UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MÁRCIO PALHETA PIEDADE

UMA ABORDAGEM DE APRENDIZAGEM PROFUNDA QUE USA FUNÇÕES ASSIMÉTRICAS PARA MODELAGEM DE PONTUAÇÃO DE CRÉDITO NO VAREJO

Manaus/AM
Junho de 2020

UNIVERSIDADE FEDERAL DO AMAZONAS

INSTITUTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

MÁRCIO PALHETA PIEDADE

UMA ABORDAGEM DE APRENDIZAGEM PROFUNDA QUE USA FUNÇÕES ASSIMÉTRICAS PARA MODELAGEM DE PONTUAÇÃO DE CRÉDITO NO VAREJO

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Computação da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Doutor em Ciência da Computação.

Orientador: Marco Antônio Pinheiro de Cristo

Manaus/AM

Junho de 2020

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Piedade, Márcio Palheta
P613a Uma abordagem de aprendizagem profunda que usa funções assimétricas para modelagem de pontuação de crédito no varejo / Márcio Palheta Piedade. 2020
123 f. : il. color; 31 cm.

Orientador: Marco Antônio Pinheiro de Cristo
Tese (Ciência da Computação) - Universidade Federal do Amazonas.

1. Funções assimétricas de perda. 2. Aprendizagem profunda. 3. Pontuação de crédito. 4. Crediário no Varejo. 5. Otimização I. Marco Antônio Pinheiro de Cristo. II. Universidade Federal do Amazonas
III. Título

À memória do meu querido avô e pai, Edson Palheta, que nunca mediu esforços para que eu tivesse acesso à melhor educação possível, estando sempre ao meu lado, sonhando, apoiando, ajudando e incentivando, até o seu último dia conosco.

Agradecimentos

A Deus.

Aos meus queridos pais Edson Palheta, Dinair Palheta e Elizabeth Palheta, por uma vida dedicada a mim e às minhas irmãs, Marta Palheta e Michelle Palheta. Vocês nos mostraram que o estudo é a única arma que temos para mudar os rumos da nossa vida. Nos ensinaram que o trabalho honesto, por mais difícil que seja, é valoroso e motivo de muito orgulho.

À minha esposa Neide Palheta, por toda a alegria, força e incentivo que você dedicou a mim. Você, Neidinha, foi o voto decisivo para eu começar essa fase da minha vida acadêmica. Mesmo sabendo que nossa vida financeira seria muito abalada pela decisão de me tornar pesquisador, você não hesitou e me disse: "Amor, não desista do seu sonho. Se faltar comida, um dia a gente almoça na minha mãe e, no outro, janta na sua".

Aos meus Palhetinhas, Matheus e Moisés Palheta, que fazem meus dias mais felizes e dinâmicos. Nossas piadas bobas, nossas corridas, nossos treinos, nossos passeios, são momentos que renovam minhas forças e me permitem avançar.

Agradeço ao Programa de Pós-Graduação do Instituto de Computação (IComp) da Universidade Federal do Amazonas (UFAM) pelo acolhimento, atenção e ensinamentos.

À FAPEAM pelo apoio financeiro, por meio do pagamento mensal de uma bolsa de pesquisa, que me deu o suporte determinante para conclusão deste trabalho.

"Sempre adiante; sempre mais longe; sempre mais alto."
(Léon Denis)

Resumo

Instituições credoras precisam lidar com as incertezas do negócio, criando estratégias que diminuam os riscos associados à concessão de crédito de seus clientes. Para lidar com este problema, foram desenvolvidos modelos quantitativos de previsão de risco baseados em dados cadastrais e comportamentais dos clientes. Nos últimos anos, novas gerações destes modelos, baseadas em aprendizagem de máquina, têm sido comumente usadas por instituições financeiras e de varejo. De forma geral, este problema é formulado como um problema de classificação binária onde se deseja discriminar bons de maus pagadores. Por este ser um problema de natureza desbalanceada (normalmente, há mais bons que maus pagadores), é comum a adoção de estratégias que levam à sub-representação ou extrapolação de dados e, consequentemente, com alteração da distribuição real das amostras, o que pode influenciar negativamente no desempenho dos modelos. Além disso, muitas vezes estes modelos não tiram proveito de particularidades das políticas de crédito nas quais eles serão empregados. Estas podem dar diferentes valores para diferentes tipos de erro, aplicando diferentes critérios para diferentes partes das listas ordenadas de escores de crédito. Uma forma de lidar com tais problemas é criar modelos que aprendam diretamente o ranking de crédito (ou seja, qual a ordem esperada entre dois clientes, dado os seus riscos) em lugar da distinção entre bons e maus. Um inconveniente desta abordagem é que ela tem custo de aprendizado maior, uma vez que o modelo deve analisar pares de instâncias. Contudo, a literatura recente de aprendizagem de máquina tem produzido muitas técnicas de equivalência de problemas capazes de otimizar tarefas de ranking de forma robusta a desbalanceamento, com custos de treino comuns aos da tarefa de classificação. Além disso, com grandes massas de dados e complexos padrões típicos de modelos de comportamento usados no varejo, é possível adotar modelos baseados em aprendizagem profunda, que têm sido usados com sucesso em uma grande variedade de aplicações. Neste trabalho, apresentamos modelos de aprendizagem profunda para o problema de modelagem de crédito para varejo que envolve dados comportamentais na entrada. Para tanto, tratamos o problema com uma solução de equivalência entre classificação binária e ranking

bipartido, utilizando para isso uma função de perda assimétrica, com hiper-parâmetros aprendidos durante o treino. Desta forma, associamos as vantagens das soluções de classificação binária com as de um modelo de ranking bipartido, ou seja, baixo custo de treinamento, possibilidade de calibrar o grau de tolerância a erros em partes específicas do ranking e robustez a desbalanceamento. Por meio da avaliação em dois conjuntos de dados de grande escala, um público e outro privado, observamos que o modelo proposto é capaz de superar vários outros modelos baseados em aprendizagem superficial e profunda.

Abstract

Credit institutions need to deal with the uncertainties of the business by creating strategies that reduce the risks associated with granting credit to their customers. To address this problem, quantitative risk prediction models based on application and behavioral customer data have been developed. In recent years, new generations of these models, based on machine learning, have been commonly used by financial and retail institutions. In general, this problem is formulated as a binary classification problem in which we want to discriminate between good and bad payers. As it is a problem of an unbalanced nature (there are generally more good than bad payers), it is common to adopt strategies that lead to underrepresentation or extrapolation of data and, consequently, to a distribution of samples other than the actual one, which affects the performance of the models. Moreover, these models usually do not take advantage of the particular credit policies adopted where they will be deployed. Such policies can weight differently different types of error by applying different criteria to different parts of the ordered lists of credit scores. An approach to deal with such problems is to create models that directly learn the credit ranking (ie, what is the expected order between two customers, given their risks) rather than the distinction between good and bad payers. A drawback of this approach is that it has a higher learning cost, since the model must analyze pairs of instances. However, the recent literature on machine learning has produced many techniques, based on problem equivalence, capable of optimizing ranking tasks in a robust way to imbalance, with the same training costs of binary classification tasks. In addition, with large datasets and the complexity of retail customer behavior, it is possible to adopt models based on deep learning that have been used successfully in a wide range of applications. In this paper, we present deep learning models for the retail credit modeling problem where the customer representation includes their behavior. For this, we cope with the problem with a solution of equivalence between binary classification and bipartite ranking, using an asymmetric loss function with hyperparameters learned during the training. By doing so, we associate the advantages of binary classification solutions with those of

a bipartite ranking model, that is, low training costs, the possibility to calibrate the degree of tolerance to errors in specific parts of the ranking and robustness to imbalance. By evaluating our technique in two large-scale datasets, a public and a private one, we observed that it is able to outperform several other shallow and deep learning strategies.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Diferentes instituições, de natureza financeira ou não, oferecem crédito aos seus clientes que, em contrapartida, assumem a responsabilidade de devolver o valor emprestado, acrescido de juros e correção monetária, por meio de parcelas que vencem ao longo de períodos de tempo previamente acordados [Wang et al., 2015]. Ao longo da vigência do compromisso entre os credores e seus clientes, as instituições precisam lidar com os riscos associados à concessão [Yeh & Lien, 2009]. Para isso, é comum que sejam adotadas estratégias que quantifiquem o risco de inadimplência de um determinado cliente. Neste trabalho, estamos interessados em estudar e avaliar essas estratégias, a fim de mitigar os riscos do negócio de concessão de crédito.

Na mercado de concessão de crédito, a ferramenta utilizada inicialmente foi a análise de risco feita por especialistas humanos, na qual os especialistas usavam abordagens manuais, empíricas e subjetivas para avaliar a capacidade de pagamento de um dado cliente. Contudo, essa não é uma abordagem escalável, uma vez que o aumento do número de clientes implica no aumento de contratações de novos colaboradores para as análises de crédito. Além disso, a eficiência dessa abordagem depende diretamente da experiência do especialista humano [Thomas, 2000]. Para contornar essas limitações, foram desenvolvidos modelos quantitativos baseados em dados cadastrais e comportamentais, capazes de prever automaticamente o risco de crédito de determinado cliente. Esses modelos são conhecidos na literatura como modelos de pontuação de crédito (*credit scoring*) [Diniz & Louzada, 2013].

A história dos modelos de *credit scoring* começou na década de 40, com o surgimento das primeiras estratégias estruturadas para previsão de risco de crédito e acompanhou o desenvolvimento e aperfeiçoamento tecnológico de hardware e software, ao longo do tempo [Thomas, 2000]. Essa evolução permitiu que classificássemos esses modelos em três gerações distintas:

- Primeira geração: essa geração marca o início do desenvolvimento e aplicação de métodos estatísticos para *credit scoring*, sendo o artigo de Durand [1941] considerado o primeiro trabalho dessa área;

- Segunda geração: acompanhando o avanço tecnológico de hardware e software, essa geração foi marcada pelo desenvolvimento de modelos baseados em métodos clássicos de aprendizagem de máquina (*Machine Learning*), como: *K-Nearest Neighbor*, *Logistic Regression*, *Classification Trees* e *Artificial Neural Networks* [Yeh & Lien, 2009; Wang et al., 2015];

- Terceira geração: essa geração representa o atual estado-da-arte em análise de crédito baseada em escores, onde o gigantesco volume de dados e o aumento do poder de armazenamento e processamento nos permitiram criar modelos de aprendizagem profunda (*Deep Learning*), que superaram a capacidade preditiva dos modelos propostos nas gerações anteriores.

Em geral, os trabalhos da literatura definem *credit scoring* como um problema de classificação binária, onde queremos discriminar entre bons e maus pagadores. Porém, como observado por Lyn et al. [2002], bases de dados reais de problemas de crédito são desbalanceadas por natureza, com uma quantidade de bons pagadores muito maior que a de maus. Este alto grau de desbalanceamento pode influenciar o desempenho de modelos de classificação. Por isso, os modelos de crédito costumam adotar estratégias para mitigar o desbalanceamento. Algumas estratégias comuns consistem em criar novos exemplos da classe minoritária de maneira artificial como, por exemplo, duplicando aleatoriamente suas instâncias, sendo essa estratégia conhecida como o como *oversampling*; ou amostrando um número menor de exemplos da classe majoritária, o que representa a estratégia conhecida como *undersampling*. Contudo, o uso dessas estratégias altera a distribuição real das amostras e pode influenciar o desempenho dos modelos, mascarando sua sensibilidade ou robustez a desbalanceamento [Khan et al., 2018].

Embora represente a minoria das instâncias de clientes, a classificação correta dos maus pagadores é importante para as instituições credoras, pois a perda gerada na concessão de crédito a um cliente ruim é maior do que o ganho obtido com um contrato assinado com um cliente que paga regularmente o empréstimo [Qian et al., 2014]. Além disso, a classificação com maior atenção à classe minoritária de crédito ruim pode, além de evitar perda em dívidas não-recebíveis, aumentar a eficiência da gestão do capital econômico das instituições credoras [Yu et al., 2018].

No dia-a-dia, agentes de crédito necessitam que os modelos propostos consigam prever corretamente a classe minoritária, além de garantir a equivalência entre o escore recebido por um cliente e o seu risco de crédito. Neste caso, a coleção de escores gerada para um dado grupo de clientes pode ser analisada como uma lista de clientes ordenada conforme seus riscos (ranking) onde, quanto maior o risco de um cliente se tornar inadimplente, menor sua pontuação no ranking. Esta abordagem é interessante uma vez que, em muitos casos, *credit scoring* pode ser visto como um problema de ranqueamento, onde estamos interessados em otimizar pontos específicos do ranking, que serão mais afetados por políticas de crédito. É comum, por exemplo, que o início e o fim do ranking sejam mais importantes para uma certa política de concessão de crédito, uma vez que clientes de menor pontuação (e maior risco de inadimplência) devem ter crédito negado, enquanto clientes do topo do ranking, onde o risco é menor, devem ter aprovação do empréstimo ou aumento automático do limite de crédito. Uma forma de lidar com esse tipo de problema é criar modelos que aprendam diretamente a ordem esperada entre dois clientes, dados os seus riscos. Contudo, essa abordagem tem maior custo de treinamento, uma vez que o modelo deve analisar tuplas de instâncias (normalmente, pares).

Problemas de ranking, como o descrito anteriormente, são conhecidos como bipartidos, uma vez que envolvem apenas duas classes de dados, em nosso caso particular, bons e maus pagadores. Na literatura de aprendizagem de máquina, sempre houve grande interesse no estudo da equivalência entre problemas de ranking bipartido e problemas de classificação binária, o que resultou no desenvolvimento de classes de funções de perda[1] que podem ser usadas simultaneamente para resolver ambas as tarefas [Ertekin & Rudin, 2011; Menon & Williamson, 2016]. Essas funções tratam diferentes erros de classificação de forma não simétrica, sendo por isso conhecidas como funções assimétricas. Dada a equivalência das funções para as tarefas, as vantagens associadas à solução da classificação binária podem ser transferidas para o ranking bipartido e vice-versa. Assim, é possível otimizar ranking sem a necessidade de análise de pares de instâncias, com redução do custo de treinamento, além de tornar possível calibrar o grau de tolerância a erros em partes específicas do ranking [Narasimhan & Agarwal, 2013].

A qualidade de um algoritmo de aprendizado de máquina pode ser avaliada a partir de diferentes métricas de desempenho, apropriadas ao contexto de cada problema. Os métodos de classificação utilizados em *credit scoring* costumam ser avaliados pela

[1]Em algoritmos de otimização tipicamente usados em aprendizagem de máquina, funções de perda (*loss functions*) são aquelas que mapeiam um evento ao seu custo. Este custo permite que os algoritmos determinem, por exemplo, o grau de proximidade de uma solução ótima para o problema em questão.

acurácia e pela área sob a curva ROC (AUC)[Zweig & Campbell, 1993]. Para medir a acurácia de um modelo de *credit scoring*, um ponto de corte (*cutoff*) é definido no ranking final de pontuação, para que os clientes com pontuação acima do *cutoff* sejam classificados como bons e os clientes com pontuação abaixo do *cutoff* sejam classificados como maus pagadores [Diniz & Louzada, 2013]. No entanto, Yan et al. [2003] afirmam que a acurácia pode não ser uma métrica adequada para avaliação de modelos que resolvem problemas em contextos onde existe alto grau de desbalanceamento, uma vez que pode ser enganada por modelos que classificam em favor da classe majoritária. AUC, por sua vez, é uma métrica para avaliação de desempenho de modelos de ranking, comumente usada para avaliação de modelos de classificação binária, que calcula a probabilidade de um modelo classificar uma instância positiva com uma pontuação maior que a pontuação de uma instância negativa (cf. Seção 2.1.2).

Embora os modelos de classificação de pontuação de crédito sejam avaliados por métricas de classificação e ranking, esses modelos geralmente são projetados para otimizar apenas métricas de classificação, como precisão, mas não são projetados para otimizar métricas de ranking, como AUC e KS [Zhu et al., 2018]. Nesse caso, seria justo que modelos avaliados por diferentes medidas de sucesso fossem projetados para otimizar diferentes métricas de desempenho [Yan et al., 2003; Ertekin & Rudin, 2011].

Finalmente, uma questão adicional para modelos de crédito baseados em comportamento é a complexidade da representação dos clientes. Em particular, o mercado de crédito no varejo, que é o foco desta pesquisa, movimenta grandes mercados consumidores em todo o mundo e registrou mais de 64 bilhões de transações em 2015 [Tran et al., 2016]. Apenas no Brasil, as compras de crédito aumentaram 14,5% e atingiram R$ 1,55 trilhão em 2018, com um crescimento estimado de 16%, totalizando R$ 1,8 trilhão em 2019 [Departamento Intersindical de Estatísticas e Estudos Socioeconômicos - DIEESE, 2018]. Esse grande volume de dados e os padrões complexos, típicos de modelos comportamentais usados no varejo, aumentam a dificuldade do processo de identificação do conjunto de atributos que melhor define os perfis de bons e maus pagadores [Neagoe et al., 2018]. Uma forma de resolver esse problema é usar o aprendizagem profunda para que o modelo, além de aprender como mapear representações de entrada para a saída, também aprenda as melhores representações de dados, através do reconhecimento de padrões em várias camadas de processamento [Goodfellow et al., 2016].

Neste trabalho, apresentamos uma abordagem profunda que utiliza funções de custo assimétricas para resolver problemas de *credit scoring* a partir da equivalência entre classificação binária e ranking bipartido. Neste caso, em vez de uma abordagem heurística de busca exaustiva de pesos, usamos uma estratégia de aprendizagem

alternada para encontrar os pesos ideais das funções de custo, durante as épocas de treinamento da rede neural. Apesar de aprendizagem profunda ter sido amplamente aplicada com sucesso em diversas áreas de pesquisa, poucos estudos utilizam arquiteturas profundas para resolver o problema de previsão de risco de crédito. Destes, até onde sabemos, nenhum trabalho propôs o uso de funções assimétricas como forma de lidar com o ranking de risco de crédito e seu desbalanceamento natural. Por isso, a fim de preencher essa lacuna na literatura, apresentamos neste trabalho um método profundo para aprendizagem ponderada assimétrica em arquiteturas profundas e duas novas funções de perda sensíveis ao custo.

1.1 Problema de pesquisa

Neste trabalho, estamos interessados em um problema de aprendizagem supervisionada envolvendo um espaço de instâncias $X \in \mathbb{R}^{n,m}$ e um espaço de rótulos $y \in \mathbb{R}^n$. Uma instância $X_i \in X$ corresponde a um solicitante de crédito cujo desempenho como pagador é resumido pelo rótulo y_i. Mais especificamente, e sem perda de generalidade, o valor $y_i = +1$ indica que ao longo de um certo período de tempo, ele se envolveu em algum evento de delinquência[2], sendo considerado, portanto, um mau pagador. O valor $y_i = -1$ indica, por sua vez, que ele é considerado um bom pagador. Cada instância X_i corresponde a um vetor de m características, em que cada característica $X_{i,m}$ é usada para descrever o $cliente_i$ (por exemplo, seu sexo, idade e renda declarada) e seu comportamento (por exemplo, o valor médio de pagamentos realizados em um certo período de tempo[3]).

Dado um conjunto de exemplos $\mathcal{D} = \{X_j, y_j\}_{j=1..N}$, em que $y \in \{-1, +1\}$ é o rótulo associado à entrada $X_i \in X$, o algoritmo de ranking bipartido deve encontrar uma função $f : X \to \mathbb{R}$ que ranqueia instâncias positivas com escore maior do que o de instâncias negativas. Esses algoritmos estão preocupados com os valores das pontuações em relação uns aos outros. Como resultado, otimizam as métricas de ranking, como AUC e KS. Os algoritmos de classificação binária, por outro lado, otimizam o erro de classificação incorreta, uma vez que eles estão preocupados com um limite de decisão que distingue claramente instâncias positivas das negativas.

Mais formalmente, considere que $\{X_1^+, X_2^+, \ldots, X_I^+\}$ é um conjunto de instâncias positivas e $\{X_1^-, X_2^-, \ldots, X_K^-\}$ é o conjunto de instâncias negativas extraído da distri-

[2]Um exemplo típico de evento de delinquência em problemas de crédito é, por exemplo, deixar de pagar uma, de qualquer uma das parcelas de sua dívida, por um período de tempo maior que d dias.

[3]No caso do comportamento, este é observado ao longo de um período de tempo anterior àquele em que foi feita a solicitação de crédito.

buição conjunta $X \times \{-1, +1\}$. Dada uma classe de funções convexas $\mathcal{F}$, sendo $\hat{y} \in \mathcal{F}$, $\hat{y} : X \times \Theta \to \mathbb{R}$, é uma função de pontuação que associa uma pontuação de valor real entre -1 e 1 a uma instância de X, parametrizada pelo conjunto de parâmetros Θ, sendo $\Theta \in \mathbb{R}^{m'}$. Esses m' parâmetros[4] são escolhidos para minimizar uma função objetivo ℓ.

Em ranking bipartido, a qualidade de classificação pode ser medida usando uma métrica associada ao erro de ranking, como o número de instâncias positivas que estão ranqueadas abaixo das negativas. Sem perda de generalidade, isto pode ser traduzido pela função de perda:

$$\ell_{ranking}(\hat{y}) = \sum_{i=1}^{I} \sum_{k=1}^{K} \mathbb{1}(\hat{y}(x_i^+; \Theta) \leq \hat{y}(x_k^-; \Theta)) \tag{1.1}$$

onde $\mathbb{1}(\gamma) = 1$ se o critério γ for satisfeito e 0, caso contrário. Por sua vez, o objetivo da classificação binária é distinguir corretamente as instâncias negativas das positivas. Seu desempenho pode ser medido pelo erro de classificação, dado por:

$$\ell_{classificacao}(\hat{y}) = \sum_{i=1}^{I} \mathbb{1}(\hat{y}(x_i^+; \Theta) \leq 0) + \sum_{k=1}^{K} \mathbb{1}(\hat{y}(x_k^-; \Theta) > 0) \tag{1.2}$$

Como essas funções de perda são difíceis de minimizar diretamente, algoritmos de ranking e classificação geralmente adotam funções de erros convexas que as aproximem. Neste trabalho, em particular, o problema que queremos resolver consiste em encontrar uma função de perda que aproxime simultaneamente as funções dadas nas equações $\ell_{ranking}$ e $\ell_{classificacao}$, adequada ao domínio de crédito.

1.2 Hipóteses e Questões de Pesquisa

Ao longo dos últimos anos, modelos baseados em aprendizagem profunda têm sido usados com sucesso em um grande número de aplicações de previsão [Khan et al., 2018; Yifei, 2016]. Seu sucesso pode ser atribuído à sua capacidade de aprender representações hierárquicas de atributos que se mostram adequadas em uma variedade de tarefas. Sua aplicação, contudo, em problemas de previsão de risco de crédito ainda é incipiente devido, em parte, ao tamanho reduzido das coleções de referência comumente usadas na literatura. Esta situação só começou a se modificar recentemente, com a disponibi-

[4]Em modelos de aprendizagem rasos, m' é tipicamente igual a m, ou seja, o modelo de fato pondera cada um dos atributos de entrada. Em modelos profundos, espera-se que m' seja (bem) maior que m, uma vez que o modelo deve aprender novos atributos para representar a entrada.

lidade de bases maiores e consequente introdução de modelos baseados em arquiteturas de redes neurais profundas.

Os trabalhos encontrados durante nossa revisão da literatura, baseados em aprendizagem profunda, não exploram aspectos inerentes ao problema, como sua natureza extremamente desbalanceada e a possibilidade de que o risco calculado seja base para políticas de crédito em que diferentes erros de previsão são tratados de formas distintas. Por exemplo, para uma certa política de crédito é possível que o tratamento como adimplente de um cliente inadimplente seja um erro mais grave que o oposto, se a certeza do modelo na adimplência do cliente é maior que um certo limiar de probabilidade.

Como observado, em paralelo, outros estudos na área de aprendizagem de máquina têm demonstrado a existência de equivalências entre problemas de classificação binária (como o problema de classificação de aplicantes como bons ou maus pagadores) e problemas de ranking bipartido (ranquear maus pagadores com risco maior que os bons) [Menon & Elkan, 2011; Ertekin & Rudin, 2011]. Essa equivalência, não estudada no contexto de aprendizagem profunda aplicada ao risco de crédito, sugere a possibilidade de que existam eficientes algoritmos de aprendizagem capazes de lidar diretamente com problemas de desbalanceamento e tratamento assimétrico de erros de classificação.

Estas observações nos levam a propor a hipótese de pesquisa que foi avaliada nesta tese:

> **Hipótese**: *Existe um modelo de previsão de risco de crédito baseado em uma arquitetura de rede neural profunda deve ser mais eficaz em aprender representações de candidatos, úteis para distingui-los entre bons e maus pagadores. Se o aprendizado de tal modelo for baseado em funções de custo assimétricas capazes de lidar com problemas de ranking bipartido usando estratégias similares a de um classificador, este aprendizado pode ser feito de forma eficiente e robusta a desbalanceamento.*

Com base nessa hipótese, surgiram as seguintes questões de pesquisa:

1. Qual é o impacto de técnicas de tratamento de desbalanceamento, tais como técnicas de *sampling*, em modelos profundos em geral e naqueles aplicados a *credit scoring* que sejam abordados por modelos rasos e profundos?

2. Os modelos profundos que usam funções de custo assimétricas são mais robustos a desbalanceamento e mais adequados a diferentes políticas de crédito do que modelos profundos que usam funções de custo simétricas?

3. Esses modelos são tão eficientes em termos de custo de treinamento e conveniência de parametrização quanto outras abordagens rasas ou profundas encontradas na literatura?

1.3 Objetivos

Nesta seção apresentamos os objetivos geral e específicos deste trabalho.

1.3.1 Objetivo geral

Nosso objetivo é desenvolver um modelo de aprendizagem profunda para prever o risco de crédito de clientes, a fim prover meios para diminuir a taxa de inadimplência de instituições que concedem crédito no varejo. Esse modelo deve ser capaz de lidar com o desbalanceamento inerente do problema e ser flexível para se adaptar a diferentes erros de classificação.

