mean , F1-score e AGm extraídas da matriz de confusão presente na Tabela 4.1 em termos da matriz de confusão aproximada (dado que os valores não foram arredondados) presente na Tabela 4.2. Essas transformações estão presentes na Tabela 4.3.

Um ponto comum das métricas mencionadas acima é que elas variam de 0 a 1, tais que 1 representa o melhor valor possível, portanto, o objetivo é encontrar os parâmetros que maximizam essas métricas. No entanto, a maioria das funções de custo nas redes neurais é representada como função de perda. Além disso, essas métricas são deriváveis, mas suas derivadas não são fáceis de calcular. Por isso, decidiu-se construir as funções sensíveis ao custo em termos da função de perda, calculando-se o valor negativo do logaritmo dessas métricas. Ao fazê-lo, a derivada pode ser extraída mais facilmente e o objetivo torna-se minimizar a função de perda criada.

Tabela 4.3: Representação em termos da matriz de confusão

Metrics	***Representation***	
	Confusion Matrix	**Approx. Confusion Matrix**
AUC	$\frac{1}{2} \cdot (1 + TPr - FPr)$	$\frac{1}{2} \cdot \left(1 + \frac{\sum_{i=1}^{n} y_i \cdot \hat{y}_i}{N_p} - \frac{\sum_{i=1}^{n} (1-y_i) \cdot \hat{y}_i}{N_n}\right)$
G-mean	$\sqrt{TPr \cdot TNr}$	$\sqrt{\frac{\sum_{i=1}^{n} y_i \cdot \hat{y}_i}{N_p} \cdot \frac{\sum_{i=1}^{n} (1-y_i) \cdot (1-\hat{y}_i)}{N_n}}$
F1-score	$\frac{2TP}{2TP+FP+FN}$	$\frac{2\sum_{i=1}^{n} y_i \cdot \hat{y}_i}{2\sum_{i=1}^{n} y_i \cdot \hat{y}_i + \sum_{i=1}^{n} (1-y_i) \cdot \hat{y}_i + \sum_{i=1}^{n} y_i \cdot (1-\hat{y}_i)}$
AGm	$\frac{(\sqrt{TPr \cdot TNr}) + (TNr \cdot P_n)}{1+P_n}$	$\frac{\left(\sqrt{\frac{\sum_{i=1}^{n} y_i \cdot \hat{y}_i}{N_p} \cdot \frac{\sum_{i=1}^{n} (1-y_i) \cdot (1-\hat{y}_i)}{N_n}}\right) + \left(\frac{\sum_{i=1}^{n} (1-y_i) \cdot (1-\hat{y}_i)}{N_n} \cdot P_n\right)}{1+P_n}$

N_p : Número de amostras positivas ($\sum_{i=1}^{n} y_i$)

N_n : Número de amostras negativas ($\sum_{i=1}^{n} (1 - y_i)$)

P_n : Proporção de amostras negativas ($\frac{N_n}{N_p+N_n}$)

TPr : True Positive rate (TP/N_p)

TNr : True Negative rate (TN/N_n)

Ao introduzir uma nova função de custo, a única coisa que precisa ser alterada durante a fase do *backpropagation* é o termo de erro (*error term*) ($delta^0$ - Equação 4.1). Feita essa alteração, todas as demais etapas fluem naturalmente. As equações 4.2, 4.3, 4.4 e 4.5 mostram o termo de erro, dado o logaritmo negativo das equações apresentadas na tabela 4.3 para AUC, G-mean, F1-score e AGm, respectivamente.

$$\delta^0 = \frac{\partial J}{\partial f(h)} \frac{\partial f(h)}{h} \tag{4.1}$$

em que:

J : Função de custo

$f(h)$: Função de ativação

h : Saída de um neurônio

$$\delta^0 = -\frac{\frac{y_i}{N_p} - \frac{1-y_i}{N_n}}{1 + \frac{\sum_{i=1}^{n} y_i \cdot \hat{y}_i}{N_p} - \frac{\sum_{i=1}^{n}(1-y_i)\cdot\hat{y}_i}{N_n}} \tag{4.2}$$

$$\delta^0 = -\frac{1}{2}\left(\frac{y_i}{\sum_{i=1}^{n} y_i \cdot \hat{y}_i} + \frac{y_i - 1}{\sum_{i=1}^{n}(1-y_i)\cdot(1-\hat{y}_i)}\right) \tag{4.3}$$

$$\delta^0 = -\frac{y_i}{\sum_{i=1}^{n} y_i \cdot \hat{y}_i} + \frac{1-y_i}{\sum_{i=1}^{n}(1-y_i)\cdot\hat{y}_i + \sum_{i=1}^{n} y_i \cdot \hat{y}_i + \sum_{i=1}^{n} y_i \cdot (1-\hat{y}_i)} \tag{4.4}$$

$$\delta^0 = \frac{\frac{(y_i-1)\cdot P_n}{N_n} + \frac{\frac{y_i}{N_p}\cdot\frac{\sum_{i=1}^{n}(1-y_i)\cdot(1-\hat{y}_i)}{N_n} + \frac{(y_i-1)}{N_n}\cdot\frac{\sum_{i=1}^{n} y_i\cdot\hat{y}_i}{N_p}}{2\cdot\sqrt{\frac{\sum_{i=1}^{n} y_i\cdot\hat{y}_i}{N_p}\cdot\frac{\sum_{i=1}^{n}(1-y_i)\cdot(1-\hat{y}_i)}{N_n}}}}{\sqrt{\frac{\sum_{i=1}^{n} y_i\cdot\hat{y}_i}{N_p}\cdot\frac{\sum_{i=1}^{n}(1-y_i)\cdot(1-\hat{y}_i)}{N_n}} + P_n\cdot\frac{\sum_{i=1}^{n}(1-y_i)\cdot(1-\hat{y}_i)}{N_n}} \tag{4.5}$$

Na próxima seção, o comportamento das funções de custo apresentadas será explorado, assim como os resultados empíricos iniciais.

4.2.1 Função de custo AUC

A curva *Receiver Operating Characteristic* (ROC) tornou-se popular no início da década de 1970, quando era muito utilizada pela comunidade de radiologia (Lusted, 1971; Goodenough et al., 1974; Swets, 1979). Todavia, apenas em 1982, com a publicação do artigo de Hanley and McNeil (1982), o conceito da área sob a curva ROC (AUC) foi introduzido como uma métrica estatística e ganhou importância.

A partir dos anos 2000, com a popularização dos algoritmos de aprendizado de máquinas, a AUC difundiu-se ainda mais como uma métrica de análise do desempenho de problemas de classes desbalanceadas (Fawcett, 2006; He et al., 2009). Tal difusão deveu-se, em grande parte, ao fato de que a acurácia por si só não é uma métrica capaz de avaliar o desempenho em problemas de dados desbalanceados, e, portanto, à necessidade de uma métrica que pudesse trazer uma melhor avaliação do classificador para esse tipo de problema. Uma das características importantes para a popularização da AUC foi o fato de esta ser uma métrica indiferente em relação ao balanceamento das classes.

Em um grande número de problemas de classes desbalanceadas, principalmente na área médica, o objetivo principal é tentar encontrar um classificador que seja capaz de rotular corretamente os eventos raros, classe minoritária, sem comprometer o desempenho da classe majoritária. Para atingir esse objetivo, a AUC torna-se extremamente útil por possuir uma propriedade estatística muito importante: a AUC de um classificador é equivalente à probabilidade de este classificar uma instância positiva escolhida

aleatoriamente mais do que uma instância negativa escolhida aleatoriamente (Fawcett, 2006).

Dada a importância da AUC em uma gama de problemas de classificação, alguns algoritmos surgiram com o objetivo de treinar o modelo de aprendizado de máquinas tendo como função objetiva a estatística de Wilconox-Mann-Whitney, a qual é equivalente à AUC (Yan et al., 2003; Castro and Braga, 2008; Rakotomamonjy, 2004). Tendo como exemplo os resultados atingidos nesses trabalhos, o objetivo deste capítulo é apresentar uma abordagem mais direta e simplificada do uso da função de custo baseada na AUC (como apresentada nesta seção) em redes neurais MLP, e analisar o comportamento da rede em comparação com a função de custo baseada no erro médio quadrático.

Utilizando-se a abordagem presente na tabela 4.3 e na equação 4.2, foi realizado um teste inicial com as bases de dados do repositório da UCI, Satimage e Abalone. Para a base Satimage, considerou-se a classe 4 ($y = 1$) contra todas ($y = 0$), e na base Abalone, a classe 19 ($y = 1$) contra as demais ($y = 0$).

Para verificar se a metodologia teria algum efeito prático para um teste futuro mais apurado, utilizou-se uma rede com apenas um neurônio na camada intermediária e um neurônio na saída com funções de ativação sigmoidal. Os testes foram realizados utilizando-se a função de custo proposta contra a função de entropia cruzada, com a abordagem de atualização dos pesos do Rprop modificado, apresentada no capítulo 3. Em ambos os testes utilizaram-se dois terços para treinamento e um terço para validação. Durante o teste, variou-se o número de amostras da classe majoritária, de forma que este número fosse 1, 2, 5 e 9 vezes o número de amostras da classe minoritária para a base de dados Satimage; enquanto para a base de dados Abalone a variação foi de 1, 10, 50 e 100 vezes o número de amostras da classe de menor representatividade.

Nas Figuras 4.1 e 4.2, é possível observar os valores da AUC durante as primeiras 5 mil iterações para as bases de dados Satimage e Abalone, respectivamente.

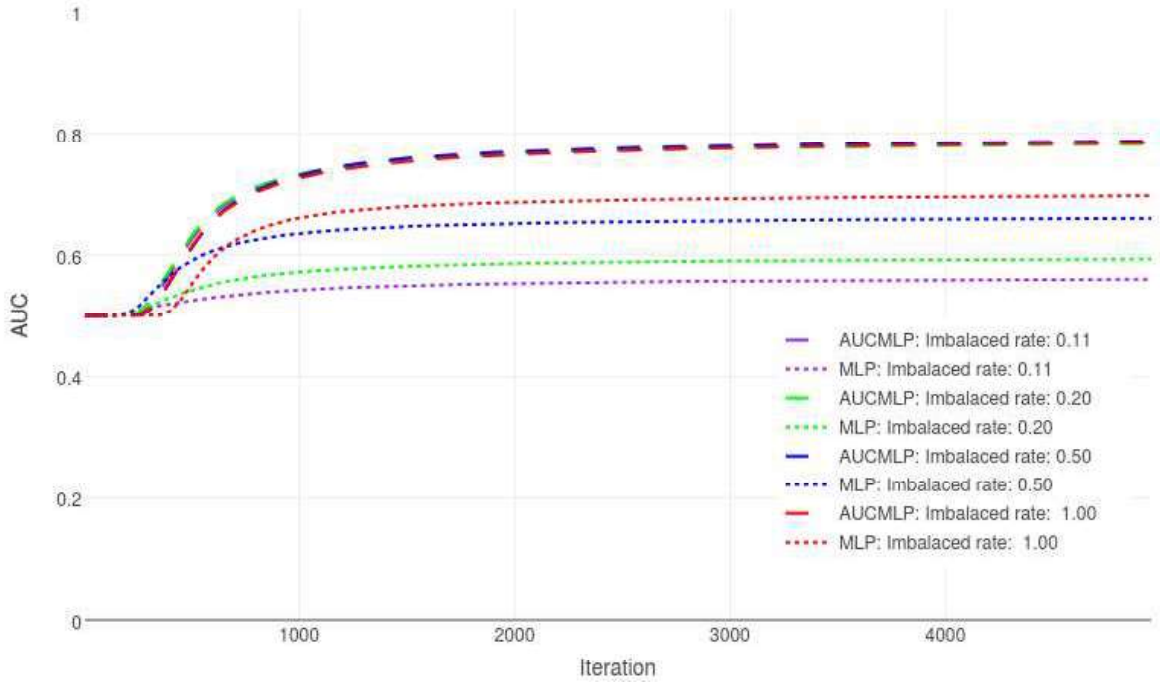

Figura 4.1: AUC durante as primeiras 5 mil iterações na base de dados Satimage

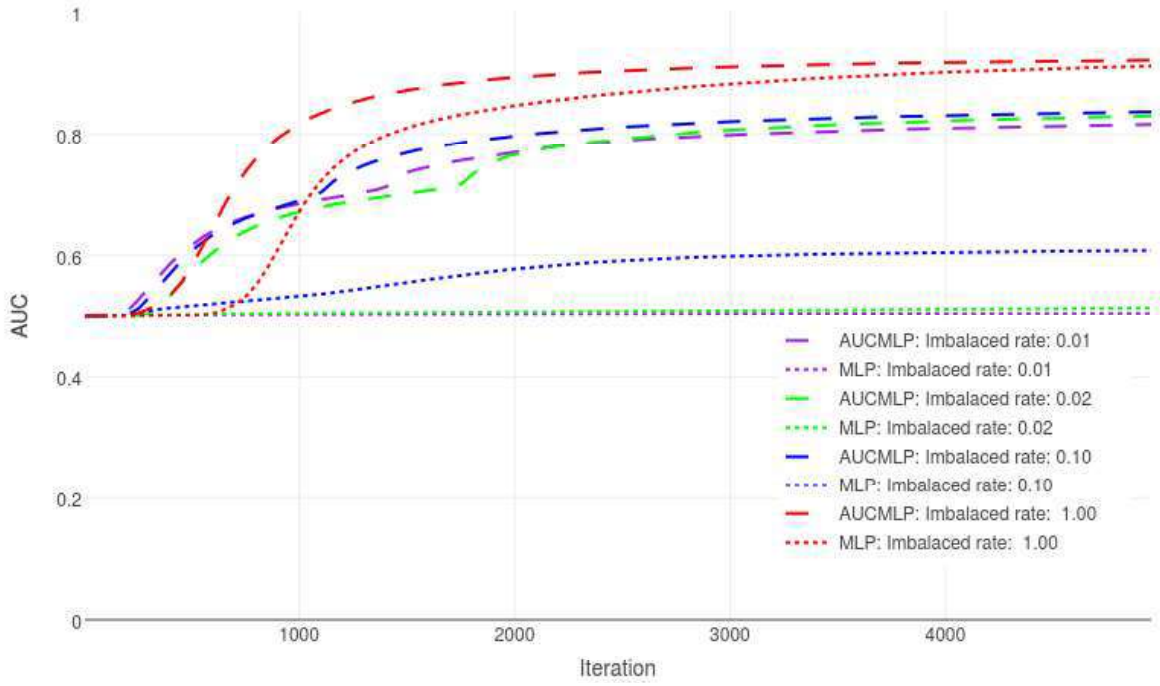

Figura 4.2: AUC durante as primeiras 5 mil iterações na base de dados Abalone

Analisando-se as Figuras 4.1 e 4.2, observa-se que, em ambos os casos, independentemente da taxa de desbalanceamento, a abordagem que utiliza a função de custo baseada na AUC apresenta valores superiores para a AUC durante as primeiras mil

iterações. Esse teste inicial mostra que a abordagem destacada pode ser benéfica, principalmente em casos nos quais o desbalanceamento é muito grande. Dado que o teste foi feito considerando apenas um neurônio na camada intermediária, é preciso que mais testes sejam feitos para se chegar a uma conclusão mais exata. Tais testes serão apresentados na seção de metodologia e resultados deste capítulo.