1.3.2 Objetivos específicos

O nosso objetivo geral se traduz nos seguintes objetivos específicos:

1. Caracterizar os modelos encontrados na literatura para *credit scoring* que utilizam técnicas de aprendizagem profunda.

2. Caracterizar técnicas para tratamento de desbalanceamento em modelos rasos e profundos adaptando, se necessário, o estado-da-arte das mesmas ao contexto de *credit scoring*.

3. Propor arquiteturas profundas para *credit scoring*, baseadas em função de custo assimétrica, capaz de lidar de forma equivalente com problemas de classificação binária e ranking bipartido, robusta a desbalanceamento, eficiente em termos de treino e conveniente em termos de parametrização.

1.4 Metodologia

A metodologia adotada consistiu no seguinte macro conjuto de atividades:

1. Levantamento bibliográfico;

2. Obtenção das coleções de dados para experimentação;

3. Seleção e implementação de métodos do estado-da-arte para comparação;

4. Implementação de novas funções de perda sensíveis ao custo;

5. Implementação de uma nova abordagem profunda para aprendizagem ponderada assimétrica;

6. Análise de resultados e conclusão;

A pesquisa foi concentrada em resolver o problema de *credit scoring* usando abordagens de aprendizagem profunda e assimétricas, sensíveis a diferentes custos, associados a diferentes tipos de erros. Uma vez definido o foco da pesquisa, a primeira fase do trabalho foi dedicada ao levantamento bibliográfico do estado-da-arte em *credit scoring* e funções assimétricas de perda. Como resultado, descobrimos poucos trabalhos que utilizam aprendizagem profunda aplicados ao problema *credit scoring*, dos quais, menos da metade utilizam alguma estratégia de pré-processamento para lidar com o desbalanceamento de classes.

A segunda fase do trabalho foi dedicada à identificação e coleta de bases de dados com características típicas do domínio de *credit scoring*, como alto grau de desbalanceamento de classes e a presença de dados cadastrais e de comportamento de pagamentos de clientes. Essas bases foram coletadas a partir de um grupo varejista e de uma competição pública na área de aprendizagem de máquina. Em ambos os casos, temos atributos e rótulo para cada instância de aplicante.

Na terceira fase do trabalho, implementamos métodos rasos, comumente utilizados para resolver o problema de *credit scoring*, a fim de estudar como esses modelos se comportam em domínios de alto grau de desbalanceamento. O desempenho de cada método foi avaliado com e sem estratégias de amostragem baseada em dados. Em seguida, implementamos os métodos que utilizam aprendizagem profunda para resolver o problema de *credit scoring*. Por fim, avaliamos o desempenho de funções assimétricas. Para isso, essas funções foram usadas como funções de perda em uma arquitetura canônica neural profunda.

Na quarta fase do trabalho, nos dedicamos à definição e implementação de funções assimétricas de perda sensíveis ao custo. Em nossas funções, penalizamos os erros e recompensamos os acertos do classificador. Para isso, definimos uma função que considera dois fatores de ponderação e outra que considera quatro fatores. Ao definirmos as funções tivemos a preocupação de que elas fossem simples e tirassem proveito do atual estado-da-arte em computação vetorial aplicada a modelos profundos de forma a termos funções eficientes e estáveis numericamente.

A quinta fase do trabalho foi dedicada à implementação de uma abordagem profunda para aprender os valores ótimos dos hiper-parâmetros das nossas funções de custo, durante o treinamento do modelo. Em nosso método, usamos a mesma estrutura canônica usada para avaliar as funções de perda da literatura, alterando apenas o algoritmo de aprendizagem para que os pesos das nossa funções fossem otimizados durante o treinamento.

A sexta e última fase pesquisa foi dedicada à análise dos resultados e conclusão, conforme resumido na seção a seguir.

1.5 Resultados e Contribuições

Neste trabalho conseguimos avançar o estado-da-arte em *credit scoring* e funções assimétricas de perda. Nossas principais contribuições foram:

1. Revisão da literatura de *credit scoring* com o uso de aprendizagem de máquina profunda.

2. Adaptação de técnica para lidar com desbalanceamento em classificação de objetos em redes de convolução para o contexto de *credit scoring* (cf. o método CoSen, descrito na Seção 3.1.3.1).

3. Proposta de funções de custo assimétricas em modelos profundos para resolver problemas de pontuação de crédito, onde as principais vantagens são: (1) criação de modelos naturalmente mais robustos a desbalanceamento de classes; (2) funções de custo que permitem que diferentes pesos sejam aplicados a diferentes pontos do ranking, o que torna o modelo mais flexível e aplicável a diferentes políticas de crédito; (3) tempo de treinamento competitivo, com custo linear por lote de exemplos de treinamento.

4. Estratégias de otimização que permitem o aprendizado automático de parâmetros de funções assimétricas, permitindo que os modelos aprendam a lidar melhor com o desbalanceamento inerente ao problema, sem custos adicionais de pré-processamento ou treinamento. A abordagem proposta foi testada em um conjunto de dados real, fornecido por um grande grupo varejista brasileiro, e em um conjunto de dados usado em um concurso público de aprendizagem de máquina, onde a nossa abordagem apresentou desempenho estatisticamente superior ao de modelos rasos e profundos.

O resultado da primeira contribuição está descrito no artigo "A Survey on Deep Learning for Credit Scoring", que foi submetido ao periódico *Journal of the Brazilian Computer Society* (B1) e se encontra na segunda fase de revisão. As duas outras contribuições estão descritas no artigo "Assymetrical Loss Functions to deal with Imbalance on Deep-learning based models applied to Credit Scoring", a ser submetido para o periódico *Expert Systems with Applications* (A1).

1.6 Estrutura da tese

Além deste capítulo introdutório, este trabalho está organizado como segue. No Capítulo 2, descrevemos os conceitos fundamentais para o entendimento deste trabalho, relacionados a teoria de *credit scoring* e à modelagem de abordagens profundas. No Capítulo 3, apresentamos métodos profundos que resolvem o problema de *credit scoring*, bem como as funções sensíveis a custo adotadas como baseline deste trabalho. No Capítulo 4, descrevemos as funções de perda e a estratégia de profunda para aprendizagem alterada propostas neste trabalho. No Capítulo 5, apresentamos o protocolo experimental adotado neste trabalho, bem como o conjunto de resultados alcançados. Por fim, no Capítulo 6, apresentamos as conclusões do nosso trabalho e sugestões de próximos passos na pesquisa.

Capítulo 2

Fundamentação teórica

A pesquisa desenvolvida neste trabalho está relacionada à aplicação de aprendizagem profunda à análise preditiva de risco de crédito, utilizando auto-balanceamento de dados e otimização de rankings de scores através de funções assimétricas. Neste capítulo, apresentamos conceitos introdutórios referentes a estas áreas.

2.1 Pontuação de crédito

Nos últimos anos, o setor de concessão de crédito ganhou grande destaque em instituições financeiras, passando a representar um percentual expressivo do faturamento dessas instituições. Trata-se de um mercado com crescimento global de mais de 231% entre os anos de 1970 e 2005. Somente no mercado norte americano, o volume de hipotecas aumentou 705%. Além disso, o crescimento do montante de crédito cedido a consumidores dos Estados Unidos registrou uma marca histórica de $1.132 bilhões no ano de 2013 [Baesens et al., 2003; Zhu et al., 2018].

A crescente demanda de consumidores por crédito representa um grande número de novas oportunidades para o mercado financeiro, além de trazer um volume quase proporcional de riscos para essas agências credoras. Este fato indica que essas instituições precisam adotar ferramentas, como a pontuação de crédito (mais conhecida pelo termo em inglês - *credit scoring*), capazes de diminuir o risco de empréstimos a terceiros [Lessmann et al., 2015].

Credit scoring é o processo de criação de modelos de avaliação de risco de crédito, capazes de apoiar ao processo de tomada de decisão de concessão de empréstimos a consumidores [Schreiner, 2000], onde podemos avaliar o risco de crédito como a probabilidade de inadimplência [Lin et al., 2012] ou como a distinção entre bons e maus pagadores [Crook et al., 2007]. Neste cenário, o uso de uma ferramenta de *credit*

scoring pode ajudar na gestão de políticas de relacionamento com o cliente [di Basilea per la vigilanza bancaria, 2004], criação de produtos customizados [SUPERVISION, 2010], precificação de novos negócios e gestão de ativos/passivos da empresa [Zhong et al., 2014].

Segundo Yeh & Lien [2009], as agências credoras lidam com a tarefa de previsão de risco de crédito utilizando todos os dados disponíveis, dividindo seus clientes em dois grupos: candidatos e recorrentes. Candidatos são indivíduos que ainda não possuem relacionamento com a instituição e, neste caso, os únicos dados disponíveis são dados cadastrais, como sexo, idade e renda declarada. Clientes recorrentes são aqueles que já possuem relacionamento com a instituição, permitindo que sejam usados, além dos dados cadastrais, dados relacionados ao seu comportamento, como quantidade de empréstimos realizados, atraso médio de parcelas e dívida acumulada. Na literatura, tanto candidatos quanto clientes recorrentes costumam ser chamados de aplicantes [Lessmann et al., 2015]. Nesse sentido, um bom modelo de previsão de risco de crédito deve ser capaz mapear os atributos de aplicantes em relação à sua probabilidade de inadimplência, a fim de evitar que a instituição credora perca dinheiro em transações ruins [Qian et al., 2014], pois a perda em contratos firmados com maus é maior que o ganho com empréstimos realizados para bons aplicantes [Yu et al., 2018].

Após a aprovação e liberação do crédito, o cliente entra no período de desempenho, onde são acompanhados os pagamentos das mensalidades do empréstimo. É neste período que o cliente é classificado como bom ou mau pagador, dependendo do seu comportamento de pagamento. Neste caso, bons pagadores são aqueles clientes que quitam seus débitos e geram lucro financeiro para a instituição credora. Por outro lado, maus pagadores são os clientes que se tornam inadimplentes, podendo gerar perda financeira para a agência credora [Kvammea et al., 2018].

Os primeiros métodos estatísticos para modelos de *credit scoring* foram desenvolvidos na década de 40, quando Durand [1941] aplicou os métodos de análise de discriminantes, propostos por Fisher [1936], para discriminar bons e maus empréstimos. Nesta época, foi definido que *credit scoring* seria o nome dado à tarefa de analisar o risco de crédito do cliente, a partir de um escore [Tomczak & Zieba, 2015]. Segundo descrito por Khashman [2011], essa tarefa foi dividida nas três categorias a seguir:

1. *Application Scoring* (AS) - modelo de pontuação baseado em dados cadastrais de cliente, utilizado para apoiar o processo de decisão de aceitação de propostas de empréstimos de novos clientes [Li et al., 2006];

2. *Behaviour Scoring* (BS) - modelo de pontuação baseado em dados transacionais de clientes, comumente utilizado nos processos de aprovação de novos contratos

e manutenção de limites de crédito de clientes recorrentes [Laha, 2007];

3. *Collection Scoring* (CS) - modelo de score baseado em dados transacionais de clientes que se tornaram inadimplentes, utilizado para definição e priorização de estratégias de investimento em cobranças [Vellido et al., 1999];

Nas seções a seguir, descrevemos os passos para criação de um modelo de *credit scoring*, comentando as peculiaridades de cada categoria, sempre que necessário.

2.1.1 Planejamento amostral

Em geral, o desenvolvimento de um modelo de *credit scoring* inicia com a definição do *período de desempenho*, que corresponde ao futuro, ou seja, representa o período de tempo para o qual desejamos fazer a previsão. Esse período se estende do momento da aprovação do crédito até o instante em que o cliente é classificado como bom ou mau pagador [Thomas et al., 2017]. O período de desempenho costuma variar entre 6 e 18 meses, sendo que é mais comum o uso de 12 meses para AS e 6 meses para BS e CS, como proposto por Lyn et al. [2002].

O tempo é uma característica tão importante na definição de modelos de crédito que, após a definição de futuro, seguimos para a definição de *passado*, que indica o período de tempo que será usado para observação dos padrões de comportamento de inadimplência, que esperamos que se repitam no futuro [Huang & Day, 2013]. Neste caso, o uso de períodos muito longos, como 24 ou 36 meses, podem prejudicar o desempenho das técnicas de previsão, uma vez que representam períodos muito distantes, onde a realidade econômica, por exemplo, pode ser completamente diferente dos dias atuais [Thomas et al., 2017]. É comum que modelos robustos adotem o uso dos últimos 12 meses como período de observação, por formarem um passado recente suficientemente grande para a análise e descoberta de padrões, com maior chance de repetição em um futuro próximo [Lyn et al., 2002].

No caso particular de técnicas de aprendizagem de máquina, diferentes estratégias de validação e avaliação podem ser adotadas de acordo com como a dimensão temporal é tratada. Normalmente, as instâncias são separadas em conjuntos de treino (onde os padrões são observados), de validação (onde os hiper-parâmetros usados são avaliados) e teste (onde o modelo como um todo é avaliado). Na validação (avaliação) *in-time*, o conjunto de validação (teste) é formado por instâncias que se referem ao mesmo período de tempo do treino. Na validação (avaliação) *out-of-time*, o conjunto de validação (teste) é formado por instâncias tomadas no futuro em relação ao treino. Alternativamente, o treino pode ser feito em janelas de dados sucessivas (deslizantes ou

não) com validação feita sempre em períodos subsequentes. Esta estratégia, chamada *"walk-forward"*, pode ser vista como uma validação e avaliação *out-of-time*. Em geral, de acordo com Stein [2007], tanto na literatura quanto na indústria, é comum a adoção de validação *in-time* com avaliação *out-of-time*, o que usamos nesta tese.

Em técnicas de previsão baseadas em métodos estatísticos, buscamos identificar a relação entre variáveis coletadas no passado e a variável alvo, coletada do futuro [Chen et al., 2016]. Com isso, podemos perceber que para a definição de modelos de crédito usaremos variáveis explanatórias coletadas no período de observação, a fim de prevermos a variável alvo, que será gerada no período de desempenho e será responsável pela classificação do cliente como bom ou mau pagador [Yu et al., 2018]. Para essa atividade, precisamos analisar e identificar as variáveis exploratórias que melhor definem o comportamento de um cliente, associando um peso a cada uma delas. Esse peso indica o grau de importância de cada variável para o modelo [He et al., 2018].

Após a definição dos critérios temporais, seguimos para definições referentes às amostras de clientes. Lewis [1992], ao considerar métodos estatísticos rasos como regressores lineares, afirma que amostras com menos de 1500 bons clientes e 1500 maus clientes podem ser insuficientes para criação de modelos robustos. Além disso, é importante que a base de dados seja grande o suficiente para que possa ser dividida em amostras de treino, validação e teste [Diniz & Louzada, 2013].

Outro aspecto referente às amostras de clientes diz respeito à distribuição de bons e maus pagadores presentes na base de treino. Lyn et al. [2002] afirmam que, em definições de modelos de crédito, bases de dados reais são desbalanceadas por natureza, onde é comum que a proporção seja de 9 bons clientes para 1 mau, sendo importante garantir que a amostra mais rara, a de maus pagadores, seja grande o suficiente para que permita a descoberta dos padrões de comportamento de clientes que ficaram inadimplentes. Contudo, os modelos de *credit scoring* propostos na literatura costumam ser criados a partir de amostras balanceadas, onde 50% da base de treinamento são compostos por bons pagadores e os outros 50% são compostos por maus pagadores. Neste cenário, uma abordagem comum para compor a base de dados é selecionar todos os n maus pagadores e escolher aleatoriamente n instâncias de bons pagadores [Lyn et al., 2002].

A Figura 2.1 apresenta um exemplo de seleção de amostras para treinar um modelo de crédito, onde selecionamos todos os 30 mil exemplos de maus pagadores e 30 mil amostras são escolhidas aleatoriamente, a partir das 597 mil amostras de bons pagadores.

Thomas et al. [2017] afirmam que, em casos de desbalanceamento de bases de treinamento, podemos escolher lidar com a distribuição real dos dados ou balancear a

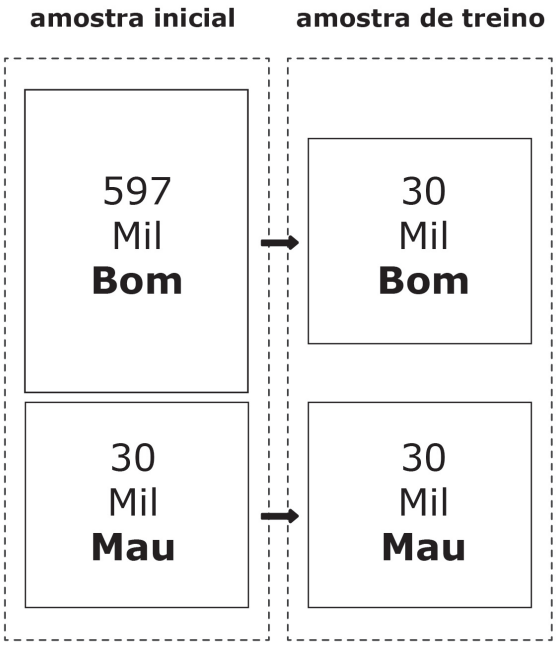

Figura 2.1: Balanceamento de amostra de treino

amostra usando estratégias de sobreamostragem da classe minoritária, subamostragem da classe majoritária ou uma combinação de ambas. *Undersampling* é a estratégia de diminuição da classe majoritária, onde são selecionadas, por amostragem simples, n instâncias da classe majoritária, onde n é o número total de instâncias da classe minoritária. Ao final do processo, a amostra resultante terá $2n$ instâncias [Chawla, 2009]. *Oversampling* é a estratégia de aumento da classe minoritária, onde há o aumento de n instâncias dessa classe. Nesse caso, n é igual à diferença entre o total de instâncias da classe majoritária e o total de amostras da classe minoritária. Ao final do processo, a amostra resultante terá $2m$ instâncias, onde m é o total de instâncias da classe majoritária [Batista et al., 2004]. Segundo [Thomas et al., 2017], a estratégia de balanceamento baseada em *oversampling* é a mais utilizada para o desenvolvimento de modelos de *credit scoring*.

Para completar as definições acerca da massa de dados envolvida no processo de desenvolvimento de modelos de crédito, costuma ser executado o processo de escolha das variáveis exploratórias que serão usadas para determinar os perfis dos clientes de crédito [Thomas et al., 2017]. Em Diniz & Louzada [2013], é apresentada uma lista com as variáveis cadastrais que são utilizadas em modelos de *credit scoring* com maior frequência. A Tabela 2.1 descreve alguns exemplos dessas variáveis, onde podemos observar a ocorrência de dados demográficos do aplicante, como Idade, Estado civil e Gênero, além de variáveis que representam seu comportamento de pagamento, como

endividamento médio mensal, média de pagamentos realizados e atraso médio no pagamento de parcelas.

Tabela 2.1: Variáveis comuns a modelos de pontuação de crédito.

VARIÁVEL	DESCRIÇÃO
Idade	Idade do cliente
Estado civil	Solteiro, casado, divorciado, viúvo
Gênero	Masculino ou feminino
Possui cartão de crédito?	Sim ou Não
Tipo de residência	Própria, emprestada, alugada
Tempo de residência	Tempo vivendo na mesma residência (em anos)
Tempo de emprego	Tempo trabalhando na mesma empresa (em meses)
Telefone comercial	O cliente declarou algum telefone comercial?
Tipo de correspondência	Residencial ou comercial ?
Renda comprometida	menos de 10%, entre 10 e 20%, mais de 20%
Limite de crédito	Valor de crédito aprovado
Profissão	Profissão do cliente
Dívida média mensal	Endividamento médio do cliente, por mês
Média de pagamentos	Valor médio pago pelo cliente
Atraso médio	Média de dias de atraso de pagamentos

Na seção a seguir, apresentamos as métricas adotadas para avaliação dos experimentos realizados em nossa pesquisa.

2.1.2 Métricas de avaliação dos modelos

A escolha de uma métrica apropriada para avaliação de modelos de aprendizagem de máquina é, geralmente, uma tarefa desafiadora, mas é particularmente difícil para problemas de classificação com dados desbalanceados, como é o caso de problemas de pontuação de crédito. Primeiro, porque a maioria das métricas amplamente utilizadas assume uma distribuição de classe balanceada e porque geralmente nem todas as classes e, portanto, nem todos os erros de previsão, são iguais para a classificação em um conjunto de dados desbalanceados [Sun et al., 2009].

Existem métricas que são amplamente usadas para avaliar modelos preditivos de classificação, como precisão ou erro de classificação, que funcionam bem na maioria dos problemas, e é por isso que são amplamente adotadas. Contudo, todas as métricas fazem suposições sobre o problema ou sobre o que é importante resolvê-lo. Portanto, uma métrica de avaliação deve ser escolhida para capturar melhor o que acreditamos ser importante sobre o resultado do modelo. Essa peculiaridade torna desafiadora a escolha das métricas de avaliação de modelo de aprendizagem de máquina. Esse desafio é ainda maior quando há um alto grau de desbalanceamento das classes. Isso acontece porque várias métricas se tornam não confiáveis ou mesmo enganosas quando as classes estão desequilibradas ou severamente desequilibradas, como a proporção 1:100 ou 1:1000 entre uma classe minoritária e a majoritária [He & Ma, 2013].

Ao contrário das métricas de avaliação padrão que tratam todas as classes como igualmente importantes, os problemas de classificação com dados desbalanceados geralmente atribuem maior importância aos erros da classe minoritária, do que aos erros da classe majoritária. Essa atividade é desafiadora porque é justamente a classe minoritária que sofre da falta de observações necessárias para o treinamento de um modelo eficaz. Além de desafiador, o acerto na classificação de instâncias dessa classe costuma ser importante para o domínio do problema, por geralmente estar associado a um viés relevante problema [Branco et al., 2016].

Dadas as diferenças entres os vários tipos de problemas, a literatura apresenta um grande número métricas distintas para avaliação do desempenho de modelos preditivos. Contudo, He & Ma [2013] afirmam que essas métricas podem ser agrupadas em três tipos fundamentais: métricas de limite, métricas de ranking e métricas de probabilidade, conforme descrito a seguir.

2.1.2.1 Métricas de limite

Métricas de limite são aquelas que quantificam os erros de previsão de classificação, sendo normalmente usadas quando queremos que um determinado modelo minimize o número de vezes em que a classe prevista não corresponde à classe esperada, em um dado conjunto de dados de validação. Uma das métricas de limite mais usadas é a precisão (acurácia) da classificação:

$$Acuracia = \frac{VP + VN}{N} \tag{2.1}$$

onde:

- VP = Verdadeiro Positivo: é o total de inadimplentes corretamente classificados;

- VN = Verdadeiro Negativo: é o total de bons pagadores corretamente classificados;
- N: é o número total de predições realizadas;

E a métrica que complementa a acurácia é o erro de classificação:

$$Erro = \frac{FP + FN}{N} \tag{2.2}$$

onde:

- FP = Falso Positivo: é o total de inadimplentes incorretamente classificados como bons pagadores;
- FN = Falso Negativo: é o total de bons pagadores incorretamente classificados como inadimplentes;
- N: é o número total de predições realizadas;

Apesar de amplamente utilizada, a acurácia é uma métrica considerada inadequada para avaliação de modelos de classificação treinados em bases de dados desbalanceadas, porque o modelo pode atingir uma acurácia alta, mesmo sem ter grande habilidade em classificar a classe minoritária. Em problemas como pontuação de crédito, onde normalmente há desbalanceamento de dados, é comum que a classe majoritária seja marcada como negativa (resultado negativo do teste) e que a classe minoritária seja marcada como positiva (resultado positivo do teste) [He & Ma, 2013]. Com base no exposto, adotamos a seguinte convenção de rótulos neste trabalho:

- Classe majoritária: Saída negativa, classe = 0;
- Classe minoritária: Saída positiva, classe = 1;

A matriz de confusão é uma estratégia que costuma ser usada para facilitar o entendimento de métricas de limite, ajudando no entendimento do que está sendo medido. A matriz de confusão oferece informações sobre quais classes estão sendo previstas corretamente, quais não estão e que tipo de erros estão sendo cometidos pelo modelo. Cada célula da matriz possui papel bem definido, como exibido na Tabela 2.2, onde apresentamos as relações entre classes reais e classes previstas.

Quando avaliamos classificação desbalanceada, existem dois grupos de métricas que podem ser úteis porque se concentram na avaliação de apenas uma das classes: são as métricas de sensibilidade-especificidade e precisão-revocação, como descrito a seguir.

Tabela 2.2: Matriz de confusão

		Classe prevista	
		Positiva	**Negativa**
Classe Real	**Positiva**	Verdadeiro Positivo (VP)	Falso Negativo (FN)
	Negativa	Falso Positivo (FP)	Verdadeiro Negativo (VN)

A métrica de sensibilidade, também chamada de revocação, avalia o quão boa é a classificação de instâncias da classe positiva:

$$\text{Sensibilidade} = \text{Revocação} = \frac{VP}{VP + FN} \tag{2.3}$$

A métrica de especificidade é o complemento da sensibilidade, avaliando o comportamento de classificação de instâncias da classe negativa:

$$\text{Especificidade} = \frac{VN}{FP + VN} \tag{2.4}$$

A métrica de precisão avalia as instâncias positivas classificadas corretamente:

$$\text{Precisão} = \frac{VP}{VP + FP} \tag{2.5}$$

A combinação das métricas de precisão e revocação gera uma pontuação que busca equilibrar os dois conceitos, denominada F_1-score:

$$\text{F}_1\text{-score} = \frac{2 \times \text{Precisão} \times \text{Revocação}}{\text{Precisão} + \text{Revocação}} \tag{2.6}$$

Por fim, outras métricas de limite encontradas usadas na literatura para avaliação de modelos de pontuação de crédito são *Miss Alarm Rate* (MAR) e *False Alarm Rate* (FAR), formalizadas nas equações:

$$MAR = \frac{FN}{FN + VP} * N \tag{2.7}$$

$$FAR = \frac{FP}{FP + VN} * N \tag{2.8}$$

2.1.2.2 Métricas de ranking

Métrica de ranking são utilizadas para avaliarmos o desempenho de modelos de classificação com base na sua eficácia na separação de classes do problema. Esse tipo de métrica é importante em problemas como o de pontuação de crédito, onde é importante

que o score de uma determinada classe seja superior ao de outra. Essas métricas exigem que um classificador preveja uma pontuação ou uma probabilidade de associação à classe. A partir dessa pontuação, diferentes limiares podem ser aplicados para testar a eficácia dos classificadores. Os modelos que mantêm uma boa pontuação em vários limites terão boa separação de classe e alcançarão os melhores resultados de ranking. [Fernández et al., 2018].

Uma métrica de ranking comumente utilizada em problemas de classificação binária é a curva ROC (do inglês *Receiver Operating Characteristic*), que avalia classificadores binários com base em sua capacidade de discriminar classes. A curva ROC é um gráfico de diagnóstico para resumir o comportamento de um modelo, calculando a taxa de falsos positivos e a taxa de verdadeiros positivos para um conjunto de previsões do modelo, sob diferentes limites. Nesse caso, a taxa de verdadeiro positivo é calculada como a métrica de sensibilidade:

$$TaxaVerdadeiroPositivo = \frac{VP}{VP + FN} \tag{2.9}$$

Já a taxa de falso positivo, por sua vez, é calculada como:

$$TaxaFalsoPositivo = \frac{FP}{FP + VN} \tag{2.10}$$

Na curva ROC, cada limite é um ponto no gráfico e os pontos são conectados para formar uma curva. Um classificador com baixo poder de classificação será representado pela curva que forma uma linha diagonal da parte inferior esquerda para a parte superior direita. Um modelo perfeito terá uma curva que passa pelo canto superior esquerdo do gráfico. Um modelo com bom poder de classificação terá uma curva entre as duas anteriores, conforme exibido na Figura 2.2:

A área sob a curva ROC (do inglês, Area Under the Curve - AUC) pode ser calculada para resumir o comportamento apresentado no gráfico, servindo como medida de comparação entre diferentes classificadores, onde um classificador arbitrariamente ruim terá $AUC \leq 0.50$, um classificador perfeito terá $AUC = 1.0$ e um bom classificador terá $0.50 < AUC < 1.0$. A AUC de um modelo pode ser calculada por:

$$AUC = \frac{1}{mn} \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbb{1}_{pi>pj} \tag{2.11}$$

onde i itera sobre todos os m pontos de gráfico com o rótulo verdadeiro igual a 1 enquanto j, sobre todos os n pontos com o rótulo verdadeiro igual a 0. As variáveis pi e pj indicam a probabilidade atribuída pelo classificador ao ponto i e j, respectivamente. A função $\mathbb{1}$ é a função indicadora que gera 1 se, e somente se, a condição $pi > pj$ for

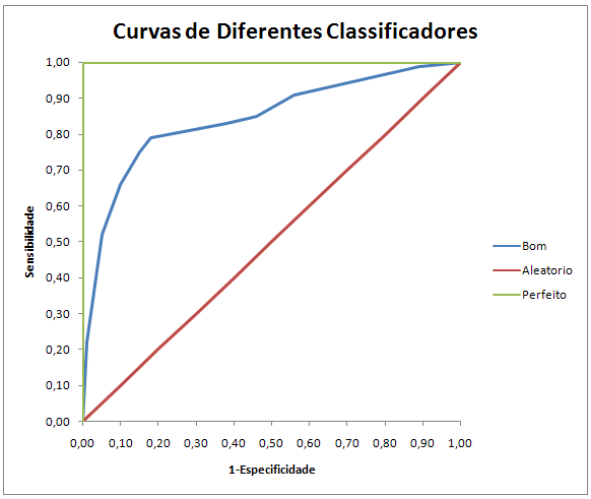

Figura 2.2: Curvas de diferentes classificadores

satisfeita [Fernández et al., 2018].

Outra métrica de ranking utilizada na avaliação de modelos de pontuação de crédito é a estatística *Kolmogorov-Smirnov* (KS), por medir a capacidade que o modelo tem de separar bons de maus pagadores [Razali et al., 2011]. Por definição, KS quantifica a distância entre as funções de distribuições empíricas que classificam clientes. Neste caso, considerando

$$F_{\text{bom}}(e) = \sum_{x \leq e} F_{\text{bom}}(x) \tag{2.12}$$

e

$$F_{\text{mau}}(e) = \sum_{x \leq e} F_{\text{mau}}(x) \tag{2.13}$$

as funções de distribuição empíricas para bons e maus pagadores, correspondendo às proporções de clientes bons e maus com pontuação menor ou igual a e, a métrica KS é dada pela distância máxima entre $F_{bom}(e)$ e $F_{mau}(e)$:

$$KS = max|F_{bom}(e) - F_{mau}(e)| \tag{2.14}$$

É comum, no mercado de crédito, que heurísticas baseadas no KS sejam adotadas como forma geral de avaliação de modelos. Em particular, de acordo com o problema a ser resolvido, os seguintes critérios são adotados:

- *Application scoring*: bons modelos devem apresentar $KS >= 20\%$;
- *Behavior* e *Collection scoring*: bons modelos devem ter KS maior ou igual a 40%;

2.1.2.3 Métricas de probabilidade

As métricas de probabilidade são projetadas especificamente para quantificar a incerteza nas previsões de um classificador. Isso é útil para problemas em que estamos menos interessados em previsões de classe incorretas versus corretas e mais interessados na incerteza que o modelo tem nas previsões e em penalizar as previsões erradas, mas altamente confiantes. Essas métricas são especialmente úteis quando queremos uma avaliação da confiabilidade dos classificadores, não apenas medindo quando eles falham, mas se eles selecionaram a classe errada com uma probabilidade alta ou baixa [Ferri et al., 2009].

Em nossa revisão bibliográfica, não encontramos trabalhos que utilizassem métricas probabilísticas para avaliação de modelos de pontuação de crédito.

2.2 Métodos estatísticos para Pontuação de Crédito

O primeiro trabalho registrado na literatura para resolver o problemas de pontuação de crédito é da década de 40, quando Durand [1941] usou a Análise de Discriminante Linear (do Inglês, *Linear Discriminant Analysis* - LDA), proposta por Fisher [1936] como método para gerar pontuações de risco de inadimplência de clientes.

Com o passar do tempo, muitos outros métodos estatísticos foram desenvolvidos e aplicados à análise de risco de crédito, como Integer Programming [Mangasarian, 1965]; Logit Regression [Wiginton, 1980; Hwang et al., 2009; Grunert et al., 2005; Martin, 1977; Kaplan & Urwitz, 1979]; Logistic Regression (LR)[Horrigan, 1966; Pogue & Soldofsky, 1969; West, 1970; Hand & Kelly, 2002]; Bayes Classifier [Hand & Henley, 1997]; and Nearest Neighbor [Thomas, 2000]. Dentre esses métodos, as abordagens LDA e LR foram as mais utilizadas e documentadas na literatura [Diniz & Louzada, 2013; Baesens et al., 2003].

Os métodos estatísticos citados acima, permitiram que os pesquisadores analisassem o efeito das variáveis exploratórias, que poderiam ser categóricas ou numéricas, sobre a variável binária de resposta do modelo [Baesens et al., 2003].

O constante interesse em melhorar o desempenho de modelos de avaliação de risco de crédito, aliado aos avanços tecnológicos em hardware e software, alavancaram o desenvolvimento de diversos modelos que utilizavam aprendizagem de máquina para resolver os problemas de pontuação de crédito [Saberi et al., 2013]. Esses modelos serão apresentados na seção a seguir.

2.3 Modelos de pontuação de crédito baseados em aprendizagem de máquina

Aprendizagem de Máquina (*Machine Learning* – ML) é uma sub-área da Inteligência Artificial que permite automatizar a construção de modelos analíticos, onde a ideia central é a de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana [Nasrabadi, 2007]. Quando comparados com os métodos estatísticos citados na seção anterior, as técnicas de ML provaram ser mais eficazes, principalmente após o crescimento do volume de transações e o aumento da complexidade da avaliação de risco de crédito [Yu et al., 2018].

Acompanhando as tendências de evolução tecnológica e crescimento do mercado de cessão de crédito, muitos estudos propuseram métodos de ML para resolver o problema de pontuação de crédito. Dentre esses métodos, podemos citar *Multivariate Discriminant Analysis* (MDA) [Altman, 1968; Michel, 1977; Shi et al., 2002]; *Decision Trees* (DT) [Han et al., 2008; Matthews, 1996]; *K-Nearest Neighbour* (KNN) [Henley & Hand, 1996; Li, 2009]; *Support Vector Machine* (SVM)[Bellotti & Crook, 2009; Huang et al., 2007, 2004; Kim & Ahn, 2012; Chen & Li, 2014]; e *Neural Networks* (NN) [West, 2000; Tsai & Wu, 2008; Atiya, 2001; Khashman, 2010]. Além da execução individual de cada um desses métodos, é possível combinarmos duas ou mais dessas técnicas, a fim de aumentar a qualidade de previsão dos modelos, construindo modelos híbridos [Hwang et al., 2009; Lin, 2009; Zhou et al., 2010; Chen & Cheng, 2013; Yeh et al., 2012].

O uso de métodos de ML para resolver o problema de pontuação de crédito costuma ser definido como uma tarefa clássica de classificação, onde estamos interessados em classificar um determinado cliente como bom ou mau pagador, dado um conjunto de variáveis que representam o perfil desse cliente [Wang et al., 2015]. Contudo, alguns gestores financeiros apontam que é mais importante gerar a probabilidade de inadimplência de um cliente, do que simplesmente rotulá-lo como mau, uma vez que essa probabilidade pode ajudar a nortear as políticas de crédito das instituições credoras [Yeh & Lien, 2009]. A aplicação desses métodos em problemas de crédito aumentou a produtividade do processo de avaliação e a capacidade de predição dos modelos, em comparação com as técnicas citadas na seção anterior [Huang & Day, 2013]. Neste caso, redes neurais (NN) e máquinas de vetor de suporte (SVM) foram as técnicas mais estudadas e aplicadas com sucesso em problemas reais, apresentando os melhores resultados em termos de AUC e KS [Yu et al., 2018].

Nas seções a seguir, apresentamos os principais conceitos e arquiteturas que su-

portam a modelagem baseada em SVM e NN.

2.3.1 *Suport Vector Machine* (SVM)

Suport Vector Machine (SVM) é um algoritmo de aprendizado supervisionado, introduzido pela primeira vez em 1992, que aplica a técnica de aprendizagem de máquina para maximizar a precisão do modelo e, ao mesmo tempo, emprega princípios matemáticos sofisticados para evitar *overfitting* [Boser et al., 1992]. Segundo Li et al. [2015], o ajuste e diminuição de *overfitting* do SVM ocorrem a partir da simples configuração de parâmetros da sua função de custo.

O SVM tornou-se famoso e muito usado para resolver problemas de classificação e previsão, apresentando bom desempenho nas métricas de avaliação, tanto em modelos simples, quanto em modelos de grande complexidade, que envolvem um grande conjunto de dados de entrada. Apesar de SVM resolver dois dos problemas clássicos em ML, Luo et al. [2017] afirmam que esse método é mais utilizado para solucionar problemas de classificação.

Em SVM, cada instância de dados é projetada em um espaço n-dimensional, onde n é o número de atributos usados para descrever a instância, com o valor de cada atributo sendo o valor de uma determinada coordenada desse espaço. Neste caso, a classificação é realizada a partir da descoberta do hiperplano que melhor separa as duas ou mais classes do problema [Cortes & Vapnik, 1995]. Apesar de sua robustez e desempenho, SVM apresenta algumas desvantagens (também presente em outros métodos clássicos de classificação), que podem ser impactantes para o contexto de problemas de pontuação de crédito [Yu et al., 2018]:

- O seu tempo de treinamento cresce em função do número de instâncias utilizadas, o que faz com que bases de dados muito grandes demandem elevado tempo de treinamento;

- O desempenho do modelo é afetado por bases de dados que apresentem alto índice de desbalanceamento, onde o SVM apresenta baixo desempenho preditivo para as classes minoritárias.

2.3.2 Redes Neurais

Redes Neurais, muito conhecidas pelo termo em inglês *Neural Networks* (NN) ou *Artificial Neural Networks* (ANN), são modelos computacionais inspirados pela forma como as redes neurais biológicas processam informações no cérebro humano. Nos últimos

anos, os resultados alcançados por modelos neurais têm despertado o entusiasmo de pesquisadores e alavancado o investimento de empresas, devido ao avanço gerado em áreas como reconhecimento de fala, visão computacional e processamento de textos [Wang et al., 2015]. Arquiteturas neurais permitem a criação de modelos preditivos com alto poder de abstração, capazes de lidar com problemas complexos, que envolvem um grande número de variáveis, como é o caso da pontuação de crédito.

A forma como os neurônios estão conectados define a arquitetura de uma NN. A Figura 2.3 mostra a arquitetura típica de uma ANN, uma rede *Multi-Layer Perceptron* (MLP), muito usada em problemas de análise de risco de crédito [West, 2000]. Nesta rede, os neurônios são organizados em camadas consecutivas, e as conexões entre as unidades são ponderadas por valores reais, denominados *pesos*. A camada que recebe os dados é chamada de *camada de entrada*. Na Figura 2.3, os neurônios na camada de entrada representam os atributos usados para descrever um aplicante: Idade, Sexo, Estado civil e Renda familiar. A última camada é chamada de *camada de saída* e representa o resultado final do processamento da rede, neste caso, se o aplicante é bom ou mau pagador. Entre as camadas de entrada e saída, pode haver uma sequência de L camadas, conhecidas como *camadas ocultas*.

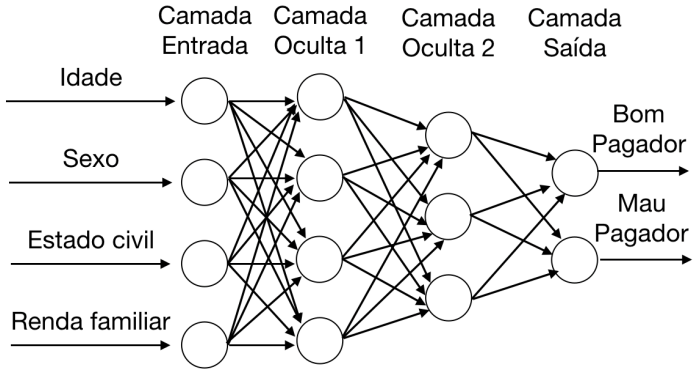

Figura 2.3: Arquitetura comum de uma MLP para problemas de pontuação de crédito

Para entender o funcionamento de uma rede neural, considere que cada neurônio de uma camada oculta recebe um vetor $\mathbf{x} = (x_1, x_2, ..., x_N)$ como entrada e realiza uma média ponderada pelos elementos de outro vetor de pesos associado entre essas camadas $\mathbf{w} = (w_1, w_2, ..., w_N)$. O valor que é gerado por essa computação, conhecida como *pré-ativação* do neurônio, é representado da seguinte forma:

$$a(\mathbf{x}) = b + \sum_i w_i x_i = b + \mathbf{w}^T \mathbf{x} \tag{2.15}$$

onde $a(\mathbf{x})$ resulta em um número escalar e b é uma parcela adicional correspondente

ao viés (*bias*), com peso constante.

Após calcular $a(\mathbf{x})$, o neurônio aplica uma transformação não-linear sobre esse valor por meio de uma *função de ativação*, $g(\cdot)$, da seguinte forma:

$$y = h(\mathbf{x}) = g(a(\mathbf{x})) \tag{2.16}$$

É importante ressaltar que diferentes funções de ativação resultam em diferentes comportamentos do neurônio. As funções mais comumente utilizadas são: sigmoide logística (Equação 2.17), sigmoide hiperbólica (Equação 2.18) e a função retificadora (Equação 2.19). Essas funções são normalmente utilizadas por serem todas diferenciáveis, fazendo a derivação da equação de atualização mais fácil (com exceção da ReLU em $z = 0$) e por serem funções estritamente crescentes. Além disso, uma característica importante das funções sigmoides é que elas mantêm a saída sempre entre 0 e 1.

$$g(z) = \frac{1}{1 + e^{-z}} \quad (2.17) \qquad g(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \quad (2.18) \qquad g(z) = max(0, z) \quad (2.19)$$

Tanto a função de pré-ativação quanto as funções de ativação de um único neurônio podem ser estendidas para redes que contenham L camadas ocultas, com vários neurônios em cada camada. A Equação 2.20 mostra a função de pré-ativação considerando toda uma camada oculta, onde o valor de k denota a k-ésima camada intermediária, $\mathbf{W}^{(k)}$ é a matriz de pesos das conexões entre os neurônios da camada $k-1$ e k, e o uso do negrito indica que a Equação 2.15 está sendo aplicada a cada neurônio dessa camada. O mesmo ocorre para a Equação 2.21 da função de ativação.

$$\mathbf{a}^{(k)}(\mathbf{x}) = \mathbf{b}^{(k)} + \mathbf{W}^{(k-1)}(\mathbf{x}) \tag{2.20}$$

$$\mathbf{h}^{(k)}(\mathbf{x}) = g(\mathbf{a}^{(k)}(\mathbf{x})) \tag{2.21}$$

De forma geral, considerando um vetor $\mathbf{x}$ de entrada, uma rede neural computa uma função $f(\mathbf{x})$ tal que $f(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^C$, de acordo com a Equação 2.22.

$$f(\mathbf{x}) = \mathbf{h}^{L+1}(\mathbf{x}) = \mathbf{o}(\mathbf{a}^{L+1}(\mathbf{x})) \tag{2.22}$$

A função $\mathbf{o}(\cdot)$ denota a função de ativação utilizada nas unidades da camada de saída. Quando o propósito de uma ANN é realizar a tarefa de classificação, normalmente seleciona-se para $\mathbf{o}(\cdot)$ a função *softmax*, uma vez que possibilita que os valores da saída sejam interpretados como probabilidades posteriores, pois geram valores em

um intervalo entre 0 e 1 [Goodfellow et al., 2016]. A Equação 2.23 mostra o valor produzido pela *softmax* para o i-ésimo neurônio da camada de saída.

$$o(a_i^{L+1}) = \frac{e^{a_i^{L+1}}}{\sum_{j=1}^{C} e^{a_j^{L+1}}} \tag{2.23}$$

2.3.2.1 Treinando uma Rede Neural

Na fase de treinamento de um processo de aprendizagem supervisionada é utilizado um conjunto de dados da forma $\{(\mathbf{x}^{(t)}, \mathbf{y}^{(t)}) : 1 \leq t \leq T\}$, onde $\mathbf{x}^{(t)}$ é o vetor de características associado ao rótulo $\mathbf{y}^{(t)}$. Em redes neurais, a etapa de treinamento consiste em utilizar tal conjunto de dados com o propósito de ajustar os parâmetros da rede, de tal forma que o erro de treinamento seja minimizado.

Após cada entrada $\mathbf{x}^{(t)}$ ser aplicada à rede, a saída gerada é comparada com a saída desejada $\mathbf{y}^{(t)}$. O algoritmo deve ajustar os parâmetros da rede (pesos e limiares), com o objetivo de minimizar a função de custo, que mede a diferença que a rede comete relativamente ao seu rótulo $\mathbf{y}^{(t)}$. A Equação 2.24 mostra o custo relativo ao conjunto de treinamento, onde $J(f(\mathbf{x}^{(t)}; \theta), \mathbf{y}^{(t)})$ é a função de custo que se deseja minimizar e θ representa o vetor de parâmetros da ANN. Existem algumas funções de custo que podem ser usadas para medir o erro associado ao treinamento, como por exemplo a *soma dos erros quadrados* e a *entropia cruzada*.

$$\mathbf{J}(\theta) = \frac{1}{T} \sum_{t=1}^{T} J(f(\mathbf{x}^{(t)}; \theta), \mathbf{y}^{(t)}) + \frac{\lambda}{2T} \sum_{k} \sum_{l} W_{k,l}^2 \tag{2.24}$$

Na Equação 2.24, a segunda parcela equivale ao custo de regularização contendo a soma dos quadrados de todos os pesos da NN. O componente λ é outro hiperparâmetro da rede que controla a importância relativa entre as duas parcelas da função $\mathbf{J}(\theta)$. Considerando que o propósito do treinamento é minimizar $\mathbf{J}(\theta)$, não são desejados valores elevados de pesos, uma vez que poderá aumentar o valor final, ao invés de diminuir. Além disso, se o vetor de pesos ficar muito grande, o método de otimização que realiza pequenas alterações no vetor de pesos a cada iteração, não será capaz de alterá-lo significativamente, o que pode levar o modelo a convergir para uma solução inadequada [Goodfellow et al., 2016].

Outro problema que uma rede neural pode sofrer no processo de treinamento é o fenômeno conhecido como sobreajuste (*overfitting*). Quando o conjunto de treino disponível não é grande o suficiente, a rede pode memorizar todos os exemplos da entrada, fazendo com que o modelo gerado não tenha uma boa capacidade de generali-

zação. Uma forma de evitar esse tipo de problema é utilizar técnicas de regularização. Na Equação 2.24, a segunda parcela é uma das possíveis formas de evitar que o processo de otimização fique preso a uma solução sub-ótima.

2.3.2.2 Gradiente Descendente

Qualquer algoritmo de otimização pode ser usado para o treinamento de uma ANN. Entretanto, o *gradiente descendente* (*gradient descent*, GD) é um dos algoritmos mais populares para solucionar problemas de otimização e é a maneira mais comum de otimizar redes neurais. Basicamente, é uma forma de minimizar a função de custo J, atualizando os parâmetros na direção oposta do gradiente da função de custo (gradiente negativo da função) $-\nabla J$. A ideia desse algoritmo é realizar, de forma iterativa, pequenas modificações no vetor de parâmetros θ com o intuito de levar esse vetor na direção da maior descida em uma superfície multidimensional representada por J.

Considere um vetor de pesos $w^{(t)}$. A Equação 2.25 mostra a atualização desse vetor usando a informação do gradiente a cada iteração. O parâmetro $\eta > 0$ é conhecido como taxa de aprendizado (*learning rate*). Após cada atualização, o gradiente é reavaliado para o novo vetor de pesos e o processo é repetido. Observe que a função de erro é definida em relação ao conjunto de treinamento e cada passo requer que todo o conjunto de treino seja processado para que o $\nabla J(w^{(t)})$ seja avaliado. Técnicas que usam todo o conjunto de dados de uma só vez são conhecidas como *batch*. No entanto, existe uma variação desse algoritmo em que a atualização do vetor de pesos é feita com base em apenas um dado, chamado de gradiente descendente estocástico (*stochastic gradient descent*) [Goodfellow et al., 2016].

$$w^{(t+1)} = w^{(t)} - \eta \nabla J(w^{(t)}) \tag{2.25}$$

2.3.2.3 Retropropagação do Erro

Um aspecto importante na Equação 2.25 é encontrar uma forma eficiente de avaliar o gradiente da função de erro $J(w)$. Isso pode ser alcançado por um esquema em que a informação calculada é enviada alternativamente para frente e para trás através da rede. Esse algoritmo é conhecido como retropropagação do erro (*backpropagation error*).

Nesse algoritmo, se a saída produzida pela rede para uma determinada entrada $\mathbf{x}^{(t)}$ é diferente da desejada, então é necessário determinar o grau de responsabilidade de cada parâmetro da rede nesse erro para, em seguida, alterar esses parâmetros com

o objetivo de reduzir o erro produzido. O algoritmo de retropropagação é realizado como segue.

1. Propague o vetor de entrada através de todas as camadas da rede, usando as funções de pré-ativação (Equação 2.20) e ativação (Equação 2.21), com o propósito de obter a saída de cada unidade.

2. Calcule o erro $\delta_i^{(L+1)}$ para as unidades de saída:

$$\delta_i^{(L+1)} = \frac{\partial J_n}{\partial y_i^{(L+1)}} f'(z_i^{(L+1)}) \tag{2.26}$$

3. Determine $\delta_i^{(l)}$ para todas as camadas escondidas l usando a retropropagação do erro:

$$\delta_i^{(l)} = f'(z_i^{(l)}) \sum_{k=1}^{m^{(i+1)}} w_{i,k}^{(l+1)} \delta_k^{(l+1)} \tag{2.27}$$

4. Calcule as derivadas requeridas:

$$\frac{\partial J_n}{\partial w_{j,i}^{(l)}} = \delta_j^{(l)} y_i^{(l-1)} \tag{2.28}$$

5. Atualize os pesos (Equação 2.25).

O algoritmo de retropropagação é uma aplicação da regra da cadeia para computar os gradientes de cada camada de uma ANN de forma iterativa. Detalhes do processo de derivação das funções podem ser verificados em Goodfellow et al. [2016].

2.3.3 Aprendizagem Profunda

Aprendizagem profunda se refere a um conjunto de técnicas de aprendizado de representação, onde novos atributos são aprendidos a partir de outros comumente organizados em camadas hierárquicas. Dada a natureza de redes neurais, estas são comumente usadas no contexto de aprendizagem profunda, de forma que a área quase se confunde com a ideia de modernas arquiteturas de redes neurais. Modelos de aprendizagem profunda tem consistentemente se tornado o estado-da-arte em áreas de pesquisas que lidam naturalmente com dados não estruturados, onde é necessário transformar os dados originais em covariáveis representativas, que servem de entrada para os modelos [Kvamme et al., 2018].

Diferente do processo manual de extração de covariáveis, comum em métodos clássicos de ML, redes neurais profundas aprendem múltiplos níveis de abstração de dados, a partir do processamento sequencial das características de entrada. Isso permite que arquiteturas profundas encontrem boas variáveis explicativas de forma automática [Kvamme et al., 2018]. Com isso, modelos de aprendizagem profunda são capazes de extrair características estruturais de alta dimensão, embutidas nos dados, através de sua estrutura profunda e alta capacidade de aprendizado [Yu et al., 2018]. Esse alto poder de aprendizagem e extração automática de características fez com que arquiteturas neurais profundas fossem aplicadas com sucesso em áreas como modelagem acústica [Mohamed et al., 2012], visão computacional [Krizhevsky et al., 2012], reconhecimento de emoções [Kim et al., 2013; Le & Provost, 2013], classificação de estado de saúde [Tamilselvan & Wang, 2013], processamento de fala e linguagem [Mohamed et al., 2012] e, mais recentemente, previsões de risco de crédito [Ribeiro & Lopes, 2011].