4.2.2 Função de custo G-mean

Em uma grande gama de problemas de classificação binária, há um desafio de conseguir encontrar um equilíbrio para o classificador. Porém, como identificar e atingir esse ponto de equilíbrio? Em 1997, Kubat, Holte e Matwin estavam desenvolvendo um projeto para detecção de vazamento de óleo por meio de imagens transmitidas por satélites, quando se depararam com a necessidade de encontrar um classificador que fosse capaz de atingir um equilíbrio entre as imagens rotuladas com e sem vazamento, conforme o desejo de seus clientes (Kubat et al., 1998). Durante o projeto, eles testaram algumas métricas próprias de problemas de classes desbalanceadas, como a AUC e F-measure. Entretanto, essas métricas não conseguiam representar bem o objetivo. Diante disso, eles tiveram a ideia de utilizar a média geométrica entre a acurácia da classe majoritária e da minoritária, calculadas separadamente. Em Kubat et al. (1997) é apresentado o processo da criação da métrica, extraída da matriz de confusão 4.1 e apresentada em 4.3.

Após sua apresentação, a métrica G-mean passou a ser muito utilizada em diversos tipos de problemas, desde acadêmicos até detecção de faltas, cujo objetivo era encontrar um ponto de equilíbrio para o classificador (Phung et al., 2009; Xu and Chow, 2006; Hong et al., 2007; Xu et al., 2007; Antanasijević et al., 2016; Kim et al., 2016). Dada a importância da métrica e da relação que ela apresenta, resolveu-se fazer um experimento semelhante ao realizado na subseção anterior com a AUC. Utilizaram-se no experimento as bases de dados Satimage e Abalone, sob as mesmas condições de base e configurações de rede. No teste, foi utilizada como função de custo a G-mean apresentada na tabela 4.3 e na equação 4.3, bem como a função de custo de entropia cruzada, com a abordagem da atualização de pesos do Rprop modificado, explicada no segundo e terceiro capítulo deste trabalho.

Nas Figuras 4.3 e 4.4 é exibido o gráfico do número de iterações pela métrica G-mean para as bases de dados Satimage e Abalone, respectivamente.

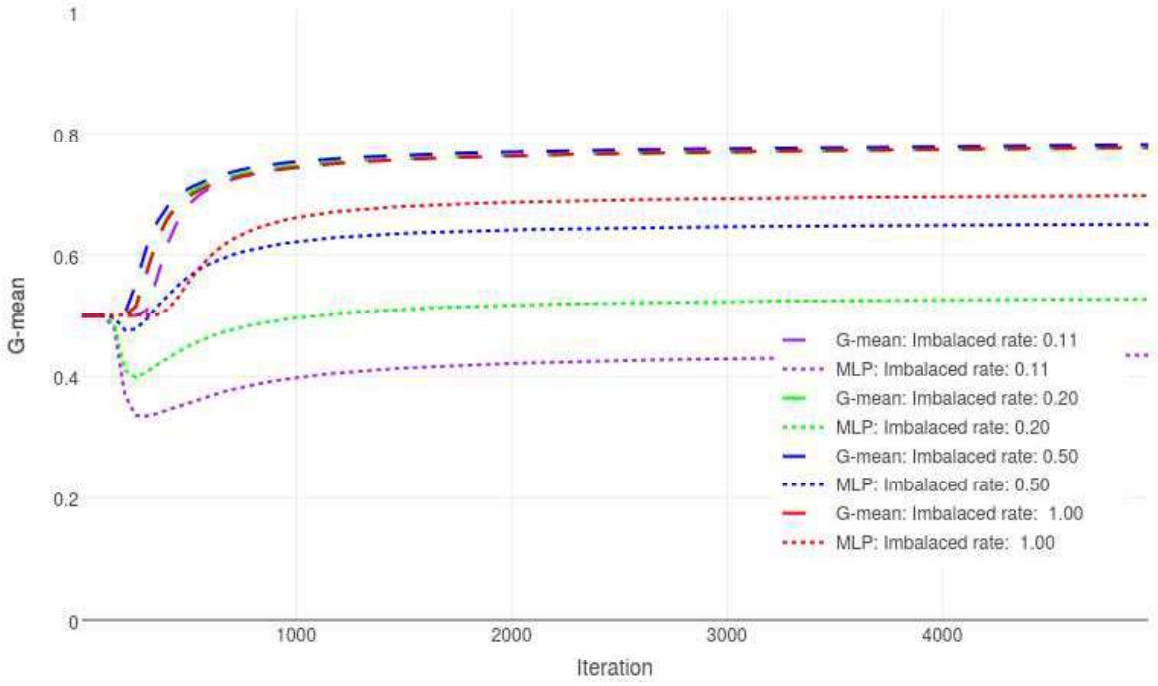

Figura 4.3: G-mean durante as primeiras 5 mil iterações na base de dados Satimage

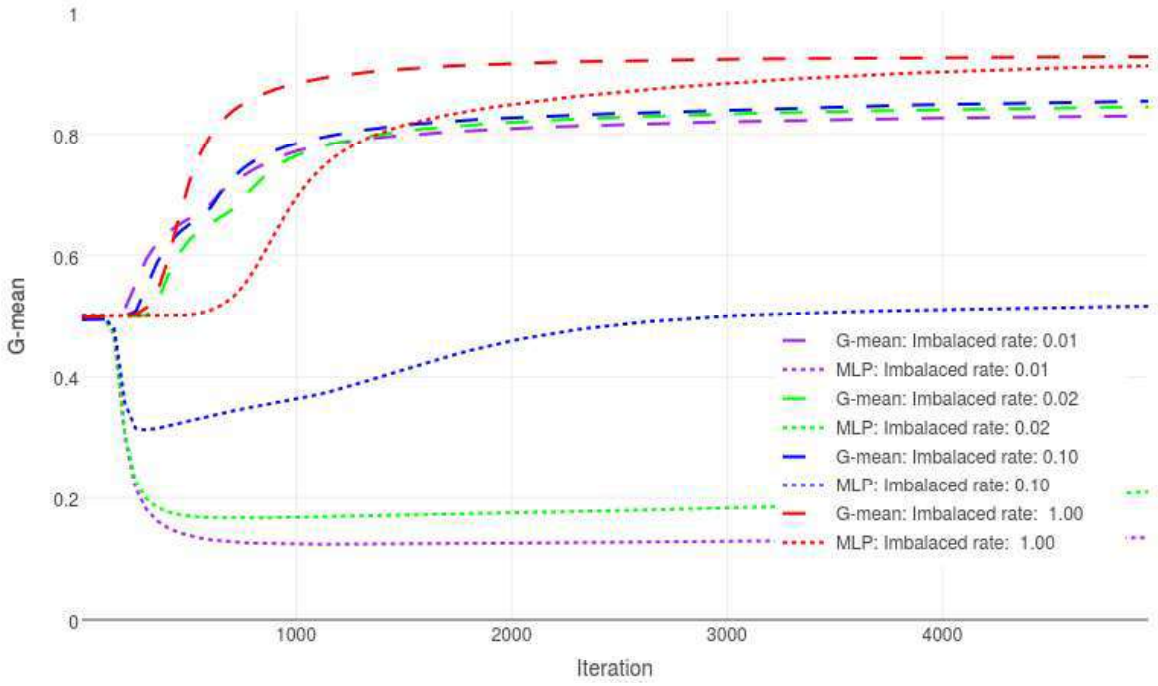

Figura 4.4: G-mean durante as primeiras 5 mil iterações na base de dados Abalone

De forma análoga ao apresentado na subseção anterior, os gráficos 4.3 e 4.4 demonstram que existe um potencial benefício em utilizar a abordagem descrita na tabela 4.3 e na equação 4.3. Para tentar comprovar o benefício desse tipo de abordagem, principal-

mente nas situações em que há um maior desbalanceamento, é preciso um estudo mais profundo, que será apresentado na seção de metodologia e resultados deste capítulo.

4.2.3 Função de custo F1-score

Em determinados tipos de problemas, principalmente na área de recuperação da informação (Ex.: classificação de documentos da internet), realizam-se estudos nos quais o classificador age sobre uma determinada base de dados cujo número de amostras não é totalmente conhecido. Nesse tipo de problema, define-se uma amostra como positiva quando esta possui alguns atributos de interesse no caso estudado. Um exemplo clássico encontra-se na busca por páginas e documentos, em que aqueles que possuem características representativas do termo de pesquisa são classificados como amostras positivas, e todas as outras são taxadas de amostras negativas (Hripcsak and Rothschild, 2005).

Nos tipos de problema em que se deseja quantificar a abordagem do classificador sem levar em consideração os casos negativos, duas métricas muito utilizadas são o *precision* e o *recall*. Em 1979, quando trabalhando em sistemas de recuperação de informação, Van Rijsbergen introduziu uma nova métrica que combinava essas duas, a qual ficou conhecida como F-score (ou F-measure) (Van Rijsbergen, 1979). A métrica F-score é representada de acordo com a Equação 4.6, em que o termo β consiste em um parâmetro que controla um equilíbrio entre o *precision* e o *recall*. Quando $\beta > 1$, a fórmula tende a ser mais orientada para o *recall*, e se $\beta < 1$, tende a favorecer mais o *precision* (Sasaki et al., 2007). Todavia, na maioria dos experimentos, não há nenhum motivo particular para favorecer um ou outro, então, o valor mais utilizado é $\beta = 1$ (Hripcsak and Rothschild, 2005). Tornando-se a média harmônica entre os dois termos, neste caso, a métrica assume o nome de F1-score ou F1-measure (Equação 4.7).

$$\textit{F-score} = \frac{(1 + \beta^2) \cdot \text{recall} \cdot \text{precision}}{\beta^2 \cdot \text{precision} + \text{recall}} \tag{4.6}$$

$$\textit{F1-score} = \frac{2 \cdot \text{recall} \cdot \text{precision}}{\text{precision} + \text{recall}} \tag{4.7}$$

Abordagens com o objetivo de maximizar o desempenho de um classificador para a métrica F-score podem ser vistas em trabalhos tais como: Joachims (2005) maximiza a métrica para problemas de máquinas de vetores de suporte (SVM, do inglês *support vector machine*); Jansche (2005) realiza uma aproximação da métrica F-score, utilizando-se de modelos de regressão logística; Nan et al. (2012) utiliza-se de um algorítimo denominado *Empirical Utility Maximization* (EUM) para maximizar o valor

da métrica; e Dembczynski et al. (2011) também cria um novo algoritmo, denominado *General F-measure Maximizer* (GFM), com o mesmo objetivo.

Por se tratar de uma métrica muito popular em problemas de classes desbalanceadas, submetida a diversas tentativas de maximização, decidiu-se utilizar a abordagem presente na Tabela 4.3 e na Equação 4.4 para explorar o comportamento da rede neural quando a métrica F1-score é utilizada como função de custo. Nesse caso, tem-se como objetivo escolher o melhor modelo de classificador com base na métrica F1-score. Para tal, a abordagem inicial para testar o valor dessa métrica foi igual à realizada nas subseções anteriores, considerando as mesmas bases de dados (Satimage e Abalone), condições de rede e de testes (uma camada intermediária com um neurônio).

Os resultados desse primeiro teste podem ser encontrados nas Figuras 4.5 e 4.6, em que é possível visualizar o valor da métrica F1-score nas primeiras 5 mil iterações para as bases de dados Satimage e Abalone, respectivamente.