Nas seções a seguir, apresentaremos as principais arquiteturas neurais profundas utilizadas em modelos de pontuação de crédito.

2.3.3.1 *Deep Feedforward Network* (DFN)

DFNs correspondem aos *multi-layer perceptrons* descritos na Seção 2.3.2, com pelo menos três camadas ocultas. Modelos com até duas camadas ocultas são, em geral, similares a *ensembles* de modelos rasos e, portanto, não considerados profundos. Assim, DFNs têm tipicamente três ou mais camadas ocultas. Os neurônios de uma camada oculta L representam novas características, aprendidas como combinações não-lineares de características advindas da camada oculta $L-1$. DFNs são chamadas de *feedforward* porque as informações só seguem adiante na rede neural, através dos nós de entrada, passando pelas camadas ocultas e, finalmente, pelos nós de saída, sem conexões de *feedback* para que a saída da rede seja realimentada. Essas redes podem ser pensadas como a combinação de muitos modelos mais simples [Addo et al., 2018].

2.3.3.2 *Convolutional neural network* (ConvNet ou CNN)

CNNs são redes neurais criadas para explorar correlações espaciais locais, inspiradas no funcionamento do córtex visual de mamíferos. Como tal, elas são aplicadas com sucesso em áreas como reconhecimento e classificação de imagens e visão computacional [LeCun et al., 1998]. Ao longo tempo, estes modelos passaram a ser usados em qualquer aplicação em que há interesse em correlação local espaço-temporal, incluindo pontuação de crédito [Zhu et al., 2018]. De forma geral, uma CNN substitui a correlação global entre neurônios de camadas sucessivas de uma MLP por um processo de convolução

compartilhado, o que reduz drasticamente o número de parâmetros usados pelo modelo, ao mesmo tempo que enfatiza correlações locais [Zhu et al., 2018].

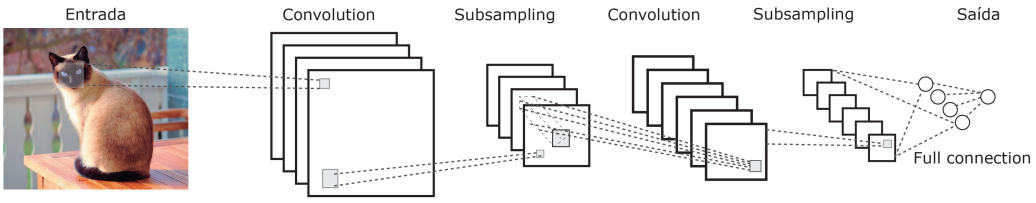

Figura 2.4: Arquitetura padrão de uma CNN.

A Figura 2.4 mostra a arquitetura base de uma CNN, dividida nas seguintes camadas: (1) entrada, que representa a instância a ser classificada, por exemplo, uma imagem ou qualquer instância cuja representação apresente correlação local; (2) camada de convolução (*Convolution*) que representa o bloco de construção principal da CNN, extraindo características a partir de um conjunto de filtros aprendidos ao longo do treino; (3) camada de *Subsampling* ou *Pooling* responsável por reduzir a dimensão da representação, o que melhora a eficiência do modelo além de melhorar a sua capacidade de generalização; e (4) camada totalmente conectada, formada normalmente por MLP. Em CNNs clássicas, as camadas (2) e (3) podem ocorrer múltiplas vezes.

2.3.3.3 Stacked auto-enconders (SAE)

SAE se refere a uma arquitetura de rede neural treinada para produzir como saída o sinal recebido como entrada. A arquitetura é formada tradicionalmente por dois componentes, o codificador (*encoder*) e o decodificador (*decoder*), respectivamente, como ilustrado na figura 2.5. Cada componente corresponde a uma rede neural distinta, com a saída do codificador compartilhada como entrada do decodificador. Assim, dado um conjunto de entrada x na camada X, o *auto-codificador* gera uma função f que aproxima x na camada X', tal que $f(x) = x' \approx x$. O vetor z na camada intermediária Z (a saída do codificador e entrada do decodificador) corresponde a uma representação compactada de x. Assim, a camada Z corresponde ao espaço latente de representação no qual z é uma instância descrita por meio de um conjunto de características latentes (o código). Note que cada duas camadas de um SAE podem ser vistas como um auto-codificador individual. Assim, um autocodificador com múltiplas camadas é chamado um autocodificador empilhado.

Basicamente, cada auto-codificador é treinado de forma separada e sequencial. Após o treinamento, as camadas do decodificador são descartadas e as camadas do

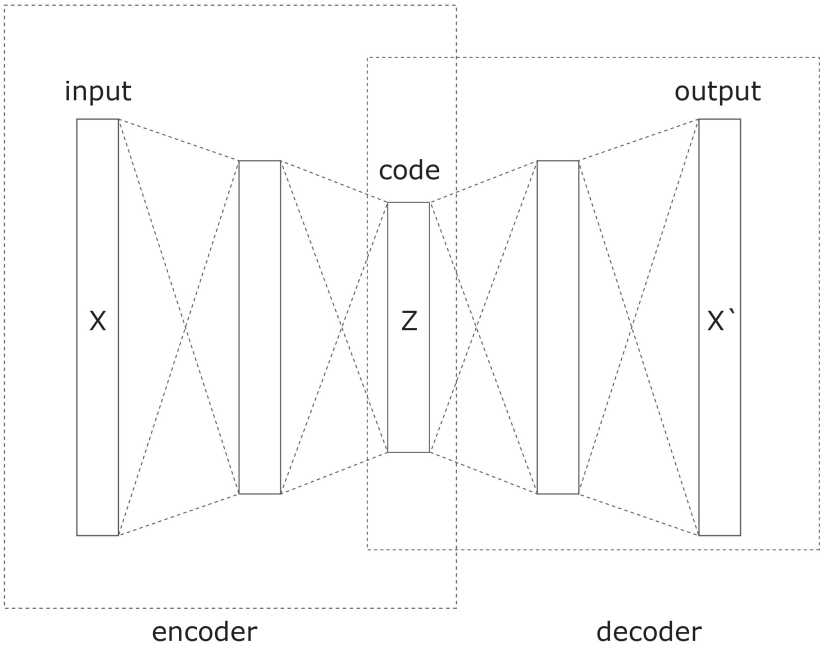

Figura 2.5: Arquitetura padrão de um auto-codificador.

codificador são transformadas nas camadas escondidas do SAE. Além disso, a camada de espaço latente é utilizada como entrada para o próximo *auto-enconder* que, por sua vez, dará origem a uma nova camada para o SAE. Por fim, o SAE é tratado como uma ANN simples.

2.3.3.4 Deep Belief Network (DBN)

DBNs são modelos gráficos geradores de várias camadas [Hinton & Salakhutdinov, 2006]. As camadas de uma DBN são formadas por *Restricted Boltzmann Machines* (RBMs). As RBMs, em si, podem ser vistas como redes neurais estocásticas recorrentes com duas camadas, uma visível e outra oculta, onde os nós de uma camada estão ligados a todos os nós da outra, não existindo ligações entre nós da mesma camada [Salakhutdinov et al., 2007]. Segundo Larochelle & Bengio [2008], essas ligações restritas entre os nós ajudam a aumentar a eficiência do processo de aprendizagem. A Figura 2.6 apresenta um exemplo de arquitetura de uma RBM que possui 5 nós na camada visível e 4 nós na camada oculta.

Uma RBM pode ser treinada de forma não supervisionada considerando um princípio biológico de minimização de energia: para aumentar sua chace de sobrevivência, seres vivos devem tomar decisões de forma efetiva e com mínimo consumo de energia. Considerando uma RBM formada por n nós visíveis $\boldsymbol{v}$ e m nós ocultos $\boldsymbol{h}$, podemos

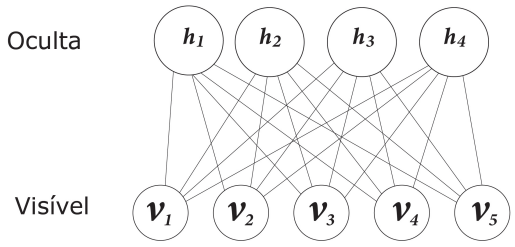

Figura 2.6: Arquitetura de uma RBM.

definir a energia da rede através da função:

$$E(v,h) = -\sum_{i=1}^{n} a_i v_i - \sum_{j=1}^{m} b_j h_j - \sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} v_i h_j \tag{2.29}$$

onde os termos a_i e b_j são os vieses para o nó visível binário v_i e nó oculto h_j, respectivamente. A variável w_{ij} é o peso entre o nó visível i e o nó oculto j. Note que os termos a_i e b_j representam os estados binários dos nós visíveis e ocultos [Hinton et al., 2006].

Em uma RBM, a probabilidade da rede ter vetor visível v e o vetor oculto h é definido pela distribuição de Boltzmann:

$$P(v,h) = \frac{1}{Z} e^{-E(v,h)} \tag{2.30}$$

onde Z é a função de partição que garante que a distribuição seja igual a 1. A partir da definição de $P(v,h)$, podemos verificar que a probabilidade de uma configuração aumenta à medida que sua energia diminui. Neste caso, treinar uma RBM de uma maneira não supervisionada envolve manipular a função de energia, de modo a atribuir baixa energia (e, portanto, alta probabilidade) a valores de v que sejam semelhantes aos dados de treinamento, e alta energia a valores distantes dos valores de treino [Hinton & Salakhutdinov, 2006].

Uma DBN pode ser obtida por simples empilhamento das RBMs. A Figura 2.7 exibe o passo-a-passo para a composição de uma DBN, onde (a) introduz uma arquitetura RBM comum, (b) mostra uma pilha RBM e (c) resume a arquitetura obtida. Neste caso, a camada oculta da primeira RBM é tratada como uma camada visível para a segunda RBM. A segunda RBM aprenderá a distribuição de características da camada oculta da RBM anterior. Além disso, a camada de entrada da primeira RBM é a camada de entrada para toda a rede. À medida que as camadas são empilhadas, a rede aprende combinações cada vez mais complexas das características originais [Hinton & Salakhutdinov, 2006].

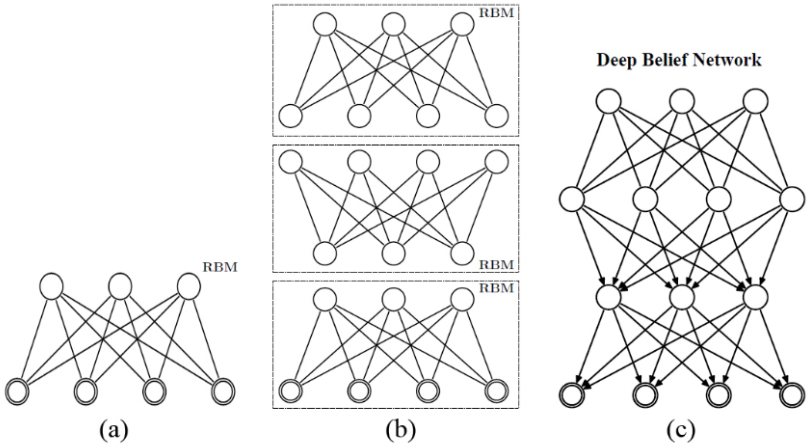

Figura 2.7: (a) RBM. (b) Pilha de RBMs. (c) DBN. Hinton & Salakhutdinov [2006]

O treinamento de uma RBM é composto por duas fases: *Pre-training* e *Fine-tuning*, como mostrado na Figura 2.8. O pré-treino é representado por uma pilha de RBMs, onde a saída de um nível é usada como entrada do próximo. A camada visível é usada para inserir dados de treinamento e a camada oculta é usada para detectar características. Nesse caso, as unidades são conectadas por pesos simétricos entre si. Essa estrutura torna o treinamento mais rápido, permitindo execuções paralelas e criando independência condicional da mesma camada [Hinton et al., 2006]. A pilha de RBMs é usada para obter pesos ideais, maximizando a função de verossimilhança que é executada [Hinton, 2012].

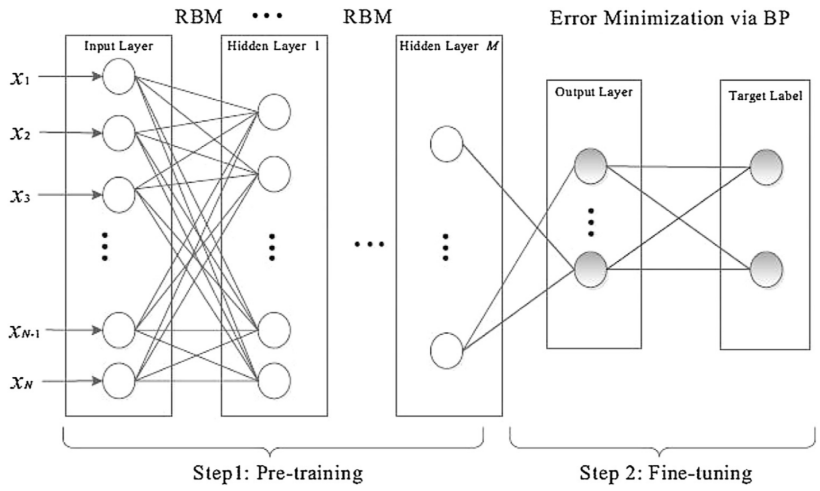

Figura 2.8: Uma arquitetura DBN usual Yu et al. [2018].

Após a etapa de pré-treinamento das camadas de RBM, a etapa *Fine-tuning* adota um algoritmo de propagação retrógrada para ajustar os parâmetros de acordo com o desempenho da classificação. Nesse caso, os rótulos das unidades são anexadas

à camada superior e uma aprendizagem *bottom-up* é realizada com os pesos adquiridos do pré-treinamento. Assim, os RBMs baseados no algoritmo de propagação retrógrada possuem maior capacidade de aprendizado e velocidade de convergência, uma vez que só precisam realizar uma busca local [Yu et al., 2018].

Hinton et al. [2006] propõe o seguinte resumo de passos para o processo de treinamento das RBMs de uma DBN:

1. Inicialização das unidades visíveis a partir de um vetor de dados de treino;
2. Atualização paralelizada das unidades ocultas, a partir das unidades visíveis;
3. Atualização paralelizada das unidades visíveis, a partir das unidades ocultas. Essa etapa é chamada de "reconstrução";
4. Atualização paralelizada das unidades ocultas, considerando as unidades visíveis reconstruídas usando a mesma equação da etapa 2;
5. Atualização dos pesos das conexões;

Em suma, DBNs foram criadas para lidar com problemas comuns em redes neurais profundas, ou seja, seu elevado tempo de treinamento e *overfitting*. O processo de treinamento de uma DBN consiste em treinar RBNs separadamente para depois empilhá-las [Salakhutdinov et al., 2007]. Ao fim do processo, se tem um modelo robusto capaz de extrair informações relevantes para um modelo de classificação tradicional como uma MLP [Luo et al., 2017]. Mais recentemente, DBNs se tornaram menos comuns devido aos avanços nos processos de treino e generalização de redes neurais profundas, além da disponibilização de arquiteturas de hardware muito focadas em arquiteturas neurais não estocásticas.

Na seção a seguir, descrevemos estratégias encontradas na literatura para lidar com desbalanceamento de classes em problemas como a pontuação de crédito.

2.4 Desbalanceamento de classes

As técnicas de aprendizado de máquina geralmente falham ou oferecem desempenho enganosamente otimista em conjuntos de dados de classificação que apresentam uma distribuição de classe desequilibrada. O motivo é que muitos algoritmos de aprendizado de máquina são projetados para operar com dados de classificação com um número igual de observações para cada classe. Quando esse não é o caso, os algoritmos podem aprender que os poucos exemplos de classes minoritárias não são importantes e podem ser ignorados para obter um bom desempenho [Branco et al., 2016].

A fim de lidar com o desbalanceamento de dados, a literatura de aprendizado de máquina fornece uma coleção de técnicas que transformam um conjunto de dados de treinamento, a fim de equilibrar ou equilibrar melhor a distribuição das classes. Após o balanceamento, os algoritmos padrão de aprendizado de máquina podem ser treinados diretamente no conjunto de dados transformado, sem a necessidade de nenhuma modificação. Isso permite que o desafio da classificação desequilibrada, mesmo com distribuições de classe severamente desbalanceadas, seja tratado com um método de preparação de dados [He & Ma, 2013].

A classificação desbalanceada envolve um conjunto de dados em que a distribuição de classes não é proporcional. Isso significa que o número de exemplos que pertencem a cada classe no conjunto de dados de treinamento pode variar muito. Em problemas de pontuação de crédito, por exemplo, é comum haver uma grande distorção na distribuição das classes, como 1:10, 1:20 ou até 1:100 na proporção de exemplos entre as classes minoritária e majoritária [Addo et al., 2018; He et al., 2008; He & Ma, 2013].

Embora frequentemente descrito em termos de problemas de classificação de duas classes, o desequilíbrio de classe também afeta os conjuntos de dados com mais de duas classes, podendo ocorrer várias classes minoritárias ou múltiplas classes majoritárias. Um dos principais problemas dos conjuntos de dados de classificação desequilibrados é que os algoritmos de aprendizado de máquina não apresentam um bom desempenho nesse grupo. Muitos algoritmos de aprendizado de máquina dependem da distribuição de classes no conjunto de dados de treinamento para avaliar a probabilidade de observar exemplos em cada classe, quando o modelo for usado para fazer previsões. Nesse sentido, algoritmos de aprendizado de máquina, como árvores de decisão, *k-nearest neighbors* e redes neurais, aprenderão que a classe minoritária não é tão importante quanto a classe majoritária, dedicando mais atenção à classe majoritária [Khan et al., 2018; Fernández et al., 2018].

2.4.1 Amostragem de dados

Problemas de desbalanceamento de classes costumam ser resolvidos a partir da modificação da composição do conjunto de dados de treinamento. Os métodos que fazem essa modificação são chamadas de métodos de amostragem e são famosos porque são simples de entender e implementar. Outra vantagem que torna esse métodos tão utilizados na literatura está no fato de que, uma vez que o conjunto de treinamento seja montado de forma balanceada, esse mesmo conjunto pode ser reaproveitado para o treinamento de muitos algoritmos diferentes de classificação, sem a necessidade de modificações nesses algoritmos. Com essa abordagem, permitimos que os algoritmos de classificação ana-

lisem de forma equivalente os atributos que melhor discriminam cada uma das classes [Branco et al., 2016; He & Ma, 2013].

Os métodos de amostragem são executados apenas no conjunto de dados de treinamento, deixando de fora os conjuntos de teste e validação. Isso acontece porque a intenção não é remover o viés de classe do ajuste do modelo, mas continuar a avaliar o modelo resultante em dados reais e representativos do domínio do problema. Existem duas abordagens principais para amostragem de dados, conhecidas como sobreamostragem (*oversampling*) e subamostragem (*undersampling*) de dados, onde métodos de sobreamostragem criam amostras balanceadas a partir do aumento do número de instâncias da classe minoritária e subamostragem diminuem a importância da classe majoritária a partir da redução do número de instâncias dessa classe. Apesar da simplificação gerada pelo balanceamento, a avaliação de um modelo treinado em um conjunto de dados onde ocorreu a transformação (aleatória ou sintética) da distribuição de classes pode fornecer uma estimativa otimista demais sobre o desempenho desses modelos [He & Ma, 2013].

Nas seções a seguir, apresentamos métodos de amostragem comumente utilizados em problemas de pontuação de crédito.

2.4.1.1 Métodos de sobreamostragem

Uma abordagem para lidar com conjuntos de dados desequilibrados é aumentar a importância da classe minoritária a partir da sobreamostragem de suas instâncias. Essa abordagem pode ser executada com base em dados reais ou dados sintéticos, conforme apresentado a seguir.

A sobreamostragem aleatória (*random oversampling*) é uma técnica simples de aumento da relevância baseada em dados reais, onde duplicamos aleatoriamente instâncias da classe minoritária, até que tenhamos uma amostra de treino onde a quantidade de instâncias de cada seja proporcional. Nessa abordagem, uma determinada instância da base original de treinamento pode ser duplicada e adicionada várias vezes à amostra transformada de treinamento [Fernández et al., 2018].

Apesar da duplicação de instâncias equilibrar a distribuição de classes, essa abordagem não adiciona novas informações ao modelo treinado. Além disso, a sobreamostragem aleatória pode aumentar a probabilidade de ocorrer super adaptação (*overfitting*) do modelo, uma vez que são geradas cópias exatas dos exemplos das classes minoritárias. Com isso, um classificador simbólico, por exemplo, pode construir regras que aparentam estar corretas, mas que, na verdade, cobrem apenas um exemplo que foi replicado muitas vezes. Dessa forma, para obter informações sobre o impacto do

método de amostragem, poderíamos avaliar o desempenho do modelo nos conjuntos de treino e teste, comparando os resultados que o mesmo classificador alcança no conjunto de dados original. O aumento no número de exemplos da classe minoritária, principalmente em casos de altíssimo grau de desbalanceamento, pode resultar em um aumento acentuado no custo computacional de treinamento do modelo [Fernández et al., 2018; Chawla et al., 2002].

Ao invés de simplesmente replicar várias vezes as mesmas instâncias da classe minoritária, poderíamos gerar novas instâncias sintetizadas a partir das instâncias já existentes. Esse é um tipo de aumento da influência da classe minoritária é conhecido como técnica de sobreamostragem de minorias sintéticas, mais conhecida pelo termo em inglês *Synthetic Minority Oversampling Technique* – SMOTE, proposta por Chawla et al. [2002]. O SMOTE seleciona amostras que estão próximas em um espaço de características. Mais especificamente, amostras sintéticas são criadas no eixo formado entre pares de amostras reais. Nesse processo, o SMOTE primeiro seleciona aleatoriamente uma instância a da classe minoritária e encontra seus k vizinhos de classe minoritária mais próximos. A instância sintética é criada escolhendo um dos k vizinhos mais próximos b aleatoriamente e conectando a e b para formar um segmento de reta no espaço de características. As instâncias sintéticas são geradas como uma combinação convexa das duas instâncias escolhidas [Chawla et al., 2002].

A Figura 2.9 apresenta graficamente a execução do SMOTE, onde os pontos $\{x_1, x_2, \ldots, x_n\}$ representam amostras da classe minoritária e os pontos $\{a, b, c, d, e\}$ representam as instâncias sintéticas criadas pelo SMOTE.

Outra abordagem utilizada para criação de amostras sintéticas é a amostragem sintética adaptativa, mais conhecida pelo termo em inglês *Adaptive Synthetic Sampling* – ADASYN. Essa abordagem é uma adaptação do SMOTE, proposta por He et al. [2008], que cria novas instâncias sintéticas de acordo com a densidade da classe minoritária no espaço de característica, uma vez que mais amostras sintéticas são criadas em regiões do espaço de características em que a classe minoritária é menos densa. Por outro lado, poucas ou nenhuma instância são criadas onde a classe minoritária tem alta densidade. Com isso, o ADASYN cria mais amostras para aquelas instâncias da classe minoritária que são mais difíceis de aprender e menos amostras para aquelas que são mais fáceis.

Uma desvantagem da abordagens baseadas em dados sintéticos é que exemplos sintéticos são criados sem considerar a classe majoritária e, com isso, possibilitando a criação de instâncias ambíguas, em bases que apresentem grande sobreposição entre as classes [Branco et al., 2016].

Na seção a seguir, apresentamos métodos de subamostragem utilizados para re-

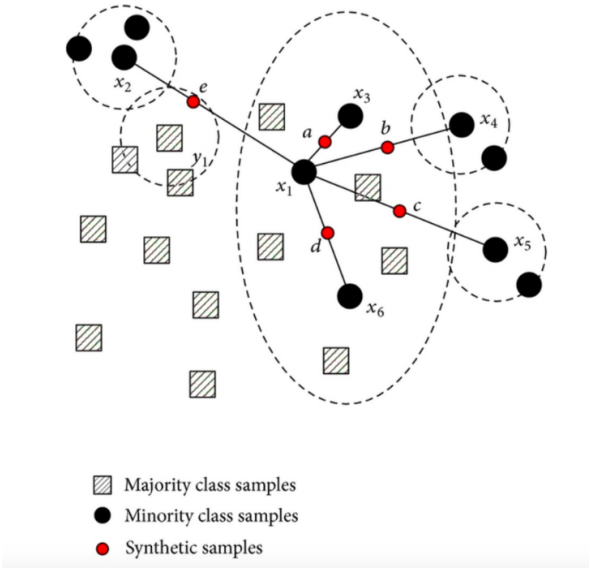

Figura 2.9: Execução do SMOTE no espaço de características. Adaptado de Chawla et al. [2002]

solver o problema de desbalanceamento de classes em pontuação de crédito.

2.4.1.2 Métodos de subamostragem

Métodos de subamostragem buscam o balanceamento dos dados de treinamento a partir da diminuição da influência das classes majoritárias. Apesar de existirem muitos tipos diferentes de técnicas de subamostragem, essas técnicas podem ser agrupadas em (1) técnicas que selecionam exemplos que serão mantidas no conjunto transformado de treino, (2) técnicas que selecionam instâncias a serem excluídas da base original de treinamento e (3) técnicas híbridas que combinam as duas técnicas anteriores.

A técnica de subamostragem utilizada em problemas de pontuação de crédito é a subamostragem aleatória (*random undersampling*) onde instâncias da classe majoritária são aleatoriamente escolhidas para serem excluídas da base de dados transformada para o treinamento. Esse processo pode ser repetido até que a distribuição de classe esteja balanceada, onde cada classe possui o mesmo número de instâncias. Essa abordagem pode ser mais adequada para os conjuntos de dados em que há um desequilíbrio de classe, embora ainda haja um número suficiente de exemplos na classe minoritária, de modo que um modelo possa ser adequadamente treinado [Fernández et al., 2018].

Uma limitação da subamostragem é que são excluídos exemplos da classe majoritária que podem ser úteis, importantes ou talvez críticos para que seja encontrado um limite de decisão robusto, pois os exemplos são excluídos aleatoriamente e não há como detectar ou preservar exemplos bons ou mais ricos em informações da classe

majoritária [Chawla et al., 2002].

Na seção a seguir, apresentamos os conceitos fundamentais relacionados à aprendizagem sensível ao custo.

2.5 Aprendizagem Sensível ao Custo

A classificação é um problema de modelagem preditiva que envolve a previsão de um rótulo (classe) para uma determinada amostra. Apesar de existirem problemas onde precisamos distinguir entre muitas classes diferentes, problemas de classificação binária, onde queremos distinguir entre duas classes, uma positiva e uma negativa, são os mais comuns [Wang et al., 2015]. O problema de pontuação de crédito pode ser tratado como problema de classificação binária, pois estamos interessados em classificar amostras de clientes como bons ou maus pagadores [Yifei, 2016].

A maioria dos algoritmos de aprendizagem de máquina, projetados para resolver o problema de classificação, partem de duas premissas: (1) todas as classes possuem mais ou menos o mesmo número de instâncias e (2) todos os erros de classificação têm a mesma importância. Ou seja, as amostras estão minimamente balanceadas e todos os erros de classificação incorreta possuem o mesmo custo. Contudo, isso não é a realidade de problemas como diagnóstico de câncer, detecção de fraude e pontuação de crédito, onde as distribuições de classe são altamente desbalanceadas e o erro de classificar incorretamente uma instância da classe minoritária tem custo maior que o erro de classificação da classe majoritária [Khan et al., 2018; Menon & Williamson, 2016].

Em problemas de classificação binária desbalanceada, é comum que instâncias da classe majoritária sejam definidas como *negativas* e rotuladas com o valor 0 (zero). Instâncias da classe minoritária, por sua vez, costumam ser marcadas como *positivas* e rotuladas com o valor 1 (um). A razão para a rotulação baseada na convenção de positivo vs negativo está no fato de que instâncias da classe majoritária representam um caso normal (ou sem evento), enquanto a amostras da classe minoritária representam o caso excepcional (ou caso de ocorrência do evento) [Ertekin & Rudin, 2011]. Neste trabalho, estamos interessados em avaliar se uma determinada instância de cliente é ou não inadimplente. Por isso, seguimos a convenção:

1. Classe majoritária: Negativa para a ocorrência do evento de inadimplência, marcada com o rótulo de valor 0 (zero);

2. Classe minoritária: Positiva para a o evento de inadimplência, marcada com o rótulo de valor 1 (um);

Tradicionalmente, algoritmos de aprendizagem de máquina são treinados em um conjunto de dados e procuram minimizar o erro do modelo. Nesse caso, para que o modelo seja adequadamente treinado, esses algoritmos resolvem um problema de otimização onde buscam explicitamente minimizar o erro. Muitas funções de perda foram proposta na literatura de aprendizagem de máquina para calcular a erro do modelo, durante a fase de treinamento [Menon & Williamson, 2016; Zhao et al., 2011]. Neste trabalho, apresentamos duas novas funções de perda para lidar com o problema de classificação binária em conjuntos de dados altamente desbalanceados.

Na seção a seguir, apresentamos como conceitos de aprendizado sensível a custos baseados em custo de classificação incorreta podem ajudar na classificação em bases de dados desbalanceadas.

2.5.1 Classificação desbalanceada sensível ao custo

O aprendizado sensível a custos para classificação desequilibrada concentra-se em atribuir custos diferentes aos diferentes tipos de erros de classificação incorreta que podem ser cometidos e, em seguida, usar métodos especializados que levem esses custos em consideração. Os custos variáveis de classificação incorreta, falso positivo (FP) e falso negativo (FN), são melhor compreendidos quando adotamos uma matriz de custos para modelar classificações incorretas e suas penalidades [Wang et al., 2015]. Na aprendizagem de máquina, a matriz de custos é usada para influenciar o modelo a minimizar o custo de classificação incorreta ou maximizar o benefício da classificação correta. Nesse caso, seja $\mathcal{R}_{i,k}$ o risco (custo) de prever uma instância da classe i como uma instância da classe k. Logo, $\mathcal{R}_{+1,-1}$ representa o risco de classificar incorretamente uma instância positiva (classe minoritária) como instância negativa (classe majoritária) e $\mathcal{R}_{-1,+1}$ é o risco do contrário [Yan et al., 2003]. As abordagens sensíveis aos custos geralmente lidam com problemas de desbalanceamento de classe aumentando a importância da classe rara, definindo $\mathcal{R}_{+1,-1} > \mathcal{R}_{-1,+1}$ e $\mathcal{R}_{+1,+1} = \mathcal{R}_{-1,-1} = 0$ para classificações corretas. Com isso, poderíamos definir uma matriz de custo Ω como mostrado na Equação 2.31.

$$\Omega = \begin{pmatrix} \mathcal{R}_{+1,+1} & \mathcal{R}_{+1,-1} \\ \mathcal{R}_{-1,+1} & \mathcal{R}_{-1,-1} \end{pmatrix} = \begin{pmatrix} 0 & \mathcal{R}_{-1,+1} \\ \mathcal{R}_{+1,-1} & 0 \end{pmatrix} \tag{2.31}$$

Algoritmos de aprendizagem sensível ao custo buscam minimizar o custo total de classificação incorreta. Baseados na matriz da Equação 2.31, podemos definir o custo total de um classificador como a soma ponderada dos custos de erros de classificação

de falsos positivos e falsos negativos:

$$CustoTotal = \mathcal{R}_{+1,-1} \times FN + \mathcal{R}_{-1,+1} \times FP \tag{2.32}$$

Em algoritmos de aprendizagem de máquina, a escolha da matriz de custo é importante para o processo de aprendizagem, por representar um dos componentes necessários ao processo de minimização do erro de classificação [Menon & Williamson, 2016]. Em alguns domínios de problema, a definição da matriz de custo pode ser óbvia, como, por exemplo, em problemas de reivindicação de seguros, onde o custo de um falso positivo pode ser equivalente ao custo financeiro da operação. Por outro lado, em domínios como o de detecção de doenças, por exemplo, o custo do falso positivo para a instituição de saúde poderia ser medido como o custo financeiro para a realização de nova bateria de exames, mas é uma tarefa muito difícil inferir o custo financeiro para um paciente que está doente, mas foi diagnosticado como saudável pelo modelo.

Algoritmos de aprendizagem de máquina podem adotar diferentes abordagens para definições da matriz de custo, sendo que uma abordagem das abordagens mais comuns, utilizada sempre que possível, é o uso de um especialista humano para a definição dos custos de erros de classificação do modelo. Uma outra abordagem comum em aprendizagem sensível ao custo é que a definição inicial da matriz de custos seja elaborada a partir da distribuição inversa de classes, o que naturalmente aumenta a importância do erro de classificação incorreta de instâncias da classe minoritária [Khan et al., 2018]. Após a sua definição inicial, a matriz de custos pode ou não ser atualizada ao longo do processo de treinamento, a fim de refletir melhor os custos mais adequados ao problema. Contudo, atualizar manualmente os componentes da matriz de custos pode ser uma atividade muito trabalhosa, sendo ideal que, de alguma forma, os custos possam ser reavaliados e atualizados, sempre que necessário, ao longo do processo de aprendizagem do modelo de classificação.

A função de perda é outro componente importante para ao processo de aprendizagem sensível ao custo, uma vez que essa função é responsável pelo cálculo que indica se o classificador está ou não caminhando na direção correta. Funções custo usadas normalmente em algoritmos de classificação binária costumam ser simétricas, no sentido de assumirem o mesmo custo para instâncias classificadas incorretamente, independente da classe [Menon & Williamson, 2016; Ertekin & Rudin, 2011; Khan et al., 2018]. No entanto, em problemas como pontuação de crédito, o custo para classificar incorretamente o mau pagador é muito maior do que classificar incorretamente o bom pagador, o que requer funções de custo que sejam assimétricas para as duas classes. Ou seja, funções que permitam que o custo da classificação incorreta da classe

positiva cresça independentemente do custo do erro de classificação da classe negativa e vice-versa.

2.6 Bases de dados públicas

Durante nossa revisão bibliográfica, verificamos que duas bases públicas são comumente utilizadas em avaliações de modelos de pontuação de crédito, apesar de não atenderem a algumas restrições relacionadas ao tamanho das amostras, apresentadas na Seção 2.1.1.

A base de dados *German credit*[1] é uma base com dados de 1000 clientes (700 bons e 300 maus), descritos por 24 atributos, como taxa de parcelamento em percentual do rendimento disponível, idade em anos e número de dependentes.

A base de dados *Australian credit*[2] é uma base de dados composta por 690 instâncias de clientes, sendo 307 maus e 383 bons pagadores, onde cada instância é representada por 15 atributos, sendo 8 atributos numéricos, 6 atributos categóricos, e uma característica discriminante que foi convertida para um atributo numérico, por questões de confidencialidade.

A base de dados *Japanese credit card*[3] é composta por 652 amostras de clientes, com 356 instâncias de bons pagadores e 296 instâncias de maus pagadores, descritas a partir de 15 atributos, sendo 6 numéricos e 9 categóricos. Os nomes dos clientes foram anonimizados e as instâncias com dados ausentes foram excluídas.

2.7 Considerações Finais

Nesta seção, apresentamos os conceitos básicos usados ao longo desta tese. Em particular, discutimos modelos de pontuação de crédito, técnicas para a amostragem de dados neste tipo de aplicação e sua típica representação das instâncias de entrada, métricas de avaliação usadas tanto em classificação como ranqueamento, abordagens para lidar com desbalanceamento de classes, além das técnicas estatísticas e de aprendizagem de máquina tipicamente empregadas. Entre as técnicas abordadas , procuramos descrever todas as observadas na literatura, com ênfase para as de aprendizagem profunda, o que Redes Neurais e o algoritmo de retropropagação de erros, CNNs, auto-codificadores e DBNs. Dos tópicos relevantes para esta tese, o único não coberto foi o estudo de

[1] `http://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)`
[2] `http://archive.ics.uci.edu/ml/datasets/statlog+(australian+credit+approval)`
[3] `https://archive.ics.uci.edu/ml/datasets/credit+approval`

função assimétricas, que será introduzido junto com nossas funções de custo propostas no Capítulo 4.

Capítulo 3

Trabalhos relacionados

Problemas como *credit scoring*, onde normalmente há um alto grau de desbalanceamento da distribuição de classes, são desafiadores para algoritmos de aprendizagem de máquina que precisam definir limites de separação entre classes, com base nas características presentes em instâncias do conjunto de dados de treinamento [Khan et al., 2018]. Neste caso, tanto modelos rasos [Brown & Mues, 2012; He et al., 2018; Haixiang et al., 2017] quanto modelos profundos [Kvamme et al., 2018; Zhu et al., 2018; Addo et al., 2018] podem ser afetados pelo desbalanceamento de classes das amostras de treino.

Neste capítulo, descrevemos o estado-da-arte em métodos de aprendizagem profunda que resolvem o problema de *credit scoring* utilizando ou não alguma estratégia de tratamento para desbalanceamento de dados. Também analisamos técnicas para lidar com o problema de desbalanceamento que poderiam ser aplicadas ao problema de *credit scoring*.

3.1 Abordagens baseadas em balanceamento de dados

O problema em lidar com conjuntos de dados desbalanceados é uma questão-chave em classificação binária e tem sido muito explorado nas últimas décadas. A fim de diminuir o impacto desse problema, muitos métodos têm sido propostos na literatura, onde os autores resolvem o desbalanceamento de classes a partir de abordagens de balanceamento baseadas em atributos, dados ou algoritmo [Abdallah et al., 2016; Pant & Srivastava, 2015], conforme descrito na Seção 2.4.

3.1.1 Balanceamento em nível de dados

As abordagens de balanceamento em nível dos dados são frequentemente usadas como uma etapa de pré-processamento nos modelos de aprendizado de máquina para re-equilibrar o conjunto de dados de treinamento antes da aplicação dos algoritmos de classificação [Abdallah et al., 2016]. Estas abordagens lidam com as representações de classe no conjunto de dados original aumentando a importância da classe minoritária ou diminuindo a importância da classe majoritária, a fim de equilibrar a distribuição resultante de dados [Fernández et al., 2018; Brown & Mues, 2012].

Classificação de crédito com DBN, SVM e amostragem de dados. Yu et al. [2018] propuseram uma estratégia híbrida que combina um *emsemble* de SVMs e uma DBN para lidar com problemas de dados desbalanceados na criação de modelos de classificação de risco de crédito. Nesse paradigma, um algoritmo de *bagging* foi usado primeiro para gerar subconjuntos de treinamento variáveis, balanceados e de tamamho adequado. Em seguida, um modelo SVM foi usado como classificador, a fim de formular diversos membros de entrada para o *ensemble*. Por fim, uma DBN é usada como método de *ensemble* para gerar os resultados da classificação. Em suma, o método é organizado em três estágios: particionamento de dados, treinamento de classificadores e classificação por *ensemble*, utilizando as bases de dados *German credit* e *Japanese credit card*, descritas na Seção 2.6.

No estágio 1 ocorre o particionamento dos dados, onde o conjunto de dados é dividido em amostras de treino (TR) e teste (TS). Em seguida, é aplicado um algoritmo de *bagging* ao conjunto de dados TR, gerando vários subconjuntos de treinamento TR_i ($i = 1, \ldots, n$) e o subconjunto de validação EI. Esses subconjuntos são compostos por instâncias diferentes e são balanceados, apresentando a mesma quantidade de instâncias e a mesma quantidade de instâncias por classe.

No estágio 2 ocorre o treinamento dos classificadores, onde os subconjuntos TR_i são usados como bases de treino para os SVMs, por meio de uma validação cruzada de 10 folds com *grid search* para ajuste dos hiper-parâmetros. Após o treino, cada modelo SVM gera uma resposta (0 ou 1) para cada uma das instâncias do subconjunto EI. Como resultado desse estágio, para cada instância de EI, é gerado um vetor com n posições, onde n é o número de classificadores SVM utilizados. Neste caso, considerando a criação de n classificadores SVM, o resultado da avaliação de uma instância de EI seria um vetor binário de n dimensões onde cada coordenada do vetor representa a resposta de um classificador SVM.

No estágio 3 ocorre a classificação por *ensemble*, onde uma estratégia de *ensemble*

baseada em DBN é usada para gerar a classificação final. Para isso, a DBN recebe como entrada os os valores de previsão gerados pelos classificadores SVM, para cada instância do subconjunto EI, bem como suas respectivas classes reais. A DBN usada neste estágio segue os padrões de treinamento descritos na Seção 2.3.3.4.

No modelo de *ensemble* de DBN, os autores construíram uma arquitetura com duas camadas ocultas, adotando 1000 épocas para treino e *fine-tuning*. Como funções de ativação, foram utilizadas *sigmoid* nas duas camadas ocultas e função linear na camada de saída. A quantidade de modelos SVM variou em função dos conjunto de dados utilizados onde, para a base *German* foram treinados 40 modelos SVM, com taxa de aprendizagem e de *momentum* de 0.001 e 0.01, respectivamente. Por outro lado, como o conjunto de dados *Japanese* era menor, foram criados apenas 25 classificadores SVM com taxas de aprendizagem e de *momentum* de 0.005 e 0.01. A avaliação dessa abordagem foi baseada em uma métrica de acurácia ponderada, proposta pelos próprios autores, que leva em consideração a taxa de desbalanceamento da amostra (IR) e o valor de receitas que o modelo pode gerar. Essa métrica de avaliação foi chamada de *Waccuracy* e usa a Tabela 3.1 como base, onde os autores consideram os seguintes critérios para cálculo da receita:

- Receita por evitar despesa (VP = receita);
- Juros recebidos com o acerto de bons pagadores (VN = juros); e
- Não há receitas em caso de erros de previsão (FP = FN = 0);

Tabela 3.1: Uma matriz sensível à receita

		Classe real	
		Mau	**Bom**
Classe Prevista	**Mau**	VP (receita)	FP (0)
	Bom	FN (0)	VN (juros)

Com base no exposto, a métrica $Waccuracy$ é calculada usando a seguinte equação:

$$Waccuracy = \frac{(VP \times VPac) + (\text{juros} \times IR \times VNac)}{\text{receita} + (\text{juros} \times IR)} \tag{3.1}$$

onde VPac é a acurácia da classificação da classe minoritária e VNac é a acurácia da classificação da classe majoritária.

Por fim, Yu et al. [2018] avaliaram seu método *Ensemble-DBN* contra classificação simples de SVM e contra um *ensemble* baseado em votação por maioria. Todos os modelos utilizaram como pré-processamento as abordagens de *oversampling* e *undersampling* aleatórios. Como resultado, o modelo proposto venceu na base de dados *German*, usando *undersampling*, atingindo $Waccuracy = 81.83\%$. Porém, o modelo proposto apenas empatou com o *Ensemble* de votação por maioria na base de dados *Japanese*, usando *oversampling* e alcançando $Waccuracy = 93.06\%$.

Previsão de inadimplência de hipotecas usando CNN e Oversampling. Kvamme et al. [2018] propuseram um método para previsão de bons e maus pagadores baseado em dados transacionais das contas bancárias dos clientes, como os balanços da conta corrente (ch), da conta poupança (sa), do cartão de crédito (cc), além do número de transações da conta corrente (trch) e a soma de transferências entre contas correntes (in). A metodologia proposta pelos autores pode ser aplicada tanto em *Application*, quanto em *Behavior Scoring*.

O método utilizado para gerar o modelo de classificação foi uma CNN. Essa escolha foi realizada, segundo os autores, porque a vantagem da CNN, sob a perspectiva de pontuação de crédito, é que em vez de usar apenas as covariáveis definidas manualmente por um especialista humano, como as apresentadas na Tabela 2.1, a CNN é capaz de aprender como gerar boas variáveis exploratórias de forma automática a partir do histórico das transações bancárias dos clientes. Dessa forma, a engenharia de atributos é realizada automaticamente pela própria CNN substituindo o processo manual de extração de características.

Os dados bancários utilizados no trabalho foram organizados em séries temporais representadas pelos balanços transacionais diários das contas bancárias dos clientes. Cada cliente possui uma coleção de seis séries temporais (ch, sa, cc, trch, in e sum). A série temporal 'sum' representa a soma das séries temporais ch, sa e cc. A base de dados foi disponibilizada pela instituição *Norwegian Financial Service Group* (DNB). Os dados consistem em amostras de 20.989 clientes, no período de janeiro de 2012 a abril de 2016. Essa base de dados foi dividida em treino e teste de acordo com a Tabela 3.2.

Para resolver o problema de desbalanceamento da base de dados foi utilizada a estratégia de *oversampling* baseada em janelas deslizantes (*sliding-window*). Basicamente, as janelas são agregadas ao longo do tempo, com uma determinada taxa de sobreposição, conforme mostra a Figura 3.1. Note que cada janela de tempo possui tamanho de 1 ano, definido pela área cinza, que é contado do mês anterior ao mês de

Tabela 3.2: Proporções de dados dos conjuntos de treinamento, validação e testes.

Conjunto de dados	Maus	Bons	Total
Treinamento	1298	11398	12696
Treinamento aumentado	95647	841247	936894
Validação	329	6043	6372
Testes	96	1825	1921

inadimplência até 12 meses no passado. As quantidades de dados dos clientes inadimplentes são aumentadas na proporção indicada na Tabela 3.2.

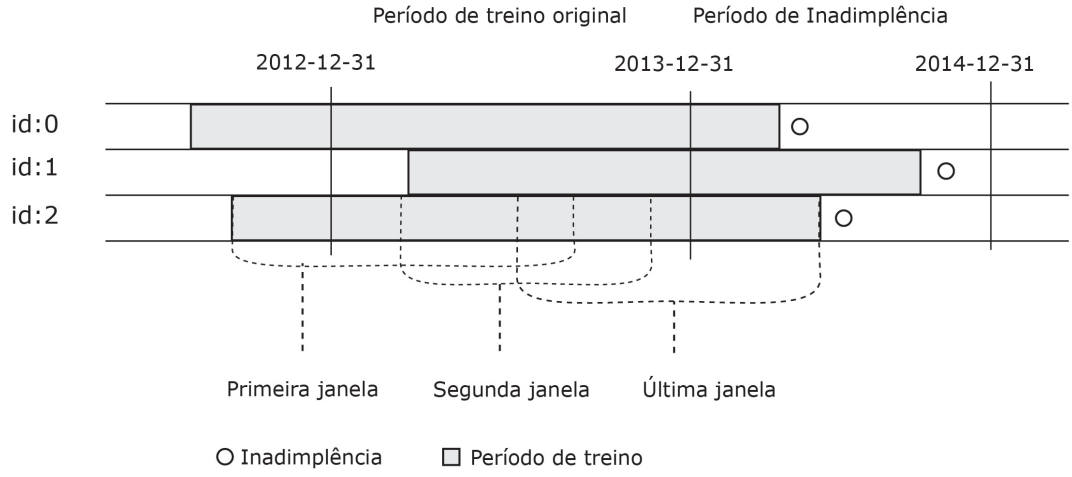

Figura 3.1: Esquema de aumento de dados. Adaptado de Kvammea et al. [2018]

A entrada de dados de uma CNN é comumente escalada no intervalo entre [0,1]. No trabalho de Kvamme et al. [2018], a série temporal x é escalada de acordo com a fórmula:

$$x = \frac{x - min(x)}{max(x) - min(x)} \tag{3.2}$$

A arquitetura da CNN utilizada foi composta por duas camadas de convolução e duas camadas densas. O formato da entrada de dados foi uma série temporal com 365 amostras (dias). As camadas de convolução operaram no modo unidimensional (Conv1D) e utilizaram a função de ativação *ReLU*. A primeira camada foi formada por 32 blocos de convolução com filtro de tamanho 9. A segunda camada usou 64 blocos com filtro de tamanho 7. Após o processamento da primeira camada foi utilizado um *max-*

pooling de tamanho 4 e um *max-pooling* de tamanho 2 na segunda camada. No caso das camadas densas, a primeira teve tamanho 64 e a segunda tamanho 2, representando as classes de *bom* e o *mau* pagador. Ao todo, a quantidade de parâmetros da CNN foi de 199234 pesos para uma entrada de 365 dias. A Tabela 3.3 mostra um resumo da arquitetura.

Tabela 3.3: Configuração de parâmetros da CNN.

CAMADA	PARÂMETROS
Convolução#1	32 filtros e kernel de tamanho 9
Max Pooling#1	Tamanho igual a 4
Convolução#2	64 filtros e kernel de tamanho 7
Max Pooling#2	Tamanho 2
Densa#3	64 unidades
Dropout #3	0.5 de taxa de *dropout*
Densa#4	2 unidades

A camada densa de duas unidades é a camada que classifica o cliente em bom ou mau pagador. Para isso, essa camada utiliza a função de ativação *Softmax*. Além disso, por se tratar de um problema de classificação binária, a função de perda utilizada foi a Entropia Cruzada Binária.

O treinamento da CNN foi baseado no SGD por meio do algoritmo de otimização *Adam*. O tamanho de *batch* utilizado foi 512. Para cada série temporal usada no treinamento, foi treinada uma CNN diferente. O resultado final consiste no valor médio da previsão das seis CNN's geradas no processo de treinamento. Para evitar *overfitting* foi utilizada a técnica de parada antecipada (*early stopping*), cujo objetivo é monitorar o erro ao longo do treinamento e, quando o *overfitting* começa a ser detectado, o treinamento é imediatamente encerrado. Além disso, os autores utilizam a técnica de regularização de *dropout* entre as duas últimas camadas densas da CNN, com uma taxa de 0.5. Na avaliação dos resultados, a CNN alcançou Acurácia = 0.95 e $AUC = 0.91$.

Análise de risco de crédito com DFN e SMOTE. Addo et al. [2018] realizaram uma análise comparativa entre algoritmos clássicos de aprendizagem de máquina e algoritmos de aprendizagem profunda no contexto de previsão da probabilidade de inadimplência durante a solicitação de um empréstimo. Os autores destacam a importância da escolha dos algoritmos de classificação, dos parâmetros, da seleção de

características e do critério de avaliação dos modelos, tendo como objetivo tornar o resultado dos modelos de classificação claros, transparentes, facilitando a tomada de decisão final das instituições credoras.

Os dados utilizados nos experimentos foram providos pelo *European Bank* e contém 117.019 amostras e 235 variáveis. Nessa base de dados, 115.288 amostras são de bons pagadores e 1.731 são de maus pagadores. Os dados foram limpos e 54 variáveis foram removidas. O problema de desbalanceamento dos dados foi resolvido por meio do algoritmo SMOTE. Após o balanceamento, a nova proporção dos dados passou a ser de 46% de amostras de maus pagadores e 53% de amostras de bons pagadores.

Os autores elegeram os seguintes algoritmos rasos:

- *Logistic Regression* (LR) baseado na estratégia de treino *Elastic Net* com taxa de penalização ($\alpha = 0.5$ e $\gamma = 1.9210^{-6}$);
- Random Forest (RF) com 120 árvores (esse número foi baseado no número de características) e o critério de parada 10^{-3};
- *Grandient Boosting* (GB) com taxa de aprendizagem 0.3, critério de parada 10^{-3} e quantidade de árvores igual a 120;

Os algoritmos de aprendizagem profunda foram treinados com o gradiente descendente estocástico (*Stochastic Gradient Descent* – SGD) e para evitar $overfiting$ foi utilizado o critério de parada com 20 épocas. Os autores decidiram avaliar uma DFN usando 4 arquiteturas distintas:

- D1: duas camadas escondidas com 120 neurônios cada;
- D2: três camadas escondidas com 40 neurônios cada;
- D3: três camadas escondidas com 120 neurônios cada uma e funções de regularização L_1 e L_2;
- D4: uma estratégia de busca de parâmetros (*grid* de hiper-parâmetros) foi utilizada para definir o melhor modelo. O resultado encontrou os seguintes parâmetros: Dropout = 0.0, $L_1 = 6,8.10^{-5}$, $L_2 = 6,4.10^{-5}$, camadas ocultas = [50, 50], função de ativação *Rectifier*;

As métricas utilizadas na avaliação dos resultados foram a AUC e a raiz do erro quadrado médio (*Root Mean Square Error* – RMSE). Os modelos de classificação foram treinados com 60% dos dados, 20% foram utilizados para validação e o restante, (20%),

foi utilizado para teste. Os resultados mostraram que os algoritmos rasos GB e RF obtiveram os melhores resultados com um AUC média de 0.994 e 0.993, respectivamente. Nessa perspectiva, os autores concluíram que nem sempre os métodos de aprendizagem profunda alcançam os melhores resultados. Com esses resultados, decidimos avaliar GB e RF em nosso trabalho, como descrito na Seção 5.5.1.