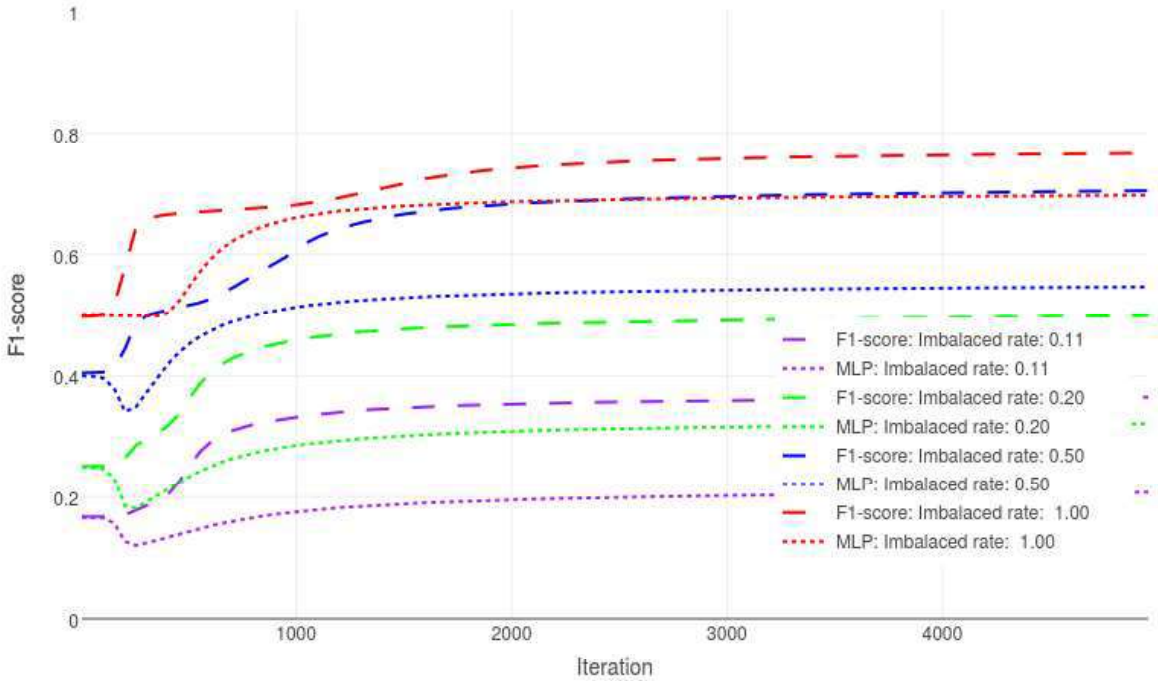

Figura 4.5: F1-score durante as primeiras 5 mil iterações na base de dados Satimage

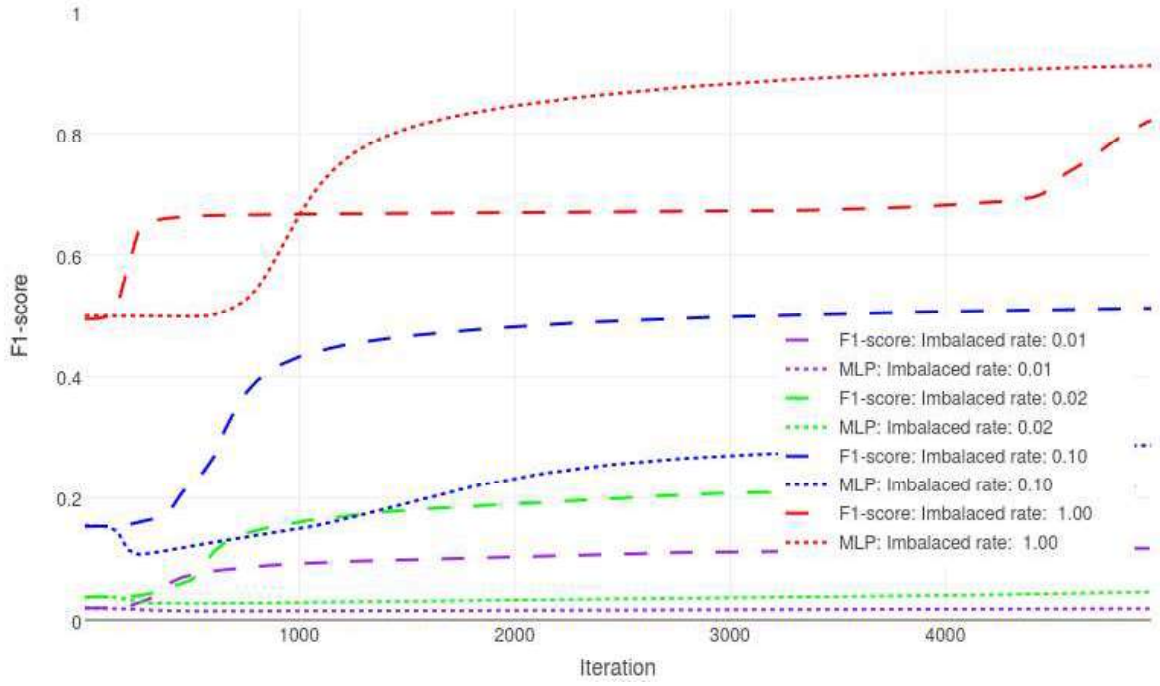

Figura 4.6: F1-score durante as primeiras 5 mil iterações na base de dados Abalone

Analisando-se os gráficos apresentados nesta análise inicial, não é possível concluir com exatidão se a abordagem utilizando F1-score como função de custo apresenta resultado melhor do que quando utilizada a função de entropia cruzada. Na próxima seção será apresentado um teste mais robusto e com um maior número de bases.

4.2.4 Função de custo AGm

Anteriormente neste capítulo, foi descrita a importância da métrica G-mean, a qual tenta encontrar um equilíbrio para o classificador, de forma que ele possa classificar com razoável precisão as classes majoritária e minoritária. Também foi apresentada, na subseção anterior, a importância da métrica F1-score na análise da capacidade do modelo em prever de forma correta as amostras da classe positiva. Todavia, existem situações (principalmente na área de saúde) em que se busca um classificador que seja capaz de melhorar a classificação das amostras positivas o máximo possível, mantendo a redução na acurácia da classe majoritária ao mínimo, pois a predição de falsos positivos pode ser ruim para a atividade fim do classificador. Nesse tipo de situação, não existe uma métrica única que consiga atender a essa necessidade. É preciso, portanto, analisar mais de uma métrica em conjunto para concluir qual o melhor classificador que a atenda. Diante desse impasse e da dificuldade em encontrar uma única métrica que fosse capaz de traduzir esta necessidade, em 2009, quando trabalhando em um problema de bioinformática, Rukshan Batuwita e Vasile Palade criaram uma nova métrica chamada

Adjusted G-mean (AGm) (Batuwita and Palade, 2009). A proposta apresentada por eles consegue mensurar um bom algoritmo, o qual tem como característica uma boa sensibilidade (SE) sem comprometer a especificidade (SP), garantindo que haja um número pequeno de falsos positivos e uma boa classificação das amostras positivas.

Um dos principais pontos dos autores na criação da métrica AGm é que, quando utilizadas técnicas de balanceamento de classes, acaba-se melhorando a classificação das amostras positivas, em contraposição a um custo muito alto na diminuição da acurácia na predição das classes negativas. Isso acontece principalmente porque aumentar a predição das amostras majoritárias e minoritárias são objetivos contraditórios (Batuwita and Palade, 2009).

A fim de exemplificar como a métrica AGm consegue atingir a ideia proposta de forma melhor do que o G-mean e F1-score individualmente, em Batuwita and Palade (2012) os autores da métrica apresentam dois casos de estudo.

Caso A

- Modelo 1
  Sensitivity (SE) $= 92,00\%$, *Specificity* (SP) $= 98,00\%$, G-mean $= = 94,95\%$
  AGm $= 96,44\%$

- Modelo 2
  Sensitivity (SE) $= 94,95\%$, *Specificity* (SP) $= 94,95\%$, G-mean $= = 94,95\%$
  AGm $= 94,95\%$

- Modelo 3
  Sensitivity (SE) $= 98,00\%$, *Specificity* (SP) $= 92,00\%$, G-mean $= = 94,95\%$
  AGm $= 93,51\%$

Caso B

- Modelo 1
  Sensitivity (SE) $= 70,00\%$, *Specificity* (SP) $= 99,00\%$, F1-score $= = 74,47\%$
  F1-score $= 90,92\%$

- Modelo 2
  Sensitivity (SE) $= 90,00\%$, *Specificity* (SP) $= 97,00\%$, F1-score $= = 73,16\%$
  F1-score $= 95,17\%$

Observa-se que, no Caso A, se a métrica utilizada para escolher o modelo fosse o G-mean, o classificador escolhido poderia gerar um resultado sub-ótimo dentre as opções

apresentadas, para o tipo de problema descrito, que tende a priorizar uma boa sensibilidade com comprometimento mínimo da especificidade. Todavia, quando a métrica AGm é empregada para análise da melhor alternativa, o modelo 1 é selecionado em detrimento dos outros, o que satisfaz o requisito proposto para esse tipo de problema.

No caso B, considerando a métrica F1-score como critério de seleção do modelo, seria possível chegar a uma escolha equivocada, em que o modelo 1 seria escolhido em detrimento do segundo. Entretanto, o AGm consegue fazer a seleção correta do modelo 2, que atende melhor ao requisito proposto.

Comprovada a característica especial da métrica e sua eficiência em lidar com esse tipo de problema específico dentro da área de desbalanceamento de classes, este trabalho propõe a utilização dessa métrica como função de custo, utilizando a metodologia apresentada na Tabela 4.3 e na Equação 4.5. Um teste análogo ao apresentado nas subseções anteriores foi realizado a fim de provar um possível valor da utilização dessa métrica como função de custo comparada à entropia cruzada, quando se deseja utilizar a AGm como métrica de seleção de modelos.

Nas Figuras 4.7 e 4.8 é exibido o gráfico do número de iterações pela métrica AGm para as bases de dados Satimage e Abalone, respectivamente.

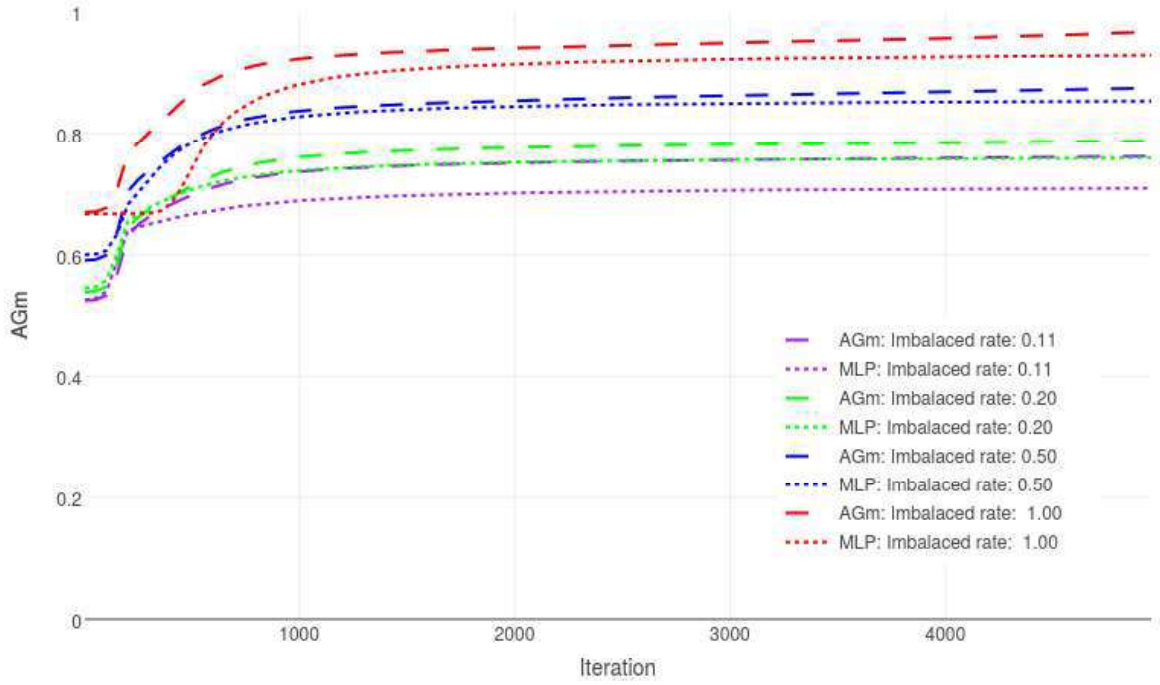

Figura 4.7: AGm durante as primeiras 5 mil iterações na base de dados Satimage

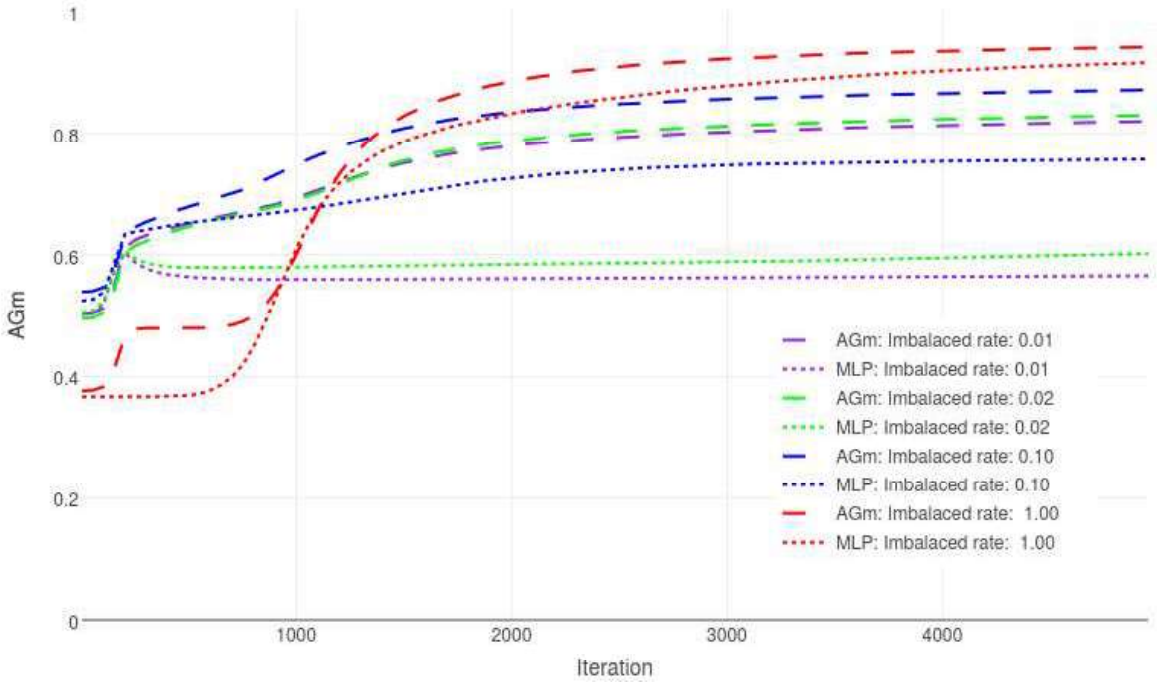

Figura 4.8: AGm durante as primeiras 5 mil iterações na base de dados Abalone

Observando os gráficos comparativos para a função de custo utilizando AGm contra a entropia cruzada, fica evidente que a abordagem que utiliza a função de custo AGm pode ser benéfica durante o treinamento quando tem como objetivo a utilização da mesma métrica como critério para seleção do modelo.