Apesar de seu uso frequente em tarefas de classificação, as abordagens de balanceamento no nível de dados podem apresentar desvantagens. Por exemplo, *undersampling* pode causar perda de informações na classe majoritária, enquanto *oversampling* pode gerar *overfitting* ao gerar cópias exatas de amostras de classes minoritárias, além de aumentar o tamanho do conjunto de treinamento e os custos de processamento [Khan et al., 2018].

3.1.2 Balanceamento em nível de atributos

O problema de desbalanceamento de dados se torna ainda mais desafiador quando o domínio do problema envolve um grande número de atributos, como acontece em problemas de *credit scoring*. Isso ocorre porque os métodos de balanceamento costumam explorar a distância entre as instâncias de treinamento para decidir como executar a reamostragem do conjunto de dados. As abordagens de distância usadas nesse cálculo, geralmente derivadas da distância euclidiana [Wilson & Martinez, 1997], são diretamente afetadas pela maldição de alta dimensionalidade [Pant & Srivastava, 2015; Shakeel et al., 2017].

A abordagem de seleção de atributos é baseada em uma etapa de pré-processamento usada para selecionar os atributos que melhor definem a classe de uma instância para diminuir a dimensionalidade, reduzir o tempo de execução e melhorar a precisão da classificação [Lango & Stefanowski, 2018]. Na literatura de aprendizado de máquina, muitas abordagens de seleção de atributos foram propostas para lidar com a maldição de alta dimensionalidade em dados desbalanceados [Haixiang et al., 2017; Pant & Srivastava, 2015; Tang et al., 2014; Chen & Wasikowski, 2008]. Entre essas abordagens, o algoritmo *Relief* e a pontuação de atributos foram usados para criar modelos profundos de avaliação de risco de crédito [Zhu et al., 2018; Ha & Nguyen, 2016], como descrito a seguir.

Pontuação de crédito ao consumidor baseada em CNN e Relief. Zhu et al. [2018] propuseram um método para classificação de bons and maus pagadores a partir da combinação de estratégia de seleção de atributos e modelos de aprendizagem profunda, onde são usados dados de transações bancárias e de cartão de crédito, como:

pagamentos diários, reembolso de micro-crédito, pagamento com cartão de crédito e saldo da conta corrente.

O método proposto foi chamado de "Relief-CNN", por combinar o algoritmo *Relief Network* com uma CNN para criação de modelos de classificação de risco de crédito. A intuição por trás dessa combinação é que a seleção das variáveis mais relevantes deve incrementar a qualidade do modelo gerado pela CNN.

A base de dados utilizada no trabalho foi coletada a partir de registros da *Chinese consumer finance company*. Essa base é composta por 24.837 registros de crédito do clientes, descritos por 576 atributos numéricos. Esse conjunto de dados apresenta uma alta taxa de desbalanceamento, com 2.098 maus (8.44%) e 22.739 bons (91.55%) clientes.

O Relief-CNN foi estruturado em três estágios distintos: seleção de atributos, transformação de dados e, por fim, classificação, conforme descrito a seguir.

Na fase de seleção de atributos ocorre o processo de seleção das variáveis mais relevantes para o desenvolvimento do modelo de crédito [Liu & Motoda, 2012]. Essa seleção reduz a complexidade do modelo e aumenta a velocidade e a eficiência do algoritmo de aprendizagem, segundo descrito por Guyon & Elisseeff [2003].

O *Relief* é um algoritmo para seleção de variáveis que foi proposto por Kira & Rendell [1992], e tem como ideia fundamental o uso de regras de vizinhança para seleção dos atributos mais relevantes para o modelo. Nesse casos, considerando um conjunto de treino $S = \{X_1, X_2, \ldots, X_n\}$, composto pelo seu conjunto de atributos $F = \{f_1, f_2, ..., f_p\}$, e assumindo que cada instância X_i pode ser vista como um vetor p-dimensional $\{x_{i1}, x_{i2}, ..., x_{ip}\}$, o *Relief* executa uma busca aleatória e, para cada instância X_i sorteada, devem ser achadas as instâncias de hit, que são as instâncias mais próximas e pertencentes à mesma classe, e as instâncias de $miss$, que são as instâncias mais próximas e pertencentes a classes diferentes. Com isso, o vetor de pesos é formado a partir da equação

$$W_i = W_{i-1} - diff(X_i - hit_i)^2 + diff(X_i - miss_i)^2 \tag{3.3}$$

O algoritmo *Relief* repete o procedimento m vezes e divide cada elemento do vetor final de pesos por m, gerando o vetor resultado de pesos. Normalmente, as características selecionadas têm peso maior que um determinado limiar τ.

A fase de transformação de dados é utilizada para converter os dados de transações de clientes em uma matriz de pixels, que servirá de entrada para a CNN. Neste caso, para cada instância de cliente $X_i = \{x_{i1}, x_{i2}, ..., x_{is}\}$, as variáveis contínuas são transformadas em categóricas com k valores. Em seguida, cada atributo x_{ij} é transfor-

mado em um vetor binário $\{l_1, l_2, \ldots, l_k\}$, onde:

$$l_i = \begin{cases} 1 & \text{quando } x_{ij} \text{ se enquadra na } i^{th} \text{ categoria} \\ 0 & \text{caso contrário} \end{cases} \tag{3.4}$$

Com a transformação, os dados de clientes rotulados são convertidos em uma imagem de cinza rotulada, como

$$y_i + \begin{pmatrix} 0 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 0 \end{pmatrix}_{k \text{ x } s} \tag{3.5}$$

Para a fase de classificação, os autores adotaram uma CNN, conforme a estrutura proposta por Krizhevsky et al. [2012], que incluiu a técnica de *dropout* para reduzir o número de co-adaptações complexas de neurônios e prevenir *overfitting*.

Os experimentos foram realizados da seguinte forma: os 50 atributos mais significativos (dos 576) foram selecionados usando o algoritmo *Relief*. Em seguida, cada instância de cliente foi transformada em uma 16x50 matriz de pixel. O conjunto de dados foi dividido em amostras de treino (70%) e validação (30%), usando a abordagem de validação cruzada com *10-fold*. O modelo foi treinado a partir de uma CNN, construída segundo as configurações apresentadas na Tabela 3.4, ao longo de 12 épocas e adotando um *batch* de tamanho 128.

Tabela 3.4: Configurações de parâmetros da Relief-CNN.

CAMADA	PARÂMETROS
Convolução #1	32 filtros, kernel de tamanho 3x3
Convolução #2	64 filtros, kernel de tamanho 3x3
Pooling	Tamanho 2x2
Dropout #1	Taxa de *dropout* de 0.25
Dropout #2	Taxa de *dropout* 0.5
Densa	128 unidades

O desempenho do modelo treinado foi comparado com o desempenho dos modelos *Random Forest* e regressão logística, utilizando as métricas Acurácia, AUC e KS, onde o Relief-CNN obteve os melhores resultados, atingindo $AUC = 0.6989$, $KS = 0.312$ e *Acurácia*= 91.60%.

Pontuação de crédito com DFN e novo método de seleção de atributos. Ha & Nguyen [2016] propuseram uma nova abordagem para seleção de variáveis baseada em pontuação de atributos, que foi combinada com um classificador baseado em aprendizagem profunda para resolver o problema de previsão de risco de crédito, a partir de dados de transações de clientes, presente em dois conjuntos de dados públicos, *Australian* e *German*, descritos anteriormente, na Seção 2.6.

O método proposto utiliza uma arquitetura profunda para as tarefas de seleção de variáveis e criação do modelos de classificação, a partir de um esquema processamento em paralelo que é executada em uma arquitetura distribuída do H2O[1], na qual os dados de treinamento são distribuídos pelo *cluster*, e cada nó opera em paralelo, em seu próprio conjunto de dados locais. Esse método foi dividido em quatro passos, como descrito a seguir.

No passo 1, o método inicia calculando a mediana da importância de cada variável, utilizando uma *Random Forest*. Neste caso, a *Random Forest* foi treinada usando os valores-padrão de cada um dos seus parâmetros.

No passo 2, cada uma das melhores variáveis é usada como entrada de uma DFN, utilizando a estratégia de validação cruzada com 20-*folds* para treinamento do modelo. Neste caso, a j^{th} validação cruzada terá a tupla $(F_j, A_j^{learn}, A_j^{validation})$, contendo os valores de importância do atributo, precisão de aprendizagem e precisão de validação, respectivamente.

O passo 3 é usado para cálculo do pontuação de cada variável, a partir da equação

$$F_i^{score} = \sum F_{ij} \text{ x } (A_j^{learn} + A_j^{validation}) \tag{3.6}$$

O passo 4 é responsável pela escolha das melhores variáveis, usando como critérios de seleção (1) os melhores scores medianos, (2) os melhores scores médios e (3) o menor desvio padrão.

O treinamento ocorreu sem a seleção de características nos dados originais. A rede profunda foi treinada por 10 épocas usando valores-padrão para parâmetros ocultos. Durante os experimentos, foram avaliadas as acurácias de modelos clássicos de classificação: SVM, CART, k-NN, *Nayve Bayes*, MLP e *Random Forest*. Além disso, foram avaliadas estratégias de seleção de atributos baseadas em filtros, como o Teste T, a análise de discriminantes lineares (LDA – *Linear Discriminant Analysis*), regressão logística e estratégias baseadas em *wrappers*, como Algoritmos Genéticos (GA) e *Particle Swarm Optimization* (PSO). Após o treinamento, esses modelos foram avaliados

[1] `http://docs.h2o.ai/h2o/latest-stable/h2o-docs/welcome.html`

segundo a acurácia alcançada e a abordagem proposta por Ha & Nguyen [2016] obteve os melhores resultados nos dois conjuntos de dados avaliados.

3.1.3 Balanceamento em nível de algoritmo

As abordagens em nível de algoritmo geralmente lidam com o problema de desbalanceamento de dados modificando diretamente o processo de aprendizagem do modelo. Geralmente, isso implica na adoção de alguma estratégia de aprendizado sensível ao custo [Khan et al., 2018; Chung et al., 2015; Wang et al., 2016] ou na seleção de funções de perda que otimizam as métricas de avaliação da classificação [Menon & Williamson, 2016; Yan et al., 2003; Zhao et al., 2011].

3.1.3.1 Aprendizagem sensível ao custo

A maioria dos modelos que minimizam o erro geral de classificação supõem implicitamente que diferentes tipos de erros de classificação têm a mesma importância e, como resultado, tendem a classificar em favor da classe majoritária [Khan et al., 2018]. Por outro lado, as abordagens de aprendizado sensíveis ao custo consideram que diferentes tipos de classificação incorreta podem estar associados a diferentes penalidades, apropriadas ao domínio do problema. Essas abordagens lidam com o problema de desbalanceamento de dados modificando diretamente o processo de aprendizado do modelo para aumentar a sua sensibilidade em favor da classe minoritária, o que geralmente melhora o desempenho do algoritmo como um todo [Haixiang et al., 2017].

Chung et al. [2015], por exemplo, propuseram um novo algoritmo de custos conhecidos que substitui a softmax tradicional por uma abordagem que leva em consideração as informações de custo nos estágios de treinamento e pré-treinamento de uma DFN, a fim de resolver o problema de classificação de imagens. Wang et al. [2016] introduziram uma nova função de perda chamada erro médio quadrático falso para o treinamento de uma DFN que usa dados desbalanceados para resolver problemas de classificação de documentos e imagens. Raj et al. [2016] apresentaram um método para resolver a classificação de imagens, que mescla diferentes métodos em uma nova abordagem que usa a pontuação de separabilidade de classe para classe para aumentar o desempenho em dados desbalanceados, usando uma CNN sensível ao custo. Um método de maior interesse para o nosso trabalho, contudo, é o CoSen de Khan et al. [2018], descrito em maiores detalhes a seguir.

CNN sensível ao custo Khan et al. [2018] propuseram uma abordagem sensível a custos para resolver o problema de classificação de imagens, chamado *CoSen*, que

modifica o processo de aprendizado de uma CNN para incorporar custos dependentes de classe durante o treinamento. Para tanto, os autores supõem a existência de uma CNN para resolver um problema de classificação envolvendo n classes, C_1, C_2, ..., C_n. Eles então introduzem uma matriz de erro dependente de classe Ω, onde $\Omega_{i,k}$ corresponde ao custo associado ao erro de prever uma instância da classe C_i como C_k. O valor $\Omega_{i,k}$ é um número real entre 0 a 1, sendo 1 o custo máximo possível. Dado que a CNN tem uma camada de decisão formada por n neurônios (correspondente às n classes), a estimativa $\hat{y}_i$ associada à i-ésima classe incorpora o erro dependente de classe através da função *softmax*, ou seja:

$$\hat{y}_i = \frac{\Omega_{j,i} \exp(o_i)}{\sum_k \Omega_{j,k} \exp(o_k)}, \tag{3.7}$$

em que o_i corresponde à i-ésima entrada da camada de decisão. Dado o vetor de estimativas $\hat{Y}$, os autores fornecem versões deriváveis para três populares funções de perda: o erro médio quadrado, a função de Hinge e a entropia cruzada. Finalmente, eles sugerem um algoritmo de otimização para o aprendizado dos pesos da CNN junto com a matriz Ω. Como resultado, o *CoSen* otimiza simultaneamente os parâmetros do classificador e aprende representações robustas de características, mesmo diante de desbalanceamento.

Embora esta técnica não tenha sido aplicada ao problema de crédito, ela é de simples adoção, uma vez que o problema de classificação de n classes poderia ser reduzido a um problema classificação binária de bons e maus pagadores. Nesse sentido, dada uma matriz de custo Ω e uma instância negativa x^-, o risco de classificação incorreta de x^- pode ser expresso como:

$$R(\hat{y} = +1 \mid x^-) = \Omega_{-1,+1} P(\hat{y} = +1 \mid x^-) \tag{3.8}$$

De forma similar, podemos expressar o risco de classificação incorreta de x^+ como:

$$R(\hat{y} = -1 \mid x^+) = \Omega_{+1,-1} P(\hat{y} = -1 \mid x^+) \tag{3.9}$$

Como resultado, o risco esperado da classificação binária pode ser expresso da seguinte forma:

$$R(c|X) = R(\hat{y} = +1 \mid x^-) + R(\hat{y} = -1 \mid x^+) \tag{3.10}$$

onde $\mathbf{X} = \{x_1, x_2, \ldots, x_N\}$ representa os dados de entrada do modelo.

De acordo com a teoria da decisão bayesiana, um classificador ideal deve selecionar

a classe c^*, $c^* \in \{-1, +1\}$, com o risco mínimo esperado:

$$c^* = \arg\min_c R(c|X) = \arg\min_c \mathbb{E}[\ell(c, \hat{Y}(X; \Theta, \Omega)] \tag{3.11}$$

onde $\ell(\cdot)$ indica um tipo de função de perda, Θ representa os pesos e vieses da rede, $\hat{Y}(X; \Theta, \Omega)$ denota as estimativas associadas a X, parametrizadas por Θ e Ω. Portanto, dada uma lista com rótulos reais $y = \{y_1, y_2, \ldots, y_N\}$ e as saídas previstas $\hat{y} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N\}$, podemos estimar $R(c|X)$ usando o risco empírico da seguinte forma:

$$R(c|X) = R_\ell(\Theta, \Omega) = \frac{1}{N} \sum_{j=1}^{N} \ell(y_j, \hat{y}_j(x_j; \Theta, \Omega)) \tag{3.12}$$

onde N é o tamanho da amostra de treinamento. Note que o valor de $R_\ell(\Theta, \Omega)$ é maior quando o modelo apresenta desempenho ruim durante o treinamento. Assim, para minimizar o risco de classificação, podemos encontrar o conjunto ótimo de parâmetros (Θ^*, Ω^*) que traz o mínimo valor possível de erro, usando a função de perda $\ell(\cdot)$. Portanto, o objetivo da otimização é dado por:

$$(\Theta^*, \Omega^*) = \arg\min_{\Theta, \Omega} \ell(y, \hat{y}(\Theta, \Omega)) \tag{3.13}$$

Entre as três funções adaptadas por Khan et al. [2018], a mais usada em aprendizagem profunda é a entropia cruzada (CE), uma função de perda que maximiza a proximidade da previsão com a saída desejada e pode ser expressa da seguinte forma:

$$\ell(y, \hat{y}) = -\sum_c (y_c \log \hat{y}_c) \tag{3.14}$$

No nosso caso, para o problema de classificação binária em que queremos distinguir entre bons e maus pagadores, podemos reescrever ℓ como:

$$\ell(y, \hat{y}) = -(y_{+1} \log \hat{y}_{+1} + y_{-1} \log \hat{y}_{-1}) \tag{3.15}$$

Logo, considerando os casos negativo e positivo separadamente, ℓ se torna:

$$\ell(y, \hat{y}) = (\ell(y_{-1}, \hat{y}), \ell(y_{+1}, \hat{y})) = (-\log y_{-1}, -\log y_{+1}) \tag{3.16}$$

Finalmente, para y_c incorporar o custo dependente da classe (Ω) relacionado à saída h, podemos usar a função softmax, como apresentado a seguir:

$$\ell(y_{-1}, \hat{y}) = -\log\left(\frac{\Omega_{p,-1}\exp(1-h)}{\Omega_{p,+1}\exp(h) + \Omega_{p,-1}\exp(1-h)}\right), \tag{3.17}$$

$$\ell(y_{+1}, \hat{y}) = -\log\left(\frac{\Omega_{p,+1}\exp(h)}{\Omega_{p,+1}\exp(h) + \Omega_{p,-1}\exp(1-h)}\right), \tag{3.18}$$

onde p indica a saída prevista pela rede. Assim, para $h < 0.5$ e $h >= 0.5$, temos:

$$\begin{aligned}\ell(y_{-1}, \hat{y}) = (&-\log\left(\frac{\Omega_{-1,-1}\exp(1-h)}{\Omega_{-1,+1}\exp(h) + \Omega_{-1,-1}\exp(1-h)}\right), \\ &-\log\left(\frac{\Omega_{-1,+1}\exp(h)}{\Omega_{-1,+1}\exp(h) + \Omega_{-1,-1}\exp(1-h)}\right))\end{aligned} \tag{3.19}$$

$$\begin{aligned}\ell(y_{+1}, \hat{y}) = (&-\log\left(\frac{\Omega_{+1,-1}\exp(1-h)}{\Omega_{+1,+1}\exp(h) + \Omega_{+1,-1}\exp(1-h)}\right), \\ &-\log\left(\frac{\Omega_{+1,+1}\exp(h)}{\Omega_{+1,+1}\exp(h) + \Omega_{+1,-1}\exp(1-h)}\right))\end{aligned} \tag{3.20}$$

3.1.3.2 Otimizações de métricas

Na literatura de aprendizagem de máquina, podemos encontrar muitos trabalhos que melhoram a taxa de classificação do modelo usando entropia cruzada e erro quadrático médio como funções de custo para otimizar o desempenho do classificador. No entanto, tanto para problemas de *credit scoring* quanto para muitos outros problemas de classificação binária do mundo real, a AUC é uma medida de desempenho mais significativa, pois indica quão bom previsor é um modelo em sua capacidade de distinguir diferentes instâncias. Ou seja, quão bons são os escores fornecidos por estes modelos para estabelecer a classe da instância? Como resultado, é usual que esses modelos de classificação sejam avaliados usando AUC, embora saibamos que minimizar a entropia cruzada ou o erro quadrado médio não maximiza necessariamente esta métrica [Yan et al., 2003; Menon & Williamson, 2016].

Otimização de AUC O AUC empírico corresponde à Equação 3.21:

$$A(\hat{y}) = \sum_{i=1}^{I}\sum_{k=1}^{K} \mathbb{1}(\hat{y}(x_i^+;\Theta) > \hat{y}(x_k^-;\Theta)) \tag{3.21}$$

onde $\hat{y}$ é um estimador parametrizado por Θ que é aplicado a exemplos positivos x_i^+ e negativos x_k^-.

Como observado por Menon & Elkan [2011], esta equação não é fácil de otimizar diretamente. Uma alternativa para lidar com esta dificuldade é buscar funções aproximadas. Por exemplo, Yan et al. [2003] observam que a AUC é exatamente igual à estatística padrão de Wilcoxon Mann-Whitney (WMW) e introduzem uma nova função objetivo, que representa uma aproximação diferenciável da WMW. O problema é então tratado como um problema de ranking (bipartido).

Contudo, como observam Menon & Elkan [2011], o problema de maximização direta do AUC pode ser descrito como na Equação 3.22.

$$\min \sum_{i=1}^{I} \sum_{k=1}^{K} \mathbb{1}(\hat{y}(x_i^+; \Theta) - \hat{y}(x_k^-; \Theta) \leq 0) \tag{3.22}$$

Esta equação corresponde a uma função de perda relacionada com a diferença entre dois escores, o que pode ser otimizado eficientemente usando gradiente descendente estocástico se os estimadores forem diferenciáveis. Em cada iteração, o gradiente pode ser calculado sobre um par de exemplos tomados de forma aleatória. Em Menon & Elkan [2011], por exemplo, os autores usam estimadores lineares que operam sobre entradas transformadas por meio de fatoração matricial para reduzir o custo de representação.

Esta ideia é basicamente a mesma usada no arcabouço RankNet [Burges et al., 2005]. Neste caso, é usada uma rede neural siamesa[2]. Um par com um exemplo positivo e outro negativo é dado para a rede que estima o quanto a diferença entre os escores probabilísticos entre os exemplos é diferente de 1. Na prática, após ser treinada com múltiplos pares, a rede aprende a estimar a probabilidade de um exemplo ser positivo. Assim, a função de perda de treino corresponde à Equação 3.22, onde cada exemplo é transformado pelas camadas de representação da rede siamesa em lugar de uma fatoração matricial, como em Menon & Elkan [2011].

[2]Rede neural aplicada a um par de instâncias. Cada elemento do par é aplicado à rede e, então, uma função de perda é calculada sobre o par, de forma que o gradiente resultante modifique os pesos da rede antes que um novo par (ou lote de pares) seja processado. O termo siamesa se refere à possibilidade de se imaginar a arquitetura deste modelo como duas redes gêmeas conectadas por uma função perda.

3.2 Abordagens profundas baseadas em dados desbalanceados

Em nossa revisão da literatura encontramos abordagens profundas que, diferente daquelas apresentadas na Seção 3.1, não utilizaram o passo de pré-processamento para balanceamento do conjunto de treinamento, lidando com a distribuição de classes real. Nesta seção, apresentamos essas abordagens.

Pontuação de crédito usando DBN e dados de *swap* Luo et al. [2017] propuseram um método que, pela primeira vez na literatura, utilizou dados de contratos de *Credit Default Swaps* (CDS) para classificação de risco de crédito de empresas. Dentre os dados utilizados, estão a taxa de recuperação e setor de atuação da empresa contrante. Diferente das abordagens tradicionais que adotam as classes alvo bom e mau, a nova abordagem utiliza três classes alvo: A, B e C, onde as empresas das categorias B e C são consideradas abaixo da linha para investimento.

O método adotado para gerar o modelo de classificação inicia com o pré-processamento dos dados de contratos de CDS, onde são aplicadas técnicas clássicas de transformação de atributos para entradas das redes neurais. Em seguida, os dados transformados são usados como entrada de uma DBN, que usa a estratégia de validação cruzada de 10-*fold* para treino e teste do modelo. Por fim, o resultado do modelo é avaliado contra os métodos de regressão logística multinomial (*Multinomial Logistic Regression*–MLR), MLP e SVM, usando como métricas de avaliação de desempenho a AUC e a Acurácia.

A base de dados utilizada é composta por contratos de 661 empresas de regiões distintas, como América do Norte, Ásia e Europa, atuantes em diferentes setores, como indústria e tecnologia. A distribuição das instâncias por classe é dada por $A = 275$ (41,60%), $B = 374$ (56,58) e $C = 12$ (1,81%). Cada empresa é representada por 11 atributos, sendo 9 numéricos e 2 categóricos. Na fase de pré processamento, atributos numéricos são normalizados e atributos categóricos são transformados em numéricos binários, usando a abordagem de *one-attribute-per-value*, onde um atributo com k valores é transformado em k atributos binários, conforme proposto em Breiman [2017].

Pontuação de crédito usando CNN e LSTM Neagoe et al. [2018] propuseram uma arquitetura profunda de CNN para gerar modelos de classificação de risco de crédito, utilizando as bases de dados *German* e *Australian* para treinamento e teste dos modelos. A arquitetura do método proposto é composta por 13 camadas:

- Uma camada de entrada de dados, com $\boldsymbol{m}$ neurônios, um para cada atributo do cliente;
- Três camadas principais, que seguem o padrão *Long Short-Term Memory* (LSTM), com $\boldsymbol{N}$ neurônios cada uma. Além disso, cada camada principal possui uma camada de *dropout* secundária com uma taxa de probabilidade $\boldsymbol{D}$;
- Uma camada totalmente conectada (FC) com N neurônios, associada a duas camadas secundárias: uma de *dropout* com probabilidade D, e uma camada de *Relu*;
- Uma camada de classificação baseada no *Softmax* com 2 neurônios;

Como *baseline*, os autores adotaram a estratégia de implementação de uma rede MLP clássica, composta por 8 camadas:

- Uma camada de entrada com $\boldsymbol{m}$ neurônios, um para cada atributo do cliente;
- $\boldsymbol{H}$ camadas ocultas, com $\boldsymbol{NH}$ neurônios cada uma, assumindo $max(H) = 6$;
- Uma camada de saída com 2 neurônios;

Para treinar a CNN, Neagoe et al. [2018] utilizaram o SGD com *momentum* e para o treinamento da MLP foi usado o *Broyden-Fletcher-Goldfarb-Shannon* (BFGS) e o algoritmo *quasi-Newton* [Schraudolph et al., 2007]. Segundo os autores, BFGS foi escolhido por apresentar desempenho superior ao tradicional método *Levenberg-Marquardt backpropagation*, conforme descrito por Lahmiri [2011].

Os autores definiram como estratégia de avaliação o uso das seguintes métricas de desempenho: acurácia, MAR e FAR, descritas na Seção 2.1.2. Como resultado, a CNN obteve melhores resultados que a MLP nos dois conjuntos de dados avaliados.

Pontuação de crédito com CNN e Map Reduce. Yifei [2016] propôs um método para gerar um modelo de previsão de risco de crédito a partir de uma CNN de treinamento paralelo. A CNN foi treinada usando uma arquitetura *Map reduce* com propósito de alcançar alta eficiência computacional e alto poder de previsão.

A base de dados utilizada para treino e avaliação dos modelos foi coletada do departamento de crédito de duas filiais do Banco Industrial e Comercial da China – ICBC. As amostras contêm dados de empréstimos de curto prazo, realizados na indústria de manufatura, com prazo máximo de 12 meses, aprovados em janeiro de

2013. Ao todo, foram coletadas 636 amostras representando uma movimentação de RMB[3]$ 16 bilhões em empréstimos.

A arquitetura utilizada para suportar o processamento em paralelo é baseada em 4 grupos de módulos:

- *Storage system*: responsável pelo armazenamento dos dados de cada documento, extração de dados de características e índices;
- *Feature extraction system*: responsável pela extração de características do documento, carregando os dados na estrutura do HDFS;
- *Index system*: responsável pela construção do índice para o banco de dados de características extraídas;
- *Query system*: responsável pela execução de consultas do sistema;

O autor definiu a arquitetura da CNN com uma camada de entrada de dados, duas camadas de convolução (C1 e C2), duas camadas de *pooling* (S1 e S2) e uma camada de saída, onde:

- A camada de convolução $C1$ é formada por três *kernels*, com comprimentos de 15 pontos de amostragem, onde a entrada é o segmento candidato de avaliação de risco de crédito para 116 pontos de amostragem, gerando três vetores de características com comprimento de 102 pontos de amostragem;
- A camada de *pooling* $S1$ executa o agrupamento dos vetores de características gerados pela camada $C1$. Esses vetores são compactados em 51 pontos de amostragem;
- A camada de convolução $C2$ é formada por três *kernels*, com comprimentos de 22 pontos de amostragem e gera como saída 9 vetores de características com tamanhos de 30 pontos de amostragem;
- A camada de *pooling* $S2$ executa novo agrupamento e envia os dados para a camada de saída calcular o resultado da classificação.

O treinamento da CNN é formado por 4 passos distintos:

Passo 1: etapa onde são definidos os pesos W e os limiares δ, um número aleatório próximo a zero; nesta etapa, também é inicializada a CNN.

[3]Moeda oficial da República Popular da China

Passo 2: Passo em que são sorteadas cem amostras do conjunto de dados de treinamento para servirem de entrada da CNN, juntamente com seus respectivos valores-alvo.

Passo 3: Passo em que ocorre o cálculo do vetor de saída atual e a comparação desse vetor com o vetor de saída do alvo previsto. Com isso, ocorre o ajuste dos valores de erro E, peso W e do valor de *threshold* δ, a partir das seguinte equações:

$$\Delta W_{jk}(n) = \frac{\alpha}{1+l} \times (\Delta W_{jk}(n-1)+1) \times \delta_k \times h_j \tag{3.23}$$

$$\delta_k = h_j * (1-h_j) \sum_{k=0}^{M-1} \delta_k W_{jk} \tag{3.24}$$

onde α é a taxa de aprendizagem; j e k são neurônios nas camadas oculta e de saída, respectivamente; M é o número de neurônios na camada de saída; h_j é o vetor de saída na camada oculta; W é o peso a ser ajustado; δ é o valor de limiar a ser ajustado.

Passo 4: passo que verifica se os requisitos de precisão foram atendidos, usando:

$$E = \frac{1}{2} \sum_{k=0}^{n-1} (d_k - y_k)^2 \leq \epsilon \tag{3.25}$$

onde E é o total da função de erro; y_k é a saída do vetor; d_k é o vetor alvo. Caso os requisitos não sejam atendidos, o algoritmo retorna ao *passo 3*. Caso contrário, o treinamento é encerrado e o valor do limiar é salvo. Em caso de novo treinamento, o peso e o valor do limiar são carregados do arquivo e não é necessária a repetição do *passo 1*.

Para treinamento da CNN usando *Map reduce* foi necessário: (i) coletar e analisar os dados bancários dos clientes avaliados; (ii) construir o sistema de índices adequado a um modelo de risco de crédito, usando informações pessoais básicas, como nível de educação, renda e outros índices relacionados; (iii) selecionar amostras de dados com maus e bons pagadores; (iv) selecionar os índices mais relevantes para o modelo, a partir do método de ganho de informação; (v) criar a CNN paralela através dos dados do índice após *screening*; e, por fim, (vi) avaliar o desempenho dos modelos gerados;

Como resultado do seu trabalho, Yifei [2016] mostrou que sua abordagem de CNN com treinamento distribuído alcançou melhores resultados que a regressão logística e uma MLP. Avaliando a eficiência computacional dos modelos, o método proposto consumiu apenas 5.9 segundos durante seu treinamento, demostrando a maior eficiência do processamento em paralelo.

Pontuação de crédito baseada em auto-codificadores e programação genética. Tran et al. [2016] propuseram um método híbrido para previsão de risco de crédito, baseado em algoritmos de aprendizagem profunda e programação genética. A motivação por trás da combinação entre essas duas abordagens é baseada nas vantagens e desvantagens dos algoritmos de aprendizagem de máquina baseados em função (ex. SVM e Redes Neurais) e indução (ex. Árvore de Decisão).

No geral, os algoritmos baseados em função geram modelos de classificação melhores, em termos de precisão, comparado aos algoritmos baseados em indução. Por outro lado, os algoritmos baseados em indução geram modelos interpretáveis e próximos da linguagem humana, facilitando a tomada de decisão. Nesse sentido, os autores propuseram um modelo híbrido denominado *Boosted Deep Network* que é baseado no algoritmo funcional *Stacked auto-enconder* (SAE) e no algoritmo indutivo de programação genética. Nesse caso, a pilha de auto-codificadores configurada de acordo com os parâmetros apresentados na Tabela 3.6. A intuição por trás da combinação desses dois algoritmos é simples. Se o modelo genético não conseguir classificar (ou explicar) os dados, então a rede neural é requisitada para tratar os dados e realizar a classificação. Os dados de entrada da rede neural são formados pela combinação dos dados brutos com os dados transformados pela programação genética.

A ideia de um algoritmo genético é baseada na evolução e seleção natural de uma população. À medida que as gerações avançam ao longo do tempo, somente os melhores indivíduos geneticamente fortes sobrevivem e, dessa forma, a população vai se adaptando ao ambiente no decorrer do tempo. As melhores informações genéticas dos indivíduos passam de geração em geração por meio de processos de cruzamento e mutações. Esse processo é repetido várias vezes e somente as características dos bons indivíduos (ou as melhores soluções) permanecem ativas.

No caso do problema de previsão de risco de crédito, são mantidas as 5 melhores soluções para bons pagadores e as 5 melhores soluções para os maus pagadores. O objetivo do algoritmo genético é encontrar padrões nos dados e transformá-los em regras na forma de uma árvore condicional (IF-THEN). As regras são limitadas a utilizarem os operadores $=$; $<$; $<=$; $>$; $>=$; $+;-$; $*;/$; *and*; *or*; *not* com propósito de manter as regras simples e interpretáveis. A Tabela 3.5 mostra os parâmetros utilizados para treinar o algoritmo genético.

Os experimentos foram realizados com duas bases de dados públicas, *Australian* e *German credit*, apresentadas na Seção 2.6. Os resultados do método proposto por Tran et al. [2016] foram comparados com os algoritmos *Adaboost*, CART, C4.5, SVM, *Logistc Regression*, KNN, Rede Neural simples e Algoritmo Genético. A métrica de avaliação utilizada foi o erro médio dos modelos. O método proposto obteve os melhores

Tabela 3.5: Configuração de parâmetros da programação genética.

PARÂMETROS	VALORES
Tamanho da população	50
Conjunto de funções	=; <; <=; >; >=; +;−; ∗;/; *and*; *or*; *not*
Conjuntos de terminais	0,0.5,1,1.5,...,14
Número de gerações	1000
Taxa de cruzamento	0.95
Taxa de mutação	0.2

Tabela 3.6: Configuração de parâmetros da SAE.

PARÂMETROS	VALORES
Nós das camadas ocultas	[200, 150, 180, 80, 100, 90, 38]
Coeficiente regular (λ)	0.02
Limiar de esparsidade (ψ)	0.2
Coeficiente de esparsidade (β)	3
Função de erro	erro quadrático médio
Função de erro de esparsidade	Kullback-Leibler divergence
Função de otimização de erro	lbfgs

resultados entre os métodos avaliados.

3.3 Considerações Finais

Neste capítulo, apresentamos um resumo dos trabalhos relacionados ao nosso, encontrados em nossa revisão da literatura. A Tabela 3.7 sintetiza estes trabalhos, em ordem decrescente de ano de publicação, caracterizando, para cada um deles, o tipo de problema resolvido pelo método (Aplicação), o tipo de arquitetura de aprendizagem usada no trabalho (Arquit.), o nível de balanceamento de dados (Nível de Balanc.), a técnica utilizada para o balanceamento (Técnica de Balanc.), as métricas utilizadas para avaliação (Métricas) e uma indicação se eles consideram ou não o uso de funções assimétricas de perda (Perda Assim.), que garantam a equivalência entre os problemas de ranking bipartido e classificação binária.

Tabela 3.7: Resumo dos trabalhos relacionados

Método	Aplicação	Arquit.	Nível de Balanc.	Técnica de Balanc.	Métricas	Perda Assim.
Addo et al. [2018]	Credit scoring	DFN	Dados	SMOTE	AUC e RMSE	-
Kvamme et al. [2018]	Credit scoring	CNN	Dados	Over	AUC e Acurácia	-
Neagoe et al. [2018]	Credit scoring	CNN	-	-	Acurácia	-
Yu et al. [2018]	Credit scoring	DBN	Dados	Over / Under	Acurácia	-
Zhu et al. [2018]	Credit scoring	CNN	Atributos	Relief	AUC e KS	-
Khan et al. [2018]	Classificação de imagens	CNN	Algoritmo	Sensível ao custo	Acurácia	-
Luo et al. [2017]	Credit scoring	DBN	-	-	AUC e Acurácia	-
Ha & Nguyen [2016]	Credit scoring	DFN	Atributos	H2O score	Acurácia	-
Raj et al. [2016]	Classificação de imagens	CNN	Algoritmo	Sensível ao custo	Acurácia	-
Tran et al. [2016]	Credit scoring	Auto-Codif.	-	-	MSE	-
Yifei [2016]	Credit scoring	CNN	-	-	Acurácia	-
Wang et al. [2016]	Classificação de imagens	DFN	Algoritmo	Sensível ao custo	Acurácia	-
Menon & Elkan [2011]	Previsão de Links	Rasa	Algoritmo	Otimização de métricas	AUC	-
Burges et al. [2005]	Ranking	Rasa	Algoritmo	Otimização de métricas	AUC	-
Yan et al. [2003]	Classificação Binária	Rasa	Algoritmo	Otimização de métricas	AUC	-
Nossos métodos	Credit scoring	DFN	Algoritmo	Sensível ao custo	AUC e KS	X

Ao analisarmos a Tabela 3.7, considerando apenas os trabalhos criados para resolver o problema de *credit scoring*, podemos verificar que:

1. Apenas nove trabalhos utilizam arquiteturas profundas para resolver o problema de pontuação de crédito, o que indica a necessidade de mais pesquisas para o avanço na solução desse tipo de problema;

2. Os trabalhos observados foram publicados entre 2016 e 2018, indicando que só recentemente o problema de *credit scoring* foi abordado com técnicas de aprendizagem profunda;

3. Avaliando as estratégias de balanceamento utilizadas, verificamos que três trabalhos utilizam técnicas baseadas em dados e dois trabalhos usaram técnicas

baseadas em atributos. Outros quatros trabalhos não usam quaisquer técnicas de balanceamento de dados. Nenhum dos trabalhos que resolvem *credit scoring* utiliza balanceamento em nível de algoritmo;

4. Ao avaliarmos o tipo de balanceamento da base de dados de treinamento dos modelos, verificamos que três trabalhos utilizam *oversampling* (sendo dois com replicação aleatória de amostras e um com SMOTE), um trabalho utiliza *undersampling* e seis trabalhos não aplicam técnica de balanceamento, utilizando a distribuição real das classes da base de treino;

5. Ao avaliarmos os tipos de arquiteturas profundas utilizadas nesses modelos, verificamos que quatro trabalhos utilizam arquiteturas de CNN, dois trabalhos implementam arquiteturas de DBN, dois trabalhos adotam a arquitetura DFN e um trabalho utiliza auto-codificadores;

6. Ao estudarmos as métricas utilizadas para a avaliação dos modelos de *credit scoring*, verificamos que Acurácia, AUC, KS, RMSE e MSE formam o grupo de métricas comumente usadas.

Por fim, verificamos que apenas nosso trabalho aborda o problema de *credit scoring* usando aprendizagem profunda com balanceamento em nível de algoritmo através de funções de perda assimétrica.

Capítulo 4

Abordagem assimétrica profunda para risco de crédito

Neste capítulo, propomos um método de aprendizagem ponderada assimétrica profundo, em inglês *Deep Asymmetric Weighted Learning - DAWL*, no qual lidamos com problemas de desbalanceamento de classes no contexto de *Credit Scoring*, modificando o processo de aprendizado de uma DFN, a fim de otimizar iterativamente os pesos da rede e custos sensíveis aos erros de classificação. Para tanto, apresentamos duas funções assimétricas exponenciais efetivas para treino e robustas a desbalanceamento. Estabelecemos a consistência destas funções para classificação sensível a custos e propomos um algoritmo, baseado em uma abordagem de aprendizagem alternada, para estimar todos os hiper-parâmetros necessários ao uso das mesmas.

Nas próximas seções, descrevemos a abordagem de aprendizado sensível a custos para arquiteturas profundas e as duas funções assimétricas de custo propostas.

4.1 Introdução

O problema de pontuação de crédito pode ser visto tanto como um problema de classificação binária quanto de ranking bipartido. Em ambos os casso, temos um conjunto de exemplos de treino associados a um rótulo binário, positivo ou negativo. No primeiro caso, pretendemos separar bons de maus pagadores. No segundo, o algoritmo precisa aprender a ordem relativa entre instâncias. Isso pode ser feito, por exemplo, como em estratégias *pointwise*, onde uma função de pontuação assinala para cada instância um número real, tal que instâncias positivas recebem pontuações maiores que instâncias negativas. Alternativamente, o algoritmo pode tentar aprender diretamente a ordem

relativa, com base nos escores, como em abordagens *pairwise* e *listwise* [Cao et al., 2007].

Soluções para o segundo problema são geralmente mais complexas de um ponto de vista de supervisão, já que o custo do aprendizado de ordem tende a ser maior que o de distinção de classes. Por outro lado, este é um objetivo particularmente útil, pois, em pontuação de crédito, temos interesse em saber quem são os melhores e os piores pagadores para definir diferentes políticas de crédito (por exemplo, conceder vantagens maiores ou menores, como diferentes limites de empréstimo). Isso se reflete nas métricas comumente usadas na literatura de crédito, como o AUC, que pode ser compreendida como uma métrica de ranking [Menon & Williamson, 2016; Yan et al., 2003]. Além disso, tal processo é mais robusto para casos desbalanceados uma vez que é mais simples evitar uma supervisão com casos sub-representados. Por exemplo, no treinamento, as instâncias podem representar pares de clientes, o que garante uma quantidade enorme de exemplos para supervisão. O inconveniente, neste caso, é o custo de treino, dado o número de pares usados.

Para ambos os problemas, violações cometidas pelo previsor durante o treino são penalizadas por uma função de perda ℓ e o risco da função de pontuação é estimado como a penalidade esperada de acordo com ℓ. O valor de uma função de perda usada nestes métodos de aprendizagem está associado a propriedades, tais como a convexidade, que facilitem a aplicação de técnicas eficientes, necessárias para processar imensas bases de dados de forma efetiva. No contexto de classificação binária, onde os custos da classificação são assimétricos e/ou a distribuição de classes é desbalanceada, muitos trabalhos têm mostrado consistentemente que funções de perda assimétricas apresentam melhor desempenho em vários cenários de aplicação [Scott, 2012; Beijbom et al., 2014; Khan et al., 2018]. Além disso, quando estas funções podem ser usadas de forma equivalente para o problema de ranking bipartido, temos simultaneamente facilidade de aprendizado e robustez a desbalanceamento.

4.2 Notação e Definições

Nesta seção introduzimos a definições usadas ao longo deste capítulo, seguindo de forma geral a notação de Menon & Williamson [2016].

Seja $(\mathcal{X}, \mathcal{Y})$ um conjunto de instâncias e alvos com distribuição $\mathcal{X} \times \{-1, 1\}$. Uma função de perda para classificação binária ℓ é uma função (mensurável) $\ell : \{-1, +1\} \times \mathbb{R} \to \mathbb{R}_+$. Dada a função de perda ℓ, $\ell_+(v) = \ell(1, v)$ e $\ell_-(v) = \ell(-1, v)$ denotam as *perdas parciais* individuais, ou seja, as perdas associadas aos exemplos positivos e

negativos. Também podemos denotar tal função ℓ em termos dos seus componentes parciais, por meio de uma tupla:

$$\ell(v) = (\ell_-(v), \ell_+(v)) \tag{4.1}$$

Defini-se o risco-ℓ condicional $L(\cdot, \cdot; \ell) : [0, 1] \times \mathbb{R} \to \mathbb{R}_+$ como:

$$L(\eta, s; \ell) = \eta \ \ell_+(s) + (1 - \eta) \ \ell_-(s) \tag{4.2}$$

onde $\eta \in [0, 1]$, indica o peso relativo das perdas parciais de ℓ, ℓ_+ e ℓ_-, quando aplicadas ao escore s, $s \in \mathbb{R}$. Por sua vez, o *risco-ℓ condicional (Bayesiano) ótimo* é definido como:

$$L^*(\eta; \ell) = \inf_{s \in \mathbb{R}} L(\eta, s; \ell) \tag{4.3}$$

onde inf indica o infimum[1].

Nós dizemos que ℓ é *simétrica* se, para todo $y \in \{-1, +1\}$ e $v \in \mathbb{R}$, $\ell(y, v) = \ell(-y, -v)$, ou seja, erros (acertos) simétricos são afetados da mesma forma. Nós chamamos ℓ de uma *função de margem* se $\ell(y, z) = \phi(yz)$ para $\phi : \mathbb{R} \to \mathbb{R}_+$, ou seja, a decisão sobre o custo da estimativa está associada ao produto entre a classe real y e a estimativa z. Este produto representa a posição em relação a uma hiper-superfície, a "margem". Uma perda é simétrica se e somente se é uma funçao de margem [Menon & Williamson, 2016].

De forma geral, como demonstrado por Bartlett et al. [2006], dizemos que uma função de perda ℓ é *calibrada para classificação* se para todo $\eta \in [0, 1]$, $\eta \neq \frac{1}{2}$:

$$L^*(\eta; \ell) < \inf_{s : s(2\eta - 1) \leq 0} L(\eta, s; \ell) \tag{4.4}$$

Intuitivamente, se a função de perda é calibrada, seu risco ótimo é menor que seu risco observado para casos de erro. Também notamos que toda a previsão ótima tem o mesmo sinal de $2\eta - 1$. A calibração nos permite estabelecer uma mínima consistência da função de perda, ou seja, ela pune erros mais que acertos.

Uma propriedade similar é a *aversão a palpites*, introduzida em Beijbom et al. [2014]. Seja ℓ_v uma função perda assimétrica para um problema de classificação com n instâncias e m classes $\ell_v : \mathbb{N}^n \times \mathbb{R}^{n,m} \to \mathbb{R}_+$, s_y o vetor de escores onde y-ésimo score

[1] O infumum de um sub-conjunto $\mathcal{S}$ de um conjunto parcialmente ordenado $\mathcal{T}$ é o maior elemento de $\mathcal{T}$ menor ou igual a todos os elementos de $\mathcal{S}$. Embora similar ao conceito de mínimo, o conceito de infimum é preferível aqui porque há conjuntos sem mínimo (ex: $\mathbb{R}_+$) que tem infimum (zero) em um super conjunto apropriado (ex: $\mathbb{R}$).

(correspondente à classe y) é o maior em valor absoluto (ou seja, o classificador que gerou o escore s_y, selecionou a classe y), e g é o vetor de escores para a qual todas as classes são igualmente prováveis. A função ℓ_v é aversa a palpites se e somente se:

$$\ell_v(s_y) < \ell_v(g) \tag{4.5}$$

Ou seja, uma função de perda aversa a palpite é a que encoraja classificações corretas a palpites arbitrários. Os pontos g são aqueles de máxima incerteza para o classificador. Como todas as classes são igualmente prováveis para estes escores, há um empate que induz o classificador a escolher de forma arbitrária uma das classes. Enquanto esta é a decisão ótima quando as probabilidades das classes são idênticas, ela é subótima quando erros são ponderados de forma diferente, como pode ocorrer em funções assimétricas.

Seja $\ell_P : \{-1, +1\} \times [0, 1] \to \mathbb{R}$ uma *função de perda para estimativas de probabilidades de classes*. Ela é chamada *própria* se seu risco-ℓ condicional é otimizado para prever a probabilidade subjacente:

$$(\forall \eta, \eta' \in [0, 1])\ L(\eta, \eta; \ell_P) \leq L(\eta, \eta'; \ell_P) \tag{4.6}$$

L é chamada *própria estrita* se a desigualdade é estrita:

$$(\forall \eta, \eta' \in [0, 1])\ L(\eta, \eta; \ell_P) < L(\eta, \eta'; \ell_P) \tag{4.7}$$

Dizemos que a função de perda ℓ é *própria (estrita) composta* se há alguma função inversível $\psi : [0, 1] \to \mathbb{R} \cup \{-\infty, +\infty\}$ tal que a função de perda para estimativas de probabilidades de classes $\ell_P(y, u) = \ell(y, \psi(u))$ é própria (estrita) [Menon & Williamson, 2016].

Seja $\mathcal{L}_{PEC}$ o conjunto de todas as funções próprias (estritas) compostas. Menon & Williamson [2016] fornecem uma completa análise das $\mathcal{L}_{PEC}$ aplicadas ao problema geral de ranking bipartido, observando, entre outros resultados, (i) as conexões entre ranking bipartido e estimativas de probabilidades de classes, (ii) a equivalência entre diferentes tipos de risco e (iii) o projeto de funções muito adequadas ao problema de maximizar a acurácia no topo do ranking. Em particular, os autores mostram que funções de perda da família $\mathcal{L}_{PEC}$ podem ser usadas para guiar processos de minimização tanto de problemas de ranking bipartido quanto classificação binária. Estas funções são interessantes por serem adequadas para otimização baseada em algoritmos de gradiente com complexidade linear no número de exemplos de treino quando comparadas a outros métodos usados em ranking, como os que operam em pares de exemplos.

Entre várias funções estudadas por Menon & Williamson [2016], temos um interesse especial pela função de perda exponencial dada na Equação 4.8:

$$\ell(v; p) = (\frac{1}{p}e^{pv}, e^{-v}) \tag{4.8}$$

onde $v, p \in \mathbb{R}$, $p \geq 1$. Para $p = 1$, ela corresponde à função usada no algoritmo AdaBoost [Freund & Schapire, 1997]. Logo, ela é uma versão mais geral da perda usada no AdaBoost e, ao longo deste texto, será chamada de *perda p-Classification*.

A perda p-Classification foi estudada por Ertekin & Rudin [2011] no contexto de previsores lineares usados em soluções Bayesinas ótimas (risco empírico). Em particular, as autoras mostraram que o risco do algoritmo de ranking bipartido p-Norm-Push, baseado na perda p-Classification, é equivalente ao risco correspondente ao algoritmo de classificação binária p-Classification, que usa a mesma função de perda.

Por sua vez, Scott [2012] provou que a função é calibrada para classificação. Para tanto, ele estendeu a Teoria de Calibração de Bartlett et al. [2006] para o caso assimétrico, sensível a custo. Beijbom et al. [2014] mostraram que esta função é aversa a palpites, mesmo quando aplicada ao problema com múltiplas classes.

Em seu trabalho, Menon & Williamson [2016] observam que, dada a equivalência com o algoritmo p-Norm-Push, fica evidente que esta função maximiza a acurácia no topo do ranking focando mais em instâncias com alto escore. Ela faz isso por meio da assimetria introduzida pelo parâmetro p, o que possibilita a definição de toda uma família de funções de ordenação do topo do ranking que são também apropriadas para problemas de classificação binária.

Baseado na p-Classification, a seguir, apresentamos duas funções assimétricas de perda para lidar com o problema de desbalanceamento de classes, onde o equilíbrio da amostra pode ser realizado por conjuntos de dois e quatro pesos.