Além dos gráficos apresentados nas Figuras 4.7 e 4.8, foram também gerados gráficos da sensibilidade (SE = TPr) e especificidade (SP = TNr) ao longo das primeiras 5 mil iterações, tanto para a base de dados Satimage quanto para a Abalone. Os resultados estão presentes nas Figuras 4.9, 4.10, 4.11 e 4.12, nas quais é possível observar a relação de *trade-off* entre a taxa de verdadeiros positivos (TPr) e verdadeiros negativos (TNr).

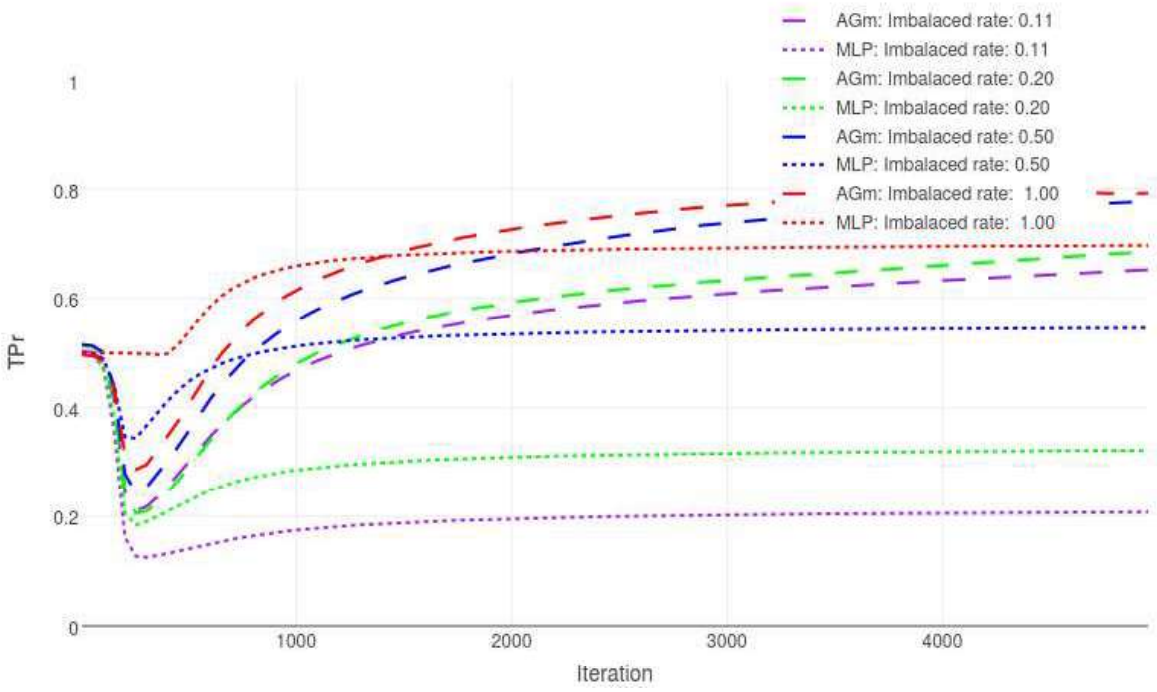

Figura 4.9: TPr durante as primeiras 5 mil iterações na base de dados Satimage

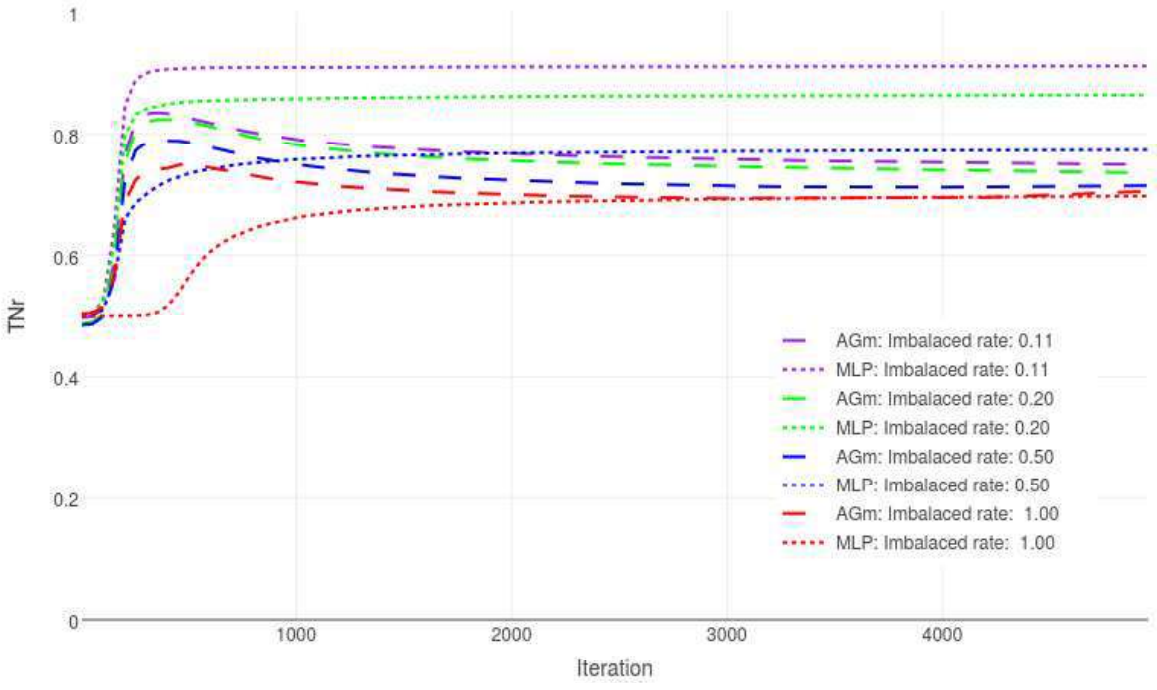

Figura 4.10: TNr durante as primeiras 5 mil iterações na base de dados Satimage

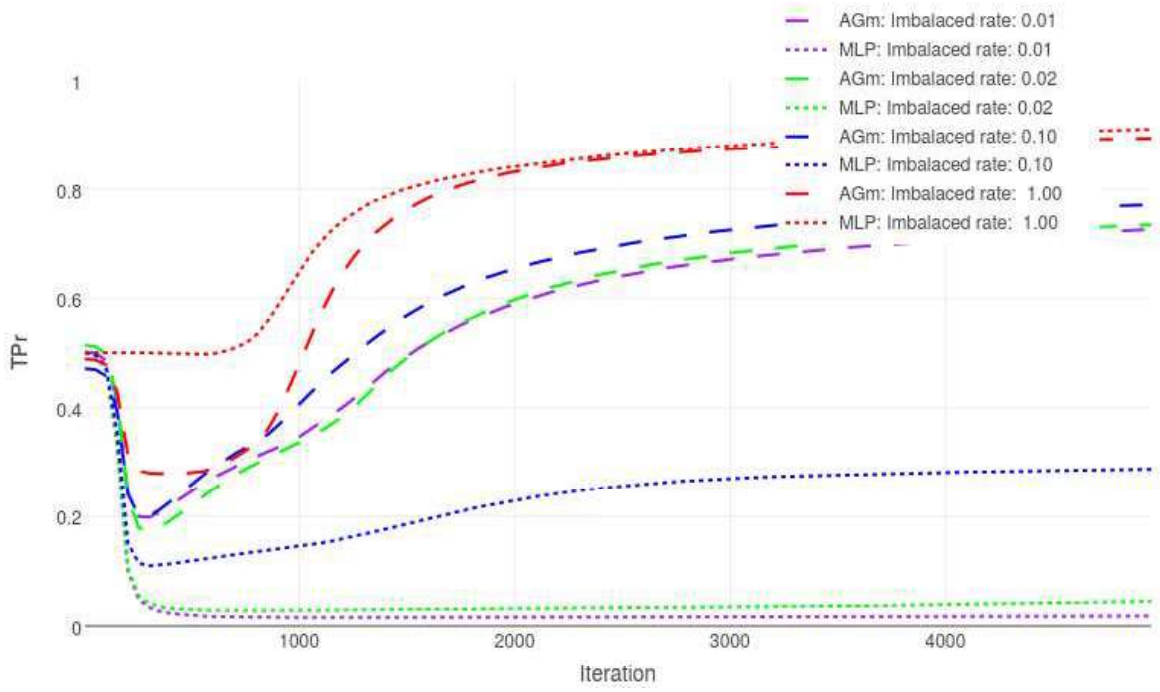

Figura 4.11: TPr durante as primeiras 5 mil iterações na base de dados Abalone

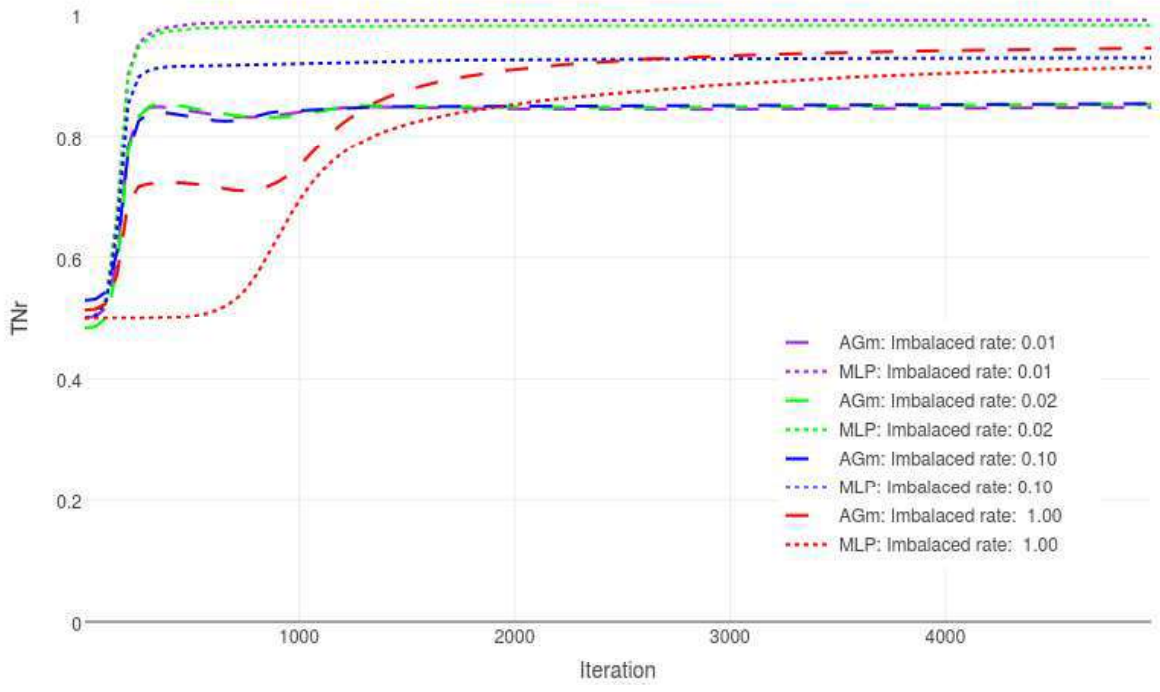

Figura 4.12: TNr durante as primeiras 5 mil iterações na base de dados Abalone

4.3 Metodologia e resultados

Neste capítulo apresenta-se a metodologia utilizada para gerar os resultados dos testes realizados para comparar as abordagens descritas na seção anterior. Na comparação serão utilizadas 16 bases de dados distintas, extraídas do repositório da UCI (Frank, 2010), com diferentes níveis de desbalanceamento e tratadas como apresentado em Castro and Braga (2013). As bases de dados utilizadas serão as mesmas do teste da Seção 1 deste trabalho. Na Tabela 4.4 são exibidas novamente as características das bases de dados utilizadas para os testes.

Para realização do teste, foi utilizada uma rede neural com apenas uma camada intermediária. A escolha de apenas uma camada deve-se à comprovação em estudos anteriores, como Castro and Braga (2013), de que para estas bases de dados uma camada intermediária era suficiente para obtenção de bons resultados. Além disso, a introdução de um maior número de camadas poderia dificultar a comparação, que é o objetivo do presente trabalho. As funções de ativação da camada intermediária utilizadas foram sigmoidais, e a saída foi constituída apenas de um neurônio sigmoidal.

Os testes foram realizados comparando cada uma das abordagens de funções de custo específicas apresentadas, contra o MLP, SMOTE (Chawla et al., 2002), SMTTL (Tomek, 1976), WWE (Barandela et al., 2004) e RAMOBoost (Provost and Fawcett, 2001). Vale ressaltar que MLP, SMOTE, SMTTL, WWE e RAMOBoost utilizaram funções de custo de entropia cruzada.

Para simplificar o número de parâmetros necessários para tunar o modelo e estabelecer uma melhor estabilidade ao algoritmo, o método de otimização escolhido para todos os modelos foi o Rprop modificado, apresentado no Capítulo 3 deste trabalho (seção 3.3).

O restante desta seção está dividido em quatro partes: metodologia dos experimentos, resultados, testes estatísticos e discussão.

4.3.1 Metodologia do experimento

De forma análoga ao apresentado no Capítulo 3 deste trabalho, um estudo empírico com 16 bases distintas (Tabela 4.4) foi realizado para cada uma das abordagens apresentadas nas subseções da seção anterior. Isto é, para cada uma das funções de custo apresentadas (AUC, G-mean, F1-score e AGm), baseadas na metodologia proposta anteriormente, foi realizado um estudo empírico distinto. Para cada um dos estudos, uma das métricas era considerada como critério de sucesso para escolha do melhor modelo.

Diferentemente do capítulo anterior, em que para cada base de dados foram criados 20 conjuntos de dados distintos, neste capítulo foram criados apenas dez conjuntos,

Tabela 4.4: Características das bases de dados

Base de dados	Alias	No. de *features*	n_1	n_2	$n_1/(n_1+n_2)$
Ionosphere	iono	34	126	225	0,359
Pima Indians Diabetes	pid	08	268	500	0,349
German Credit	gmn	24	300	700	0,3
WP Breast Cancer	wpbc	33	47	151	0,237
Vehicle (4 versus. all)	veh	18	199	647	0,235
SPECTF Heart	hrt	44	55	212	0,206
Segmentation (1 versus. all)	seg	19	30	180	0,143
Glass (7 versus. all)	gls7	10	29	185	0,136
Euthyroid (1 versus. all)	euth	24	238	1762	0,119
Satimage (4 versus. all)	sat	36	626	5809	0,097
Vowel (1 versus. all)	vow	10	90	900	0,091
Abalone (18 versus. 9)	a18-9	08	42	689	0,057
Yeast (9 versus. 1)	y9-1	08	20	463	0,041
Car (3 versus. all)	car	06	69	1659	0,04
Yeast (5 versus. all)	y5	08	51	1433	0,034
Abalone (19 versus. all)	a19	08	32	4145	0,008

devido ao tempo de processamento dos dados. A fim de criar esses conjuntos distintos, para cada base de dados foram sorteados dez vezes, de forma aleatória, 70% dos dados para treinamento e 30% para validação. Sendo assim, para cada base de dados, foram criados dez conjuntos diferentes de treinamento e validação.

Para definir o melhor número de neurônios na camada escondida para cada uma das bases de dados, foi implementada a metodologia de busca *grid search* com validação cruzada *k-fold* com k = 5, em que o número de neurônios variava de 1 até 13 com passo de 3, podendo então assumir os seguintes valores: 1, 4, 7, 10 e 13.

Após o processo do *k-fold* para um conjunto de dados, uma matriz $10x5$ foi obtida. O melhor h^0 para cada caso (linha da matriz) foi selecionado. A moda de h^0 foi escolhida como o melhor número de neurônios para a camada escondida da rede neural para aquela determinada base de dados. Em um passo subsequente, todos os *folds* utilizados na validação cruzada, para cada um dos conjuntos gerados, foram utilizados para o treinamento do modelo. Após o treinamento, avaliou-se seu desempenho, considerando-se o respectivo conjunto de validação. A média do resultado dos dez conjuntos de validação é, então, apresentada como resultado final. Junto à avaliação do desempenho, também foi calculado o tempo de treinamento e validação de cada um dos algoritmos nas diferentes abordagens. O resultado do tempo apresentado consiste, então, na média

do tempo de treinamento e validação de cada um dos dez conjuntos de dados.

Para cada uma das quatro diferentes abordagens (AUC, G-mean, F1-score e AGm) foi utilizada a respectiva métrica durante o processo de validação cruzada como critério para definição do melhor número de neurônios da camada intermediária.

4.3.2 Resultados

Conforme a metodologia apresentada na subseção anterior, o valores médios para as abordagens AUC, G-mean, F1-score e AGm, obtidos pelos algoritmos Rprop, SMOTE, SMTTL, WWE, RMBoost, e para a abordagem apresentada (AUCMLP, do inglês *Area Under the ROC Curve multilayer perceptron*; GMLP, *G-mean multilayer perceptron*; FMLP, *F-score multilayer perceptron*, ou AGMLP, *Adjusted G-mean multilayer perceptron*) são exibidos nas tabelas 4.5, 4.7, 4.9 e 4.11 para as 16 bases de dados, seguidos dos respectivos tempos médios de treinamento e validação, apresentados nas tabelas 4.6, 4.8, 4.10 e 4.12.

Os maiores valores para a métrica em análise estão marcados em negrito.

Tabela 4.5: Valores médios para AUC

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'AUCMLP'
iono	85,66%	87,66%	88,56%	89,15%	**89,62%**	80,40%
pid	66,26%	69,01%	70,44%	71,98%	72,38%	**74,34%**
gmn	60,02%	62,40%	65,64%	63,64%	64,99%	**69,36%**
wpbc	56,25%	59,52%	56,55%	52,48%	**60,61%**	60,51%
veh	95,26%	96,16%	97,36%	96,79%	**97,72%**	96,43%
hrt	61,32%	60,53%	63,66%	**70,08%**	65,46%	69,86%
seg	99,57%	99,72%	99,69%	99,55%	**99,80%**	98,71%
gls7	88,69%	91,64%	91,28%	**92,99%**	92,45%	91,24%
euth	84,80%	89,15%	86,83%	88,23%	88,75%	**90,87%**
sat	76,66%	78,16%	78,09%	**79,80%**	77,71%	77,21%
vow	98,72%	97,39%	95,88%	95,43%	**99,39%**	96,77%
a18-9	71,83%	**82,12%**	81,69%	70,80%	78,39%	82,03%
y9-1	72,32%	67,31%	73,01%	69,88%	71,90%	**78,07%**
car	93,25%	90,78%	91,71%	**97,09%**	94,05%	95,73%
y5	61,21%	71,85%	71,17%	71,35%	65,85%	**79,34%**
a19	53,80%	76,91%	76,90%	57,57%	73,73%	**83,06%**
Ranking Médio	5.06	3.56	3.50	3.50	2.63	2.75

Tabela 4.6: Tempo médio de treinamento e validação para AUC (segundos)

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'AUCMLP'
iono	1,370	1,608	2,039	1,334	36,793	1,376
pid	1,359	1,966	1,984	1,740	37,806	1,323
gmn	2,022	2,089	2,380	2,403	50,548	1,924
wpbc	1,141	1,150	1,073	0,980	24,202	1,141
veh	1,446	1,785	3,862	2,103	56,437	1,867
hrt	1,387	2,190	3,197	1,292	73,011	1,246
seg	2,393	3,817	16,486	5,881	176,462	3,238
gls7	1,023	1,053	1,007	1,022	22,875	1,042
euth	2,662	2,868	6,353	4,193	101,608	2,895
sat	11,051	12,379	16,490	31,102	531,254	6,542
vow	1,765	1,771	1,975	1,962	58,863	1,941
a18-9	1,050	1,059	1,379	1,257	27,063	1,353
y9-1	1,218	1,540	2,104	1,228	32,348	1,402
car	2,297	3,883	7,565	2,974	311,684	1,872
y5	2,528	1,647	2,306	2,879	57,395	2,213
a19	4,534	2,767	18,052	11,981	147,837	4,002
Ranking Médio	2.06	3.00	4.31	3.25	6.00	2.37

Em uma primeira análise visual, observa-se que a abordagem proposta para AUC apresenta um resultado melhor que o Rprop clássico, além de ter *ranking* médio equivalente ao RAMOBoost, com tempo de processamento muito menor que este em todas as bases de dados.

Tabela 4.7: Valores médios para G-mean

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'GMLP'
iono	83,86%	89,19%	88,14%	86,14%	**90,57%**	78,84%
pid	72,07%	76,17%	76,00%	**76,45%**	74,23%	76,37%
gmn	67,16%	**70,97%**	69,37%	67,28%	67,13%	68,77%
wpbc	54,10%	50,23%	**62,26%**	56,46%	61,35%	59,07%
veh	95,53%	96,72%	**97,41%**	96,71%	97,27%	97,07%
hrt	55,97%	65,57%	66,06%	**76,21%**	66,03%	70,31%
seg	99,57%	99,44%	99,54%	99,57%	**99,95%**	98,93%
gls7	92,19%	90,31%	**93,01%**	91,39%	92,45%	91,40%
euth	90,67%	90,20%	90,46%	91,08%	88,07%	**91,43%**
sat	73,16%	76,36%	76,21%	79,22%	74,27%	**85,96%**
vow	99,11%	98,13%	98,81%	98,48%	**100,00%**	99,55%
a18-9	71,91%	82,54%	82,48%	80,85%	74,01%	**82,91%**
y9-1	64,78%	67,70%	54,79%	68,08%	71,10%	**73,86%**
car	94,58%	**97,62%**	92,98%	98,08%	94,58%	97,52%
y5	54,44%	80,91%	**81,11%**	70,64%	61,96%	78,85%
a19	10,49%	**84,49%**	84,46%	25,26%	34,48%	80,32%
Ranking Médio	4.88	3.56	3.06	3.25	3.50	2.75

Tabela 4.8: Tempo médio de treinamento e validação para G-mean (segundos)

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'GMLP'
iono	1,427	1,575	1,746	1,281	39,170	1,192
pid	1,106	1,076	1,437	1,201	31,527	1,147
gmn	1,270	1,379	1,888	1,642	31,603	1,872
wpbc	1,054	1,199	1,036	1,061	25,348	1,073
veh	1,273	1,472	3,595	1,917	53,847	1,658
hrt	1,200	2,376	3,153	1,133	33,443	1,214
seg	2,518	4,799	14,498	5,221	172,374	1,672
gls7	0,985	0,842	1,021	0,910	22,144	0,912
euth	1,638	2,301	5,805	3,471	83,363	2,694
sat	9,541	11,872	14,135	29,844	428,639	9,858
vow	1,956	1,840	2,161	1,964	54,982	1,853
a18-9	1,134	1,104	1,484	1,233	31,856	1,043
y9-1	1,226	1,376	1,739	1,241	23,452	1,049
car	1,990	2,809	5,658	3,431	232,341	2,199
y5	2,237	1,289	1,825	2,734	47,847	1,972
a19	3,337	2,805	18,315	11,947	522,545	2,863
Ranking Médio	2.19	2.44	4.44	3.50	6.00	2.44

Observando-se a proposta apresentada para a métrica G-mean como função de custo, em um primeiro momento pode-se constatar que esta teve um desempenho médio melhor que os outros algoritmos testados, tendo um tempo de processamento competitivo com o modelo padrão Rprop.

Tabela 4.9: Valores médios para F1-score

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'FMLP'
iono	83,51%	**85,95%**	83,06%	85,20%	88,60%	77,91%
pid	65,11%	69,17%	**69,65%**	68,98%	64,11%	66,42%
gmn	55,58%	59,51%	**60,07%**	57,16%	53,19%	57,66%
wpbc	30,36%	36,90%	39,74%	31,87%	**46,49%**	42,03%
veh	92,09%	92,44%	92,55%	92,02%	**94,29%**	92,30%
hrt	47,33%	43,43%	42,20%	52,40%	**54,00%**	52,17%
seg	97,89%	**98,51%**	99,40%	98,20%	99,70%	98,47%
gls7	**87,67%**	76,88%	77,63%	75,50%	79,27%	80,13%
euth	80,56%	76,08%	77,97%	80,16%	79,86%	**82,59%**
sat	57,93%	56,22%	56,91%	57,48%	**64,30%**	62,43%
vow	95,27%	91,94%	96,74%	93,04%	**99,44%**	95,01%
a18-9	**68,49%**	45,24%	38,29%	68,10%	51,80%	64,35%
y9-1	46,97%	37,04%	35,62%	45,69%	53,33%	**58,89%**
car	**91,61%**	86,11%	82,00%	71,87%	89,22%	84,47%
y5	22,76%	34,34%	35,82%	30,10%	**46,16%**	40,22%
a19	4,17%	**10,46%**	4,64%	9,81%	5,02%	6,36%
Ranking Médio	3.81	3.88	3.81	4.06	2.44	3.00

Tabela 4.10: Tempo médio de treinamento e validação para F1-score (segundos)

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'FMLP'
iono	2,238	2,635	2,896	2,262	63,497	1,968
pid	1,680	1,764	2,108	1,831	55,017	1,947
gmn	2,076	2,301	2,753	2,394	67,442	2,434
wpbc	1,856	1,898	1,606	1,691	37,363	1,771
veh	2,695	3,546	4,877	2,775	102,566	3,080
hrt	2,172	4,297	5,158	1,976	95,999	2,049
seg	3,397	7,898	17,590	6,072	228,785	2,806
gls7	1,730	1,514	1,598	1,610	39,222	1,481
euth	2,689	3,401	7,896	4,341	180,492	3,986
sat	15,741	16,405	19,713	34,682	707,848	18,664
vow	3,285	3,211	3,371	3,305	75,572	3,159
a18-9	1,988	2,120	2,747	1,823	64,545	1,953
y9-1	1,587	2,843	2,741	1,582	76,564	1,911
car	3,538	6,751	10,312	5,039	511,339	3,055
y5	3,543	2,643	3,442	4,137	88,564	3,071
a19	5,857	10,774	24,958	15,174	507,426	3,810
Ranking Médio	2.31	3.13	4.38	3.00	6.00	2.19

Ao inspecionar visualmente o desempenho da abordagem para a métrica F1-score e os algoritmos testados, pode-se constatar que a utilização da F1-score como função de custo traz uma pequena vantagem competitiva em relação ao resultado médio do Rprop, SMOTE, SMTTL e WWE. Apesar de ter *ranking* médio pior que o RAMOBoost, o tempo de processamento da FMLP é muito menor que o deste.

Tabela 4.11: Valores médios para AGm

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'AGMLP'
iono	90,32%	90,71%	90,35%	89,91%	**92,38%**	86,85%
pid	75,10%	70,94%	73,14%	75,53%	71,71%	**76,92%**
gmn	69,03%	67,45%	68,99%	72,90%	70,94%	**73,12%**
wpbc	65,99%	66,31%	66,14%	65,17%	**70,28%**	69,69%
veh	**96,95%**	96,60%	96,46%	96,23%	97,48%	96,73%
hrt	73,42%	71,87%	70,04%	73,17%	**74,19%**	69,57%
seg	99,60%	99,56%	99,84%	99,51%	**99,86%**	99,50%
gls7	92,73%	93,06%	92,65%	92,96%	**93,89%**	93,73%
euth	92,96%	92,43%	92,35%	**93,39%**	92,39%	93,38%
sat	84,89%	84,41%	85,15%	**86,57%**	85,46%	85,72%
vow	98,59%	99,00%	98,81%	98,43%	**99,86%**	96,22%
a18-9	87,50%	89,26%	89,16%	**89,84%**	82,88%	87,48%
y9-1	83,05%	76,97%	79,02%	80,45%	82,42%	**85,80%**
car	94,56%	96,65%	95,94%	96,01%	**97,67%**	97,52%
y5	72,84%	**83,94%**	84,19%	78,14%	76,55%	83,78%
a19	52,55%	76,35%	76,63%	62,39%	69,45%	**77,41%**
Ranking Médio	4.00	3.75	3.94	3.63	2.63	3.06

Tabela 4.12: Tempo médio de treinamento e validação para AGm (segundos)

Base	'Rprop'	'SMOTE'	'SMTTL'	'WWE'	'RMBoost'	'AGMLP'
iono	0,954	1,199	1,507	0,863	26,710	0,900
pid	0,843	0,925	1,347	0,923	25,072	1,131
gmn	1,075	1,297	2,112	1,324	34,804	1,223
wpbc	0,973	0,930	0,937	0,714	17,089	0,883
veh	1,279	1,696	4,452	1,814	45,801	1,329
hrt	0,956	1,710	2,880	0,810	53,612	0,918
seg	2,633	4,311	14,145	4,765	126,171	1,572
gls7	0,700	0,689	0,731	0,712	16,568	0,681
euth	1,218	1,890	5,682	2,941	66,049	1,556
sat	9,280	9,431	14,688	29,389	469,620	7,745
vow	1,762	2,042	2,083	2,528	40,000	1,368
a18-9	0,724	0,743	1,057	0,892	27,726	0,926
y9-1	1,023	1,400	2,009	1,141	39,244	1,022
car	1,705	2,534	6,423	2,722	220,024	1,280
y5	1,648	1,049	1,875	2,709	70,839	2,021
a19	2,775	2,427	18,402	11,711	250,892	2,455
Ranking Médio	2.13	2.81	4.69	3.38	6.00	2.00

Ao se observarem os resultados presentes na Tabela 4.11, verifica-se que o *ranking* médio da abordagem proposta, representada pelo algoritmo AGMLP, teve desempenho melhor que a abordagem clássica Rprop e os métodos SMOTE, SMTTL e WWE. Todavia, o desempenho médio foi pior que o RAMOBoost, de acordo com o *ranking* médio. Por outro lado, quando se observa a Tabela 4.12, o desempenho da abordagem proposta apresenta-se muito melhor que o RAMOBoost.

4.3.3 Testes estatísticos

Após calculados os resultados médios para cada uma das abordagens, testes estatísticos foram realizados para comparação dos modelos. Conforme apresentado anteriormente no Capítulo 3, foi demonstrado por Demsar (Demšar, 2006) que, para problemas de aprendizado de máquinas, quando se deseja comparar modelos de classificadores, os testes não-paramétricos de Wilcoxon (Gibbons and Chakraborti, 2011), para comparação de dois classificadores, e Friedman (1937), para comparação de múltiplos classificadores, são mais aconselháveis do que testes paramétricos como *Analysis of Variance* (ANOVA).

Assim como descrito na seção 3.4.2 deste trabalho, o teste de Friedman faz um

ranqueamento de L algoritmos ao longo de M bases de dados. O melhor desempenho recebe o valor 1, o segundo, 2, e assim por diante; o valor médio desse ranqueamento pode ser visualizado na última linha das tabelas dos resultados médios para cada um dos testes. Assumindo-se a hipótese nula, segundo a qual todos os algoritmos são equivalentes, o valor médio dos *rankings* deve ser equivalente, de forma que a estatística dada pela Equação (4.9) é distribuída de acordo com a distribuição-F com $L-1$ e $(L-1)(M-1)$ graus de liberdade.

$$F_F = \frac{(M-1)\chi_F^2}{M(L-1)-\chi_F^2} \tag{4.8}$$

$$\chi_F^2 = \frac{12M}{L(L+1)}\left(\sum_t R_t^2 - \frac{L(L+1)^2}{4}\right) \tag{4.9}$$

Quando a hipótese nula é rejeitada, Demšar (2006) sugere que outro teste seja feito, a fim de quantificar a diferença entre os algoritmos. O modelo mais utilizado para analisar o comportamento de um algoritmo em relação aos demais é o teste Bonferroni-Dunn *post hoc* (Dunn, 1961). Neste, o classificador em foco é considerado estatisticamente diferente de forma significativa se as médias dos *rankings* apresentarem uma diferença crítica (CD) mínima conforme a Equação 4.10, que pode ser encontrada em Demšar (2006).

$$CD = q_a\sqrt{\frac{L(L+1)}{6M}} \tag{4.10}$$

em que q_a é o valor crítico no nível de confiança de $1-\alpha$. Neste caso, o valor utilizado de α será de 0.1.

Obtidos os presentes resultados, os testes de significância foram conduzidos para analisar a abordagem apresentada. Primeiramente, o teste de Friedman (1937) foi conduzido com o valor do *ranking* médio de cada uma das métricas (presentes na última linha de cada uma das tabelas de resultados 4.5, 4.7, 4.9 e 4.11), e com o valor de *ranking* médio do tempo (presente na última linha de cada uma das tabelas de resultados 4.6, 4.8, 4.10 e 4.12). Conforme a tabela apresentada em Sheskin (2007), o valor crítico mínimo, quando $\alpha = 0.1$, $M = 16$ (número de base de dados) e $L = 6$ (número de algoritmos sob comparação), o qual garante que a hipótese nula seja rejeitada (H_0), deve ser $F_F = 1.9256$. Para os *rankings* médios apresentados nas tabelas de resultados das métricas, os valores da estatística F_F encontrada foram: 4.1235, 2.7496, 1.9697 e 1.3823, respectivamente. No caso dos *rankings* médios para o tempo médio de treinamento e validação, os valores da estatística F_F encontrada foram: 22.7953, 26.0758, 22.7104 e 35.3748.

Descartada a hipótese nula para todas as métricas (exceto para a métrica AGm [1]) e tempos analisados, o teste *post hoc* de *Bonferroni-Dunn* foi aplicado para checar se o desempenho do algoritmo proposto é diferente dos outros (metodologia um contra todos) no nível de confiança de $1 - \alpha$. Para $\alpha = 0.1$, a mínima CD alcançada para considerar dois algoritmos estatisticamente diferentes em relação ao desempenho deve ser de 1.7125. A diferença entre os *rankings* médios, de acordo com a metodologia de Bonferroni-Dunn *post hoc*, pode ser visualizada nas Tabelas 4.13, 4.14, 4.15 e 4.16, para as abordagens AUC, G-mean, F1-score e AGm, respectivamente. Os valores que superam o valor de CD mínima encontram-se em negrito.

Tabela 4.13: Bonferroni-Dunn *post hoc* (AUCMLP x todos)

AUCMLP versus					
	Rprop	SMOTE	SMTTL	WWE	RAMOBoost
AUC	**2.3125**	0.8125	0.7500	0.7500	0.1250
Tempo	0.3125	0.6250	**1.9375**	0.8750	**3.6250**

Tabela 4.14: Bonferroni-Dunn *post hoc* (GMLP x todos)

GMLP versus					
	Rprop	SMOTE	SMTTL	WWE	RAMOBoost
G-mean	**2.1250**	0.8125	0.3125	0.5000	0.7500
Tempo	0.2500	0.0000	**2.0000**	1.0625	**3.5625**

Tabela 4.15: Bonferroni-Dunn *post hoc* (FMLP x todos)

FMLP versus					
	Rprop	SMOTE	SMTTL	WWE	RAMOBoost
F1-score	0.8125	0.8750	0.8125	1.0625	0.5625
Tempo	0.1250	0.9375	**2.1875**	0.8125	**3.8125**

Tabela 4.16: Bonferroni-Dunn *post hoc* (AGMLP x todos)

AGMLP versus					
	Rprop	SMOTE	SMTTL	WWE	RAMOBoost
AGm	0.9375	0.6875	0.8750	0.5625	0.4375
Tempo	0.1250	0.8125	**2.6875**	1.3750	**4.0000**

[1] Apesar de a hipótese nula ter sido considerada, também foram realizados os testes *post hoc* para AGm.

Além dos testes de Friedman e Bonferroni-Dunn *post hoc*, o teste *post hoc* de Nemenyi (1962) foi utilizado para comparar os algoritmos no estilo "um contra um". Os resultados do teste *post hoc* estão representados nas Figuras 4.13, 4.14, 4.15, 4.16, 4.17, 4.18, 4.19 e 4.20, no formato de diagramas de diferença (Demšar, 2006), para cada uma das abordagens (AUC, G-mean, F1-score e AGm). No diagrama apresentado, os valores do *ranking* médio são dispostos de forma crescente na linha horizontal, tal que o melhor algoritmo encontra-se à esquerda e o pior à direita. Nesse tipo de representação, os algoritmos que não possuem diferença significativa, isto é, que possuem diferença crítica menor que 1.7213, são conectados por retas horizontais.

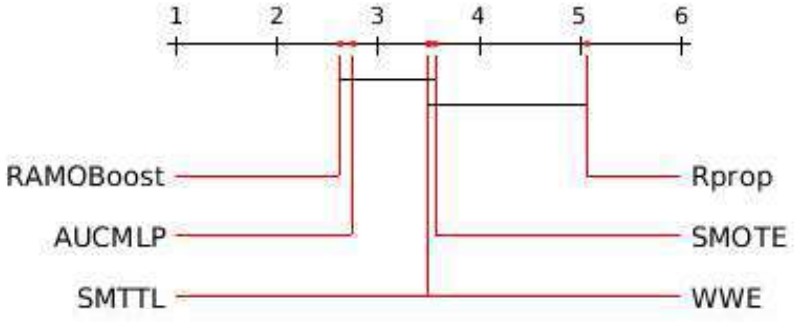

Figura 4.13: Diagrama de diferença crítica para abordagem AUC

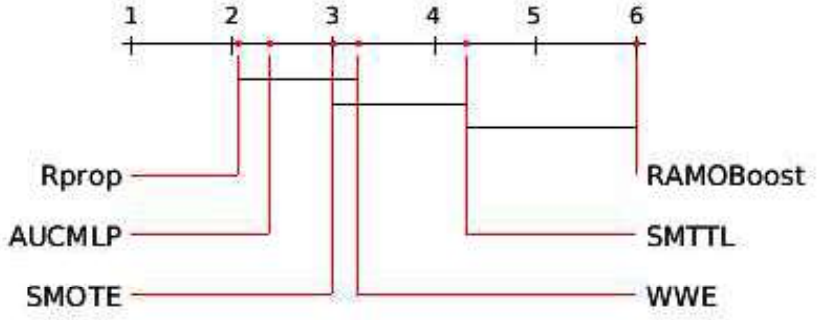

Figura 4.14: Diagrama de diferença crítica para abordagem AUC (tempo)

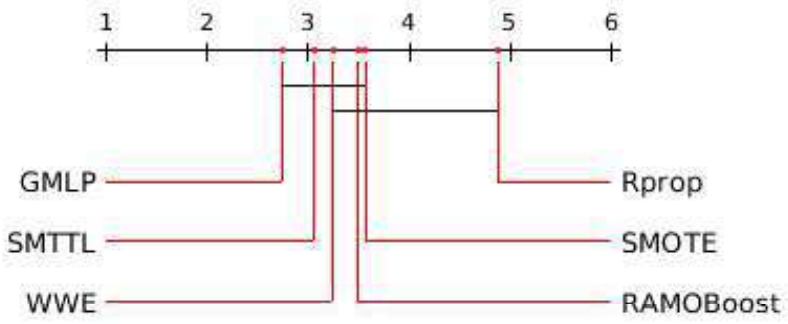

Figura 4.15: Diagrama de diferença crítica para abordagem G-mean

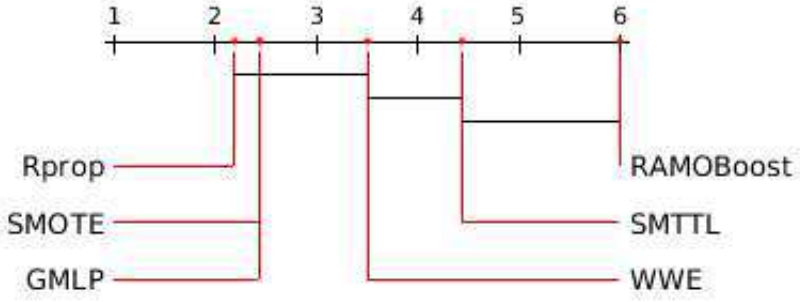

Figura 4.16: Diagrama de diferença crítica para abordagem G-mean (tempo)

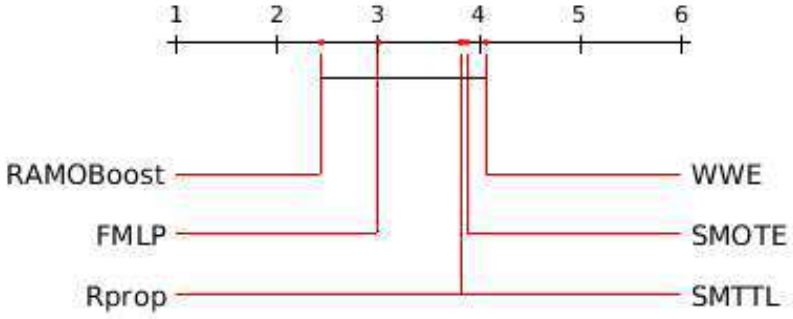

Figura 4.17: Diagrama de diferença crítica para abordagem F1-score

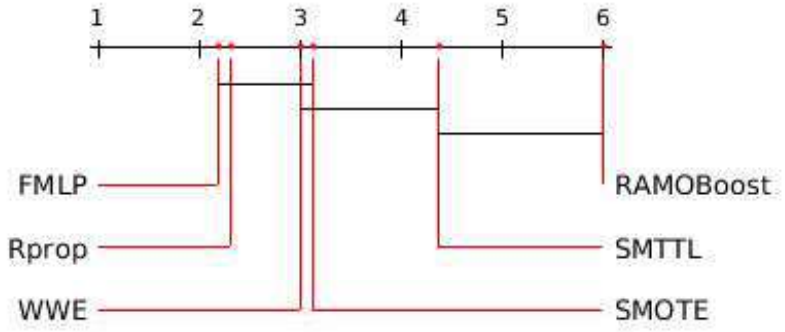

Figura 4.18: Diagrama de diferença crítica para abordagem F1-score (tempo)

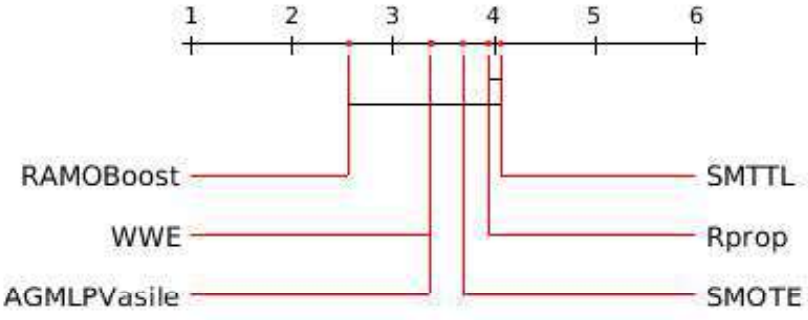

Figura 4.19: Diagrama de diferença crítica para abordagem AGm

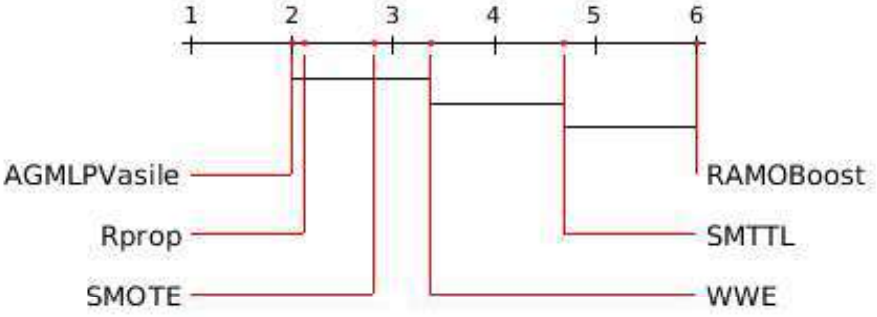

Figura 4.20: Diagrama de diferença crítica para abordagem AGm (tempo)

4.3.4 Discussão

4.3.4.1 Abordagem Função de custo AUC

Ao analisar os resultados dos *rankings* médios para a abordagem comparativa da função de custo AUC, observa-se que os resultados da métrica obtida são semelhantes aos obtidos pelo RAMOBoost e superiores à abordagem clássica e aos métodos de pré-processamento testados. Além disso, o tempo de execução médio para treinamento e

validação da abordagem proposta apresenta-se competitivo em relação à abordagem padrão, sem técnica para dados desbalanceados, e muito superior ao RAMOBoost.

Quando testes estatísticos são aplicados, a hipótese nula prova que os algoritmos são estatisticamente diferentes, tanto em relação ao resultado da AUC quanto em relação ao tempo de processamento. Quando a abordagem em análise é comparada contra as demais na Tabela 4.13, nota-se que o resultado da métrica é estatisticamente superior à abordagem padrão e igual às demais. Por outro lado, quando o tempo de processamento é levado em consideração, a abordagem apresentada torna-se estatisticamente diferente e melhor que o RAMOBoost e SMTTL.

Considerando o valor médio da métrica em análise e o tempo de processamento, a abordagem proposta mostrou-se estatisticamente equivalente ao SMOTE e WWE com *ranking* médio maior, tanto para o valor da métrica quanto para o tempo médio de treinamento e validação.

Estes resultados mostram que a abordagem proposta, quando utilizada para a métrica AUC, apresenta-se como uma solução estatisticamente equivalente ao SMOTE e WWE, com valores levemente superiores, provando-se uma boa alternativa para problemas nos quais a métrica em análise para seleção do melhor modelo é a AUC.

4.3.4.2 Abordagem Função de custo G-mean

Na primeira análise que considera o valor do *ranking* médio das Tabelas 4.7 e 4.8, a abordagem que utiliza o G-mean como função de custo teve o melhor valor de *ranking* médio e o segundo melhor valor de tempo de processamento (junto ao SMOTE e SMTTL). Ao ser aplicado o teste de Friedman, a hipótese nula foi descartada, tanto para a métrica do G-mean quanto para o tempo de processamento.

No teste estatístico de Bonferroni-Dunn *post hoc* (4.14), a abordagem apontada apresentou-se melhor do que o MLP padrão com Rprop, e estatisticamente igual às outras técnicas para dados desbalanceados para os valores de G-mean. Em relação ao tempo médio de treinamento e validação, o GMLP mostrou-se mais rápido que o RAMOBoost e o STMTTL, sendo estatisticamente diferente destes, e igual à abordagem padrão e ao SMOTE.

Quando analisados em conjunto os resultados do teste de Bonferroni-Dunn *post hoc* para o valor da métrica e para o tempo, percebe-se que a abordagem apresentada, tendo como função de custo o G-mean, mostra-se equivalente ao SMOTE e WWE, apresentando-se melhor que a abordagem padrão e mais rápida que o RAMOBoost e SMTTL. Sendo assim, revela-se uma alternativa bastante viável com bom resultados e bom tempo de processamento.

4.3.4.3 Abordagem Função de custo F1-score

A abordagem que utiliza a função de custo F1-score mostrou-se competitiva em relação às demais métricas, tomando-se como base os valores do *ranking* médio da Tabela 4.9. O tempo médio de treinamento e validação, entretanto, mostrou-se vantajoso para a abordagem FLMP em comparação com os outros algoritmos voltados para problemas de classes desbalanceadas.

O teste de Friedman mostrou que os algoritmos testados rejeitam a hipótese nula, tanto em relação aos valores da métrica quanto ao tempo. O teste de Bonferroni-Dunn *post hoc* foi então realizado e apresentado na Tabela 4.15, na qual se pode observar que, para valores de F1-score, os algoritmos são todos estatisticamente iguais ao FMLP para o valor da métrica F1-score. Quanto ao tempo de processamento, a abordagem que se utiliza da função de custo apresentou-se estatisticamente diferente do RAMOBoost e do SMTTL, tendo tempos de treinamento e validação inferiores aos destes.

Verifica-se pelos testes estatísticos que a metodologia apresentada para a função de custo F1-score é estatisticamente igual à abordagem padrão do MLP com RProp, apesar de apresentar melhores resultados de *ranking* médio.

Nota-se que todos os algoritmos testados apresentaram-se estatisticamente iguais à abordagem padrão para o F1-score. Um dos motivos apresentados para isso é o fato de o F1-score ser uma métrica constituída por uma função de difícil convergência, como apresentado em Dembczynski et al. (2011), Jansche (2005), Nan et al. (2012) e Parambath et al. (2014). Outro fator que pode ter influenciado o resultado é o número de iterações utilizado, o qual para esta abordagem fosse necessário um número mais elevado para a convergência.

4.3.4.4 Abordagem Função de custo AGm

Ao analisarem-se os valores da Tabela 4.11, observa-se que o RAMOBoost apresenta-se como o algoritmo com melhores valores de AGm para a maioria das bases de dados utilizadas no teste empírico, seguido da metodologia apresentada AGMLP. Todavia, a vantagem competitiva do RAMOBoost, apresentada em uma primeira análise visual, tem um custo alto no tempo para treinamento e validação, quando comparada à abordagem do MLP com Rprop, como pode ser observado na Tabela 4.12. Isso não acontece quando se utiliza a abordagem do AGMLP.

Quando o teste de Friedman foi realizado para os valores médios de AGm, a hipótese nula não foi rejeitada, mostrando que todos esses algoritmos poderiam ser considerados estatisticamente iguais em relação ao valor de AGm apresentado por eles. Todavia, quando o mesmo teste foi realizado para os valores da tabela de tempo médio (4.12), a hipótese nula foi rejeitada.

Aplicado o teste de Bonferroni-Dunn *post hoc*, observou-se que, apesar de serem estatisticamente iguais em relação à métrica em análise, o Rprop com função de custo AGm (AGMLP) mostrou-se diferente quanto ao RAMOBoost e ao SMTTL, e continuou igual ao RProp, SMOTE e WWE.

Esperava-se que o valor apresentado para a métrica AGm fosse melhor e estatisticamente diferente para a abordagem apresentada, todavia essa hipótese não foi confirmada. Uma das hipóteses pode estar correlacionada à derivada de alta complexidade. Entretanto, mesmo sendo estatisticamente igual ao algoritmo padrão, observa-se que, para a maioria das bases de dados utilizadas, o resultado apresentado pelo algoritmo AGMLP foi melhor que o Rprop com função de entropia cruzada.

4.4 Conclusão

A abordagem apresentada para transformação de métricas de análise de algoritmos em funções de custo, apresentou-se como uma alternativa fácil e viável para implementação, sendo preciso realizar apenas uma alteração durante a fase do *backpropagation*, o termo de erro (*error term*, Equação 4.1).

Ao analisarem-se os primeiros testes para as diferentes funções de custo, observa-se que, quando foi utilizado apenas um neurônio na camada escondida, a abordagem proposta se mostrou interessante por apresentar resultados superiores, principalmente para valores de desbalanceamento alto. Isso fez com que testes mais específicos e completos fossem apresentados.

Ao analisarem-se os resultados do teste empírico, os valores de *ranking* médio para a abordagem apresentada mostraram-se superiores ao MLP padrão com RProp em todas as quatro abordagens apresentadas. Nos testes estatísticos, foi confirmado que as abordagens com AUC e GMLP eram também estatisticamente diferentes em relação às demais. Já a abordagem com F1-score e AGm, apesar de apresentar valores de *ranking* médio superior, eram estatisticamente iguais ao RProp, assim como os demais algoritmos.

A técnica aqui apresentada mostra que para as métricas AUC, G-mean, F1-score e AGm, nas quais os algoritmos SMOTE, SMTTL, WWE e RAMOBoost são utilizados para tratar do desbalanceamento, a abordagem aqui apresentada mostra-se superior a todas quando levado em conta o *ranking* médio da métrica em análise ou do tempo gasto durante o treinamento e validação. Quando a abordagem de utilização dos termos da matriz de confusão para criação de funções de custo não se apresenta estatisticamente superior ao MLP padrão com o RProp modificado, as outras técnicas aqui testadas para problemas de classes desbalanceadas em redes neurais também não o fazem.

A abordagem apresentada neste capítulo mostra-se uma boa técnica para solução de problemas, cujo objetivo é utilizar métricas de desbalanceamento para a escolha dos melhores parâmetros de uma rede neural, mostrando-se estatisticamente igual às outras técnicas e com melhor tempo de processamento que o SMTTL e o RAMOBoost.

Capítulo 5

Conclusões e trabalhos futuros

Esta dissertação de mestrado tratou do impacto das funções de custo em redes neurais quando estas são utilizadas para problemas de dados desbalanceados, e pode ser dividida em duas partes.

Na primeira parte, apresentou-se um estudo mais profundo sobre o impacto da utilização da informação *a priori* na função de custo de entropia cruzada, e demonstrou-se que uma modificação simples no algoritmo de atualização dos pesos RProp pode levar a uma melhor estabilidade.

A inclusão da informação *a priori* na função de custo de entropia cruzada, junto ao algoritmo de atualização dos pesos do RProp modificado, assumiu neste trabalho o nome de *Cost sensitive logistic function multilayer perceptron* (CSLFMLP). Nos testes empíricos para esta abordagem, foram avaliadas três diferentes métricas: G-mean, AUC e AGm. Nas dezesseis bases de dados utilizadas, a abordagem apresentou um ranque médio melhor que a abordagem padrão do MLP com função de entropia cruzada e com o mesmo RProp modificado, sendo estatisticamente diferente para o G-mean e AGm. Mostrou-se também estatisticamente equivalente ao SMTTL, WWE e RAMOBoost para as três métricas analisadas.

A segunda parte do trabalho teve como objetivo apresentar uma abordagem de como transformar os elementos da matriz de confusão em relação à variável real (y) e observada ($\hat{y}$), a fim de que possa ser utilizada para construção de funções de custo. Nestas, tem-se como objetivo utilizar métricas específicas de problemas de classes desbalanceadas para avaliação do desempenho dos algoritmos de redes neurais. Tais métricas são então utilizadas para construção de funções de custo a serem otimizadas pela rede neural.

Tomaram-se quatro diferentes métricas de análise de problemas de classes desbalanceadas para exemplificar a aplicação da abordagem apresentada: AUC, G-mean, F1-score e AGm. Uma primeira análise mostrou que a abordagem se apresentava su-

perior à MLP padrão com função de custo de entropia cruzada. Desta maneira, testes empíricos de maior complexidade foram realizados.

Os resultados médios dos testes empíricos mostraram que a abordagem apresentada possuía valores de ranque médio superiores ao MLP padrão com o RProp modificado. Durantes os testes estatísticos, quando a abordagem apresentada era estatisticamente igual à abordagem padrão, os outros algoritmos utilizados para tratamento de problemas de classes desbalanceadas também eram. Entretanto, quando a abordagem se mostrava estatisticamente superior ao MLP sem técnica de desbalanceamento, os gráficos de Nemenyi mostravam que os outros algoritmos (SMTTL, SMOTE, WWE e RAMOBoost) nem sempre o eram.

A criação de funções de custo utilizando os termos da matriz de confusão mostrou-se uma abordagem abrangente, podendo ser muito explorada na utilização e criação de funções de custo que representem a realidade da métrica do problema de classe desbalanceada a solucionar.

O trabalho aqui desenvolvido conseguiu, assim, cumprir com o seu objetivo, ao mostrar que existe espaço para melhorar os resultados dos algoritmos de redes neurais alterando-se a função de custo de acordo com a métrica que se deseja utilizar para avaliar o desempenho do classificador, em problemas de classes desbalanceadas.

5.0.1 Trabalhos futuros

As propostas de continuidade em trabalhos futuros podem ser divididas em três campos distintos: criação de diferentes métricas como funções de custo, análise das métricas como funções de custo em diferentes tipos de redes e utilização de diferentes algoritmos de otimização.

A primeira trata-se de utilizar e adaptar a abordagem apresentada na segunda parte deste trabalho para criar outras funções de custo, baseadas em outras métricas de análise de problemas de classes desbalanceadas.

A segunda proposta é analisar as abordagens apresentadas em redes de maior complexidade, como *Recurrent Neural Network* (RNN), *Long-short term memory* (LSTM) e CNN, observando seu comportamento para diferentes tipos de problemas específicos. Para isso, será preciso um grande poder computacional e tempo de processamento, sendo necessários processamentos múltiplos em serviços de nuvem pública.

A última proposta de continuidade é utilizar diferentes funções de otimização e atualização dos pesos, como por exemplo: Adadelta (Zeiler, 2012), Adam (Kingma and Ba, 2014) e *Root Mean Square Propagation* (RMSProp) (Tieleman and Hinton, 2012), as quais se popularizaram após o início deste trabalho.

Referências Bibliográficas

Alejo, R., García, V., Sotoca, J. M., Mollineda, R. A., and Sánchez, J. S. (2007). Improving the performance of the rbf neural networks trained with imbalanced samples. In *Computational and Ambient Intelligence*, pages 162–169. Springer.

Antanasijević, J., Antanasijević, D., Pocajt, V., Trišović, N., and Fodor-Csorba, K. (2016). A qspr study on the liquid crystallinity of five-ring bent-core molecules using decision trees, mars and artificial neural networks. *RSC Advances*, 6(22):18452–18464.

Arar, Ö. F. and Ayan, K. (2015). Software defect prediction using cost-sensitive neural network. *Applied Soft Computing*, 33:263–277.

Bacha, K., Henao, H., Gossa, M., and Capolino, G.-A. (2008). Induction machine fault detection using stray flux emf measurement and neural network-based decision. *Electric Power Systems Research*, 78(7):1247–1255.

Barandela, R., Sánchez, J. S., García, V., and Ferri, F. J. (2003). Learning from imbalanced sets through resampling and weighting. In *IbPRIA*, pages 80–88. Springer.

Barandela, R., Valdovinos, R. M., Sánchez, J. S., and Ferri, F. J. (2004). The imbalanced training sample problem: Under or over sampling? In *Structural, Syntactic, and Statistical Pattern Recognition*, pages 806–814. Springer.

Batista, G. E., Bazzan, A. L., and Monard, M. C. (2003). Balancing training data for automated annotation of keywords: a case study. In *WOB*, pages 10–18.

Batista, G. E., Prati, R. C., and Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, 6(1):20–29.

Batuwita, R. and Palade, V. (2009). A new performance measure for class imbalance learning. application to bioinformatics problems. In *Machine Learning and Applications, 2009. ICMLA'09. International Conference on*, pages 545–550. IEEE.

Batuwita, R. and Palade, V. (2012). Adjusted geometric-mean: a novel performance measure for imbalanced bioinformatics datasets learning. *Journal of bioinformatics and computational biology*, 10(04):1250003.

Breiman, L. (1996). Bagging predictors. *Machine learning*, 24(2):123–140.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Caruana, R. and Niculescu-Mizil, A. (2004). Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 69–78. ACM.

Castro, C. L. and Braga, A. P. (2008). Optimization of the area under the roc curve. In *Neural Networks, 2008. SBRN'08. 10th Brazilian Symposium on*, pages 141–146. IEEE.

Castro, C. L. and Braga, A. P. (2013). Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *Neural Networks and Learning Systems, IEEE Transactions on*, 24(6):888–899.

Castro, C. L. and de Pádua Braga, A. (2009). Artificial neural networks learning in roc space. In *IJCCI*, pages 484–489.

Chawla, N., Japkowicz, N., and Kolcz, A. (2004a). Special issue on learning from imbalanced datasets, sigkdd explorations. In *ACM SIGKDD*.

Chawla, N., Lazarevic, A., Hall, L., and Bowyer, K. (2003). Smoteboost: Improving prediction of the minority class in boosting. *Knowledge Discovery in Databases: PKDD 2003*, pages 107–119.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, pages 321–357.

Chawla, N. V., Japkowicz, N., and Kotcz, A. (2004b). Editorial: special issue on learning from imbalanced data sets. *ACM Sigkdd Explorations Newsletter*, 6(1):1–6.

Chen, S., He, H., and Garcia, E. A. (2010). Ramoboost: ranked minority oversampling in boosting. *IEEE Transactions on Neural Networks*, 21(10):1624–1642.

Cho, B. H., Yu, H., Kim, K.-W., Kim, T. H., Kim, I. Y., and Kim, S. I. (2008). Application of irregular and unbalanced data to predict diabetic nephropathy using visualization and feature selection methods. *Artificial intelligence in medicine*, 42(1):37–53.

Dembczynski, K. J., Waegeman, W., Cheng, W., and Hüllermeier, E. (2011). An exact algorithm for f-measure maximization. In *Advances in neural information processing systems*, pages 1404–1412.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 155–164. ACM.

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64.

Fallahi, A. and Jafari, S. (2011). An expert system for detection of breast cancer using data preprocessing and bayesian network. *International Journal of Advanced Science and Technology*, 34:65–70.

Fan, W., Stolfo, S. J., Zhang, J., and Chan, P. K. (1999). Adacost: misclassification cost-sensitive boosting. In *Icml*, volume 99, pages 97–105.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874.

Frank, A. (2010). Uci machine learning repository. *http://archive. ics. uci. edu/ml*.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *Icml*, volume 96, pages 148–156.

Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701.

Gibbons, J. D. and Chakraborti, S. (2011). Nonparametric statistical inference. In *International encyclopedia of statistical science*, pages 977–979. Springer.

Goodenough, D. J., Rossmann, K., and Lusted, L. B. (1974). Radiographic applications of receiver operating characteristic (roc) curves. *Radiology*, 110(1):89–95.

Guo, H. and Viktor, H. L. (2004). Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM Sigkdd Explorations Newsletter*, 6(1):30–39.

Hanley, J. A. and McNeil, B. J. (1982). The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143(1):29–36.

Hart, P. (1968). The condensed nearest neighbor rule (corresp.). *IEEE transactions on information theory*, 14(3):515–516.

He, H., Bai, Y., Garcia, E. A., and Li, S. (2008). 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*, pages 1322–1328. IEEE.

He, H., Garcia, E., et al. (2009). Learning from imbalanced data. *Knowledge and Data Engineering, IEEE Transactions on*, 21(9):1263–1284.

Hong, X., Chen, S., and Harris, C. J. (2007). A kernel-based two-class classifier for imbalanced data sets. *IEEE Transactions on neural networks*, 18(1):28–41.

Hripcsak, G. and Rothschild, A. S. (2005). Agreement, the f-measure, and reliability in information retrieval. *Journal of the American Medical Informatics Association*, 12(3):296–298.

Jansche, M. (2005). Maximum expected f-measure training of logistic regression models. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 692–699. Association for Computational Linguistics.

Japkowicz, N. et al. (2000). Learning from imbalanced data sets: a comparison of various strategies. In *AAAI workshop on learning from imbalanced data sets*, volume 68, pages 10–15. Menlo Park, CA.

Joachims, T. (2005). A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM.

Kim, H.-J., Jo, N.-O., and Shin, K.-S. (2016). Optimization of cluster-based evolutionary undersampling for the artificial neural networks in corporate bankruptcy prediction. *Expert Systems with Applications*, 59:226–234.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kubat, M., Holte, R. C., and Matwin, S. (1998). Machine learning for the detection of oil spills in satellite radar images. *Machine learning*, 30(2-3):195–215.

Kubat, M., Matwin, S., et al. (1997). Addressing the curse of imbalanced training sets: one-sided selection. In *ICML*, volume 97, pages 179–186. Nashville, USA.

Kukar, M., Kononenko, I., et al. (1998). Cost-sensitive learning with neural networks. In *ECAI*, pages 445–449.

Lan, J., Hu, M. Y., Patuwo, E., and Zhang, G. P. (2010). An investigation of neural network classifiers with unequal misclassification costs and group sizes. *Decision Support Systems*, 48(4):582–591.

Lusted, L. B. (1971). Decision-making studies in patient management. *New England Journal of Medicine*, 284(8):416–424.

Mani, I. and Zhang, I. (2003). knn approach to unbalanced data distributions: a case study involving information extraction. In *Proceedings of workshop on learning from imbalanced datasets*, volume 126.

Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1994). Machine learning, neural and statistical classification.

Nan, Y., Chai, K. M., Lee, W. S., and Chieu, H. L. (2012). Optimizing f-measure: A tale of two approaches. *arXiv preprint arXiv:1206.4625*.

Nemenyi, P. (1962). Distribution-free multiple comparisons. In *Biometrics*, volume 18, page 263. INTERNATIONAL BIOMETRIC SOC 1441 I ST, NW, SUITE 700, WASHINGTON, DC 20005-2210.

Oh, S.-H. (2011). Error back-propagation algorithm for classification of imbalanced data. *Neurocomputing*, 74(6):1058–1061.

Parambath, S. P., Usunier, N., and Grandvalet, Y. (2014). Optimizing f-measures by cost-sensitive classification. In *Advances in Neural Information Processing Systems*, pages 2123–2131.

Pazzani, M. and Billsus, D. (1997). Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331.

Phung, S. L., Bouzerdoum, A., and Nguyen, G. H. (2009). Learning pattern classification tasks with imbalanced data sets.

Provost, F. and Fawcett, T. (2001). Robust classification for imprecise environments. *Machine learning*, 42(3):203–231.

Rakotomamonjy, A. (2004). Optimizing area under roc curves with svms.

Riedmiller, M. and Braun, H. (1992). Rprop-a fast adaptive learning algorithm. In *Proc. of ISCIS VII), Universitat*. Citeseer.

Riedmiller, M. and Braun, H. (1993). A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE.

Sasaki, Y. et al. (2007). The truth of the f-measure. *Teach Tutor mater*, 1(5).

Sheskin, D. J. (2007). Handbook of parametric and nonparametric statistical procedures.

Song, L., Li, D., Zeng, X., Wu, Y., Guo, L., and Zou, Q. (2014). ndna-prot: identification of dna-binding proteins based on unbalanced classification. *BMC bioinformatics*, 15(1):298.

Sordoni, A., Galley, M., Auli, M., Brockett, C., Ji, Y., Mitchell, M., Nie, J.-Y., Gao, J., and Dolan, B. (2015). A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714*.

Sun, Y., Kamel, M. S., Wong, A. K., and Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, 40(12):3358–3378.

Swets, J. A. (1979). Roc analysis applied to the evaluation of medical imaging techniques. *Investigative radiology*, 14(2):109–121.

Thai-Nghe, N., Gantner, Z., and Schmidt-Thieme, L. (2010). Cost-sensitive learning methods for imbalanced data. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Tomek, I. (1976). Two modifications of cnn. *IEEE Trans. Systems, Man and Cybernetics*, 6:769–772.

Van Rijsbergen, C. (1979). Infonnation retrieval.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *ACM Sigkdd Explorations Newsletter*, 6(1):7–19.

Wilson, D. L. (1972). Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man, and Cybernetics*, 2(3):408–421.

Xu, L. and Chow, M.-Y. (2006). A classification approach for power distribution systems fault cause identification. *IEEE Transactions on Power Systems*, 21(1):53–60.

Xu, L., Chow, M.-Y., Timmis, J., and Taylor, L. S. (2007). Power distribution outage cause identification with imbalanced data using artificial immune recognition system (airs) algorithm. *IEEE Transactions on Power Systems*, 22(1):198–204.

Yan, L., Dodier, R. H., Mozer, M., and Wolniewicz, R. H. (2003). Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 848–855.

Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.

Zhao, X.-M., Li, X., Chen, L., and Aihara, K. (2008). Protein classification with imbalanced data. *Proteins: Structure, function, and bioinformatics*, 70(4):1125–1132.

Zheng, J. (2010). Cost-sensitive boosting neural networks for software defect prediction. *Expert Systems with Applications*, 37(6):4537–4543.

Zhou, Z.-H. and Liu, X.-Y. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, 18(1):63–77.