4.3 Função assimétrica de perda AWT

A primeira função que propomos acrescenta um peso $\frac{1}{q}$ na perda parcial positiva da p-Classification. De fato, esta variante é sugerida como uma função a ser analisada como trabalho futuro em Ertekin & Rudin [2011], a fim de lidar com a precisão e a revocação. Chamamos esta função AWT (de *Asymmetric Weighted approach with Two weights*). Ela é dada na Equação 4.9.

$$\ell_{AWT}(v; p, q) = (\frac{1}{p}e^{pv}, \frac{1}{q}e^{-qv}) \tag{4.9}$$

onde $v \in [-1, 1]$ é a estimativa do previsor e $\frac{1}{p}$ corresponde simultaneamente à penalidade aplicada aos falsos negativos, bem como à recompensa aplicada aos acertos de negativos. De forma similar, $\frac{1}{q}$ corresponde simultaneamente à penalidade aplicada aos falsos positivos, bem como à recompensa aplicada aos acertos de instâncias positivas.

Note que, mesmo com seus parâmetros de sensibilidade a custo, a função AWT é própria (estrita) composta, aversa a palpite e calibrada para classificação sensível a custo, como descrito nos Lemas 4.3.1, 4.3.2 e 4.3.3.

Lema 4.3.1. *Para quaisquer $p, q > 0$, seja ℓ a função de perda AWT, dada na Equação 4.9. Então, $\ell \in \mathcal{L}_{PEC}$, ou seja, ela é própria (estrita) composta.*

Demonstração. A função de perda ℓ é *própria (estrita) composta* se há alguma função inversível $\psi^{-1} : [0, 1] \to \mathbb{R} \cup \{-\infty, +\infty\}$ tal que a função de perda para estimativas de probabilidades de classes $\ell_P(y, u) = \ell(y, \psi(u))$ é própria (estrita). Conforme Menon & Williamson [2016], se ℓ é diferenciável, a condição acima implica que:

$$\begin{aligned}(\forall v \in \mathbb{R})\psi^{-1}(v) &= \frac{1}{1 - \frac{\ell'_+(v)}{\ell'_-(v)}} \\ &= \frac{1}{1 + e^{-(p+q)v}}\end{aligned}$$

Como $\psi^{-1}(v) = \frac{1}{1+e^{-(p+q)v}}$ é inversível (com inversa dada por $\psi(v) = \frac{1}{p+q} \log \frac{v}{1-v}$), ℓ é própria (estrita) composta.

□

Lema 4.3.2. *A função de perda AWT é aversa a palpites. Mais precisamente, para $p, q \geq 1$, dada estimativas v corretas produzidas por um estimador, a função $\ell_{AWT}(v)$ é aversa a palpite se e somente se $\ell_{AWT}(v) < \ell_{AWT}(c)$ onde c é o conjunto de todos os pontos que representam palpites.*

Demonstração. Para uma rede neural com ativações reais normalizadas na última camada com escores entre -1 e 1, palpites mapeiam para 0, o escore de maior incerteza. Assim:

$$(\forall v \in [-1, 1])\ \ell_{AWT}(v) < \ell_{AWT}(0) \tag{4.10}$$

é o esperado para os casos de classificação correta de acordo com a definição de aversão a palpites. Supondo que o previsor (por exemplo, uma rede neural) classifica cor-

retamente n_+ instâncias positivas e n_- instâncias negativas, a Equação 4.10 resulta em:

$$\frac{n_+}{q}e^{-vq} + \frac{n_-}{p}e^{vp} < \frac{n_+}{q} + \frac{n_-}{p} \tag{4.11}$$

que é uma inequalidade verdadeira, uma vez que (i) $e^{-vq} < 1$ para as n_+ instâncias corretas pois, neste caso, $v > 0$ e $q \geq 1$; e (ii) $e^{vp} < 1$ para as n_- instâncias corretas pois, neste caso, $v < 0$ e $p \geq 1$.

□

Lema 4.3.3. *A função de perda AWT é calibrada para classificação sensível a custo.*

Demonstração. Uma função de perda para classificação *sensível a custo* com parâmetro $\alpha \in (0, 1)$ pode ser escrita como:

$$U_\alpha(v) = (\alpha \mathbb{1}_{v>0}, (1-\alpha)\mathbb{1}_{v\leq 0}) \tag{4.12}$$

onde $\mathbb{1}_c$ é 1 se o critério c verdadeiro, caso contrário é 0. Embora seja possível formular muitas diferentes funções sensíveis a custo, a parametrização dada é conveniente porque um classificador ótimo é $sinal(\eta(x) - \alpha)$ onde $\eta(x) = P(Y = 1|X = x)$ [Scott, 2012].

Embora fosse desejável minimizar o risco de U_α, este não pode ser feito de forma eficiente. Isto nos leva a minimizar o risco empírico L de uma função substituta (*surrogate function*) com propriedades como convexidade e diferenciabilidade. Dada a nossa definição de risco-ℓ condicional:

$$L(\eta, v; \ell) = \eta\ell_+(v) + (1-\eta)\ell_-(v) \tag{4.13}$$

onde $\eta \in [0, 1]$, $v \in \mathbb{R}$, temos o risco-ℓ condicional ótimo[2] dado por $L^*(\eta, v; \ell) = \inf_{v\in\mathbb{R}} L(\eta, v; \ell)$. Se $L^-_\alpha(\eta, v; \ell) = \inf_{v\in\mathbb{R}|v(\eta-\alpha)\leq 0} L(\eta, v; \ell)$ é o risco-ℓ condicional ótimo quando o classificador comete erros, de acordo com Scott [2012], a função de perda ℓ é calibrada para classificação ponderada (α-CC, $\alpha \in (0, 1)$) se e somente se $L^-_\alpha(\eta, v; \ell) > L^*(\eta, v; \ell)$ com $\eta \in [0, 1]$, $\eta \neq \frac{1}{2}$ e $\eta \neq \alpha$.

Para a função AWT, temos:

$$\ell_{AWT}(v; p, q) = (\frac{1}{p}e^{pv}, \frac{1}{q}e^{-qv})$$

então o risco condicional, dada a Equação 4.13, é:

[2]Note que η denota tanto o escalar $\eta \in [0, 1]$ quanto a função $\eta(x) = P(Y = 1|X = x)$.

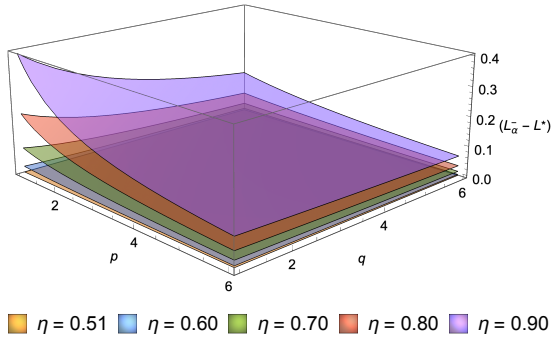

Figura 4.1: $L_\alpha^- - L^*$ para diversos valores de p, q e $\eta = \{0.51, 0.6, 0.7, 0.8, 0.9\}$

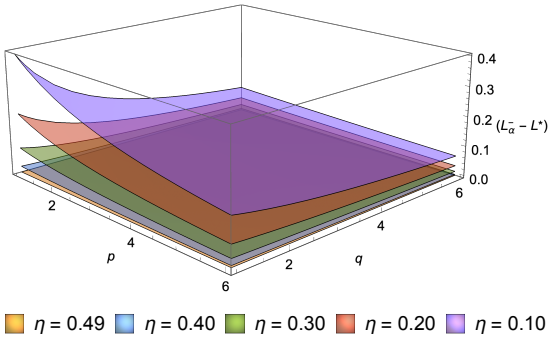

Figura 4.2: $L_\alpha^- - L^*$ para diversos valores de p, q e $\eta = \{0.49, 0.4, 0.3, 0.2, 0.1\}$

$$L(\eta, v; \ell_{AWT}) = \left(\frac{\eta}{q} e^{-qv}\right) + \left(\frac{1-\eta}{p} e^{pv}\right)$$

que é minimizado por:

$$v^* = \arg\min_v L(\eta, v; \ell_{AWT}) = \frac{1}{p+q} \log\left(\frac{\eta}{1-\eta}\right)$$

desde que:

$$\frac{\partial}{\partial v} L(\eta, v; \ell_{AWT}) = -\eta e^{-qv} + (1-\eta) e^{pv}$$

Assim, temos:

$$\begin{aligned} L^*(\eta, v; \ell_{AWT}) &= L(\eta, v^*; \ell_{AWT}) \\ &= \frac{\eta}{q}\left(\frac{\eta}{1-\eta}\right)^{-\frac{q}{p+q}} + \frac{1-\eta}{p}\left(\frac{\eta}{1-\eta}\right)^{\frac{p}{p+q}} \\ L_\alpha^-(\eta, v; \ell_{AWT}) &= \frac{\eta}{q} + \frac{1-\eta}{p} \end{aligned}$$

Logo, $L_\alpha^-(\eta, v; \ell_{AWT}) > L^*(\eta, v; \ell_{AWT})$, já que $\eta \in (0, 1)$, $\eta \neq \frac{1}{2}$ e $p, q \geq 1$, como queríamos provar. A título de ilustração, nas Figuras 4.1 e 4.2, podemos ver como a diferença $L_\alpha^-(\eta, v; \ell_{AWT}) - L^*(\eta, v; \ell_{AWT})$ é sempre positiva, considerando diferentes valores de η, p e q. □

Seja $\mathcal{F}$ o conjunto de funções $f : \mathcal{X} \to \mathbb{R}$. Cada $f \in \mathcal{F}$ representa um classificador $x \to sinal(f(x; \Theta))$ com $sinal(0) = -1$, parametrizado por um conjunto de hiperparâmetros Θ que produz um vetor de estimativas para as classes positivas e negativas. Seja $\Omega = \langle \frac{1}{p}, \frac{1}{q} \rangle$ um vetor de pesos. Assumindo que f é uma DFN com pesos e vieses

Θ, o risco empírico $R_{\ell_{AWT}}$ associado à rede neural usando ℓ_{AWT} é dado pela Equação 4.14:

$$R_{\ell_{AWT}}(\Theta, \Omega) = \frac{1}{N}\sum_{j=1}^{N} \begin{cases} \frac{1}{q}\ \exp(-f(x_j;\Theta)\ q), & y_j \geq 0 \\ \frac{1}{p}\ \exp(f(x_j;\Theta)\ p), & y_j < 0 \end{cases} \tag{4.14}$$

onde N é o número de exemplos de treino.

4.4 Função assimétrica de perda AWF

Em nossa segunda função de perda, propomos uma abordagem de ponderação assimétrica com quatro pesos, chamada AWF[3], para lidar com o problema de desbalanceamento de classes. Nesta abordagem, consideramos que tanto erros quanto acertos de classificação podem ser ponderados de maneiras diferentes, como observado na Equação 4.15:

$$\begin{aligned} \ell_{AWF}(v) =& (\frac{\mathbb{1}_{\{v\leq 0\}}}{p'} exp(v \cdot p') + \frac{\mathbb{1}_{\{v>0\}}}{p} exp(v \cdot p), \\ & \frac{\mathbb{1}_{\{v\leq 0\}}}{q} exp(-v \cdot q) + \frac{\mathbb{1}_{\{v>0\}}}{q'} exp(-v \cdot q')) \end{aligned} \tag{4.15}$$

onde v corresponde à estimativa e $\Omega = \begin{pmatrix} 1/p' & 1/q \\ 1/p & 1/q' \end{pmatrix}$ corresponde aos pesos atribuídos a casos de classificação correta e incorreta. Neste caso, o risco empírico $R_{\ell_{AWF}}$ pode ser expresso por:

$$R_{\ell_{AWF}}(\Theta, \Omega) = \frac{1}{N}\sum_{j=1}^{N} \begin{cases} \frac{1}{p'}\ \exp(f(x_j;\Theta)\ p'), & f(x_j;\Theta) \leq 0, y_j \leq 0 \\ \frac{1}{p}\ \exp(f(x_j;\Theta)\ p), & f(x_j;\Theta) > 0, y_j \leq 0 \\ \frac{1}{q}\ \exp(-f(x_j;\Theta)\ q), & f(x_j;\Theta) \leq 0, y_j > 0 \\ \frac{1}{q'}\ \exp(-f(x_j;\Theta)\ q'), & f(x_j;\Theta) > 0, y_j > 0 \end{cases} \tag{4.16}$$

Note que quando $\Omega = \mathbf{1}$ tanto a Equação 4.9 quanto a Equação 4.15 são reduzidas para a função de perda exponencial usada por *AdaBoost*. Por outro lado, quando

[3]O nome da função tem origem no termo em inglês *Asymmetric Weighted approach with Four weights (AWF)*

$\Omega = \begin{pmatrix} 1/p & 1/q \\ 1/p & 1/q \end{pmatrix}$, AWF (Equação 4.15) é reduzida para AWT (Equação. 4.9).

Como a AWT, a AWF é uma função própria (estrita) composta que satisfaz as propriedades de aversão a palpite e calibração para classificação sensível a custo, descritas nos Lemas 4.4.1, 4.4.2 e 4.4.3.

Lema 4.4.1. *Para quaisquer $p, q, p', q' > 0$, seja ℓ a função de perda AWF, dada na Equação 4.15. Então, $\ell \in \mathcal{L}_{PEC}$, ou seja, ela é própria (estrita) composta.*

Demonstração. Para o caso $v \leq 0$, $\ell = (\frac{1}{p'}e^{v\ p'}, \frac{1}{q}e^{-v\ q})$. Dado que ℓ é diferenciável, podemos obter função inversível $\psi^{-1} : [0, 1] \to \mathbb{R} \cup \{-\infty, +\infty\}$, como feito antes na prova do Lema 4.3.1, observando que:

$$(\forall v \in \mathbb{R})\psi^{-1}(v) = \frac{1}{1 - \frac{\ell'_+(v)}{\ell'_-(v)}} = \frac{1}{1 + e^{-(p'+q)v}}$$

Como $\psi^{-1}(v)$ é inversível, a função obtida no caso 1 é própria (estrita) composta. De forma similar, para o caso $v > 0$, $\ell = (\frac{1}{p}e^{v\ p}, \frac{1}{q'}e^{-v\ q'})$. Dado que ℓ é diferenciável, obtemos:

$$(\forall v \in \mathbb{R})\psi^{-1}(v) = \frac{1}{1 + e^{-(p+q')v}}$$

Que também é inversível. Logo, a função obtida no caso 2 também é própria (estrita) composta.

□

Lema 4.4.2. *A função de perda AWF é aversa a palpites. Mais precisamente, para $p, q, p', q' > 1$, dadas estimativas corretas v, a função $\ell_{AWF}(v)$ é aversa a palpite se e somente se $\ell_{AWF}(v) < \ell_{AWF}(c)$ onde c é conjunto de todos os pontos que representam palpites.*

Demonstração. Dadas as mesmas condições da prova do Lema 4.10, com palpites mapeados para 0, n_+ instâncias positivas e n_- instâncias negativas classificadas corretamente, temos:

$$\frac{n_+}{q'}e^{-vq'} + \frac{n_-}{p'}e^{vp'} < \frac{n_+}{q'} + \frac{n_-}{p'} \tag{4.17}$$

o que é verdade desde que (i) $e^{-vq'} < 1$ para as n_+ instâncias corretas pois, neste caso, $v > 0$ e $q' \geq 1$ e (ii) $e^{vp'} < 1$ para as n_- instâncias corretas pois, neste caso, $v < 0$ e $p' \geq 1$.

□

Lema 4.4.3. *A função de perda AWF é calibrada para classificação sensível a custo.*

Demonstração. Como observado por Khan et al. [2018], uma forma de determinar se uma função de perda é calibrada para classificação é verificar se existe uma relação inversa entre o risco condicional associado a esta função com a saída ótima do estimador. Considere os pesos $\Omega = \begin{pmatrix} \Omega_{--} & \Omega_{-+} \\ \Omega_{+-} & \Omega_{++} \end{pmatrix} = \begin{pmatrix} 1/p' & 1/q \\ 1/p & 1/q' \end{pmatrix}$ e suponha uma rede neural com um único neurônio na última camada com saída denotada por o ($o \in [-1, 1]$). A AWF pode ser escrita como:

$$\begin{aligned} \ell(o;\Omega) =& (\mathbb{1}_{\{o>0\}}\Omega_{+-}e^{-\frac{o}{\Omega_{+-}}} + \mathbb{1}_{\{o\leq 0\}}\Omega_{--}e^{-\frac{o}{\Omega_{--}}}, \\ & \mathbb{1}_{\{o>0\}}\Omega_{++}e^{-\frac{o}{\Omega_{++}}} + \mathbb{1}_{\{o\leq 0\}}\Omega_{-+}e^{-\frac{o}{\Omega_{-+}}}) \end{aligned} \tag{4.18}$$

Dada esta função, o risco condicional de classificação pode ser expresso em termos do valor esperado para uma instância x como:

$$\begin{aligned} L(\Omega, o; \ell) =& \mathbb{E}(\ell(o;\Omega)) \\ =& \sum_{\pi\in\{+1,-1\}} P(\pi|x)\ell(o;\Omega) \\ =& P(\pi = +1|x)(\Omega_{++}e^{-\frac{o}{\Omega_{++}}} + \Omega_{-+}e^{-\frac{o}{\Omega_{-+}}}) + \\ & P(\pi = -1|x)(\Omega_{+-}e^{\frac{o}{\Omega_{+-}}} + \Omega_{--}e^{\frac{o}{\Omega_{--}}}) \end{aligned} \tag{4.19}$$

Para achar o conjunto ideal de saídas o da rede neural, basta minimizarmos o risco. Considere $s_\pi = sinal(\pi)$ e $s_o = sinal(o)$.

$$\frac{\partial}{\partial o}L(\Omega, \pi; \ell) = \left.\frac{\partial L(\Omega, \pi; \ell)}{\partial o}\right|_{s_o \neq s_\pi} + \left.\frac{\partial L(\Omega, \pi; \ell)}{\partial o}\right|_{s_o = s_\pi} \tag{4.20}$$

Resolvendo os termos da Equação 4.20, obtemos:

$$\left.\frac{\partial L(\Omega, \pi; \ell)}{\partial o}\right|_{s_o = s_\pi} = \frac{\partial}{\partial o}(P(\pi|x)\Omega_{tt}e^{-t\frac{o}{\Omega_{tt}}}) = -tP(\pi|x)e^{-t\frac{o}{\Omega_{tt}}} \tag{4.21}$$

$$\left.\frac{\partial L(\Omega,\pi;\ell)}{\partial o}\right|_{s_o \neq s_\pi} = \frac{\partial}{\partial o}((1-P(\pi|x))\Omega_{-tt}e^{t\frac{o}{\Omega_{-tt}}}) = t(1-P(\pi|x))e^{t\frac{o}{\Omega_{-tt}}} \tag{4.22}$$

Somando as Equações 4.21 e 4.22 e igualando o resultado a zero, obtemos uma função para o. Assim, de:

$$\left.\frac{\partial L(\Omega,\pi;\ell)}{\partial o}\right|_{s_o \neq s_\pi} + \left.\frac{\partial L(\Omega,\pi;\ell)}{\partial o}\right|_{s_o = s_\pi} = 0$$

obtemos:

$$o = \frac{\Omega_{tt}\Omega_{-tt}}{t(\Omega_{tt}-\Omega_{-tt})}(1+\log tP(\pi|x) - \log t(1-P(\pi|x))) \tag{4.23}$$

ou seja, existe uma relação inversa entre o risco condicional e a saída ótima do previsor, como queríamos provar. □

Para treinar uma rede neural com parâmetros Θ usando as funções AWT e AWF é necessário definir empiricamente os valores ótimos para os hiper-parâmetros Ω. Isto é oneroso uma vez que diferentes bases de dados podem implicar em diferentes valores ótimos. Seria mais conveniente que o modelo aprendesse tanto Θ quanto Ω durante o processo de treinamento, o que é proposto na seção a seguir.

4.5 Algoritmo de treinamento

Dadas as definições anteriores, agora vamos minimizar o risco de classificação empírico R_ℓ encontrando parâmetros Θ e Ω, que minimizem uma função de perda ℓ. Para isso, usamos um algoritmo de otimização alternada baseada em gradientes, similar ao proposto por Khan et al. [2018]. A ideia é que a otimização dos parâmetros Θ e Ω ocorra de acordo com a Equação 4.24:

$$F(\Theta,\Omega) = \frac{1}{2} \parallel T-\Omega \parallel_2^2 + R_\ell^{val}(\Theta,\Omega) \tag{4.24}$$

onde R_ℓ^{val} é o risco estimado no conjunto de validação e o tensor T representa as penalidades e recompensas desejadas a serem aplicadas às estimativas. Portanto T pode ser considerado como o alvo para Ω na iteração de aprendizado atual. Mais especificamente, a diagonal principal de T corresponde às recompensas aplicadas aos verdadeiros positivos e verdadeiros negativos, enquanto sua diagonal secundária corresponde às penalidades aplicadas a falsos positivos e falsos negativos.

Para induzir Ω a reduzir os erros e recompensar as estimativas corretas, levamos em consideração a taxa de desbalanceamento do conjunto de treinamento e os erros observados em um conjunto de validação. A intuição, de forma geral, é que devemos dar importância maior para instâncias da classe menos frequente de forma proporcional à taxa de desbalanceamento. Contudo, isto deve ser feito de forma a não aumentar o erro de classificação das instâncias. Como consequência, T pode ser definido como:

$$T = H \circ \exp\big(-\frac{1}{2}\big(\frac{E-\mu_E}{\sigma_E}\big)^2\big) \tag{4.25}$$

onde $\circ$ representa o produto de Hadamard, H é o histograma de classes que captura o desbalanceamento de dados e é definido como:

$$H = \begin{pmatrix} \frac{I}{I+K} & \frac{K}{I+K} \\ \frac{I}{I+K} & \frac{K}{I+K} \end{pmatrix} \tag{4.26}$$

onde I corresponde ao número de instâncias positivas e K, o número de instâncias negativas. Finalmente, E é a matriz de erro normalizada, definida como:

$$E = \begin{pmatrix} 0 & \frac{FN}{FN+FP} \\ \frac{FP}{FN+FP} & 0 \end{pmatrix} \tag{4.27}$$

Sem considerar o erro, as recompensas indicadas por T são proporcionais ao número de exemplos de cada classe, $\frac{I}{I+K}$ para instâncias positivas e $\frac{K}{I+K}$ para instâncias negativas. O erro é capturado pelo componente exponencial $\exp(-\frac{1}{2}\big(\frac{E-\mu_E}{\sigma_E}\big)^2)$. O expoente corresponde ao erro centrado na média, escalado por dois desvios padrões. Os parâmetros μ_E e σ_E podem ser estimados a partir do conjunto de treinamento (ou por uma amostra dele) usando validação cruzada. Quanto maior é o erro, maior é a penalidade aplicada à diagonal secundária de T com penalidades proporcionais ao tamanho relativo da classe incorretamente estimada.

Para aprender os melhores valores para os parâmetros Θ e Ω, definimos uma abordagem baseada em épocas de treino, conforme apresentado no Algoritmo 1. Especificamente, para otimizar Ω, usamos o algoritmo de gradiente descendente para calcular a direção do passo para atualizar os parâmetros (linhas 7-11). Em seguida, para otimizar Θ, usamos o gradiente descendente estocástico com a retropropagação de erro (linhas 12-15). Esses passos seguem alternadamente até que um número máximo de épocas seja atingido (linha 6)[4]. O tamanho dos passos (taxas de aprendizado) para Ω e Θ são dados por γ_Ω e γ_Θ. O tamanho de γ_Ω reduz sempre que erro de validação

[4]Alternativamente, pode-se usar um critério de parada antecipada, onde o algoritmo interrompe a iteração quando ele não consegue melhorar o resultado em uma certa janela de tentativas.

diminui. Note que o algoritmo inicia sem nenhuma preferência particular por penalidades e recompensas, ou seja, Ω é iniciado com $\mathbf{1}$. Por sua vez, como usual, Θ inicia com valores aleatórios.

Algorithm 1 Otimização iterativa de parâmetros (Θ, Ω)

Entrada:
- Conjunto de treino: $\mathcal{D} = X \times Y = \{x_j, y_j\}_{j=1..N}$ com I instâncias positivas e K negativas
- Conjunto de validação: $\mathcal{D}^{(V)} = X^{(V)} \times Y^{(V)}$
- Taxas de aprendizagem: $\gamma_\Theta, \gamma_\Omega$
- Número de épocas: $\mathcal{E}_{max}$
- Número de *batches*: B

Saída: parâmetros aprendidos (Θ^*, Ω^*)

procedure
  NN $\leftarrow$ NeuralNet()
  $\Theta \leftarrow$ init(NN) ▷ Inicialização aleatória
  $\Omega \leftarrow \mathbf{1}$
  $\xi_{min} \leftarrow \infty$ ▷ Erro mínimo
  for $e = 1$ **to** $\mathcal{E}_{max}$ **do**
    $T = H \circ \exp(-\frac{(E-\mu_E)^2}{2\sigma_E^2})$
    $F(\Theta, \Omega) = \frac{1}{2} \parallel T - \Omega \parallel_2^2 + R^{val}_{\ell_{PC}}(\Theta, \Omega)$
    $\nabla_\Omega \leftarrow$ compute-grad(X, Y, $F(\Theta, \Omega)$) $\approx -(vec(T) - vec(\Omega))\mathbf{1}^T$
    $\Omega^* \leftarrow$ update-CostParams(Ω, γ_Ω , ∇_Ω)
    if $\Omega > \Omega^*$ **then**
      $\Omega \leftarrow \Omega^*$
    for $b = 1$ **to** B **do**
      $\Theta^* \leftarrow$ fit(NN, X_b,Y_b, Θ, Ω)
      $\Theta \leftarrow \Theta^*$
    end for
    $\xi^{(V)} \leftarrow$ predict(NN, $X^{(V)}$, $Y^{(V)}$, Θ) ▷ Erros de validação
    if $\xi_{min} > \xi^{(V)}$ **then**
      $\xi_{min} \leftarrow \xi^{(V)}$
      $\gamma_\Theta \leftarrow \gamma_\Theta \times 0.01$ ▷ Diminuir o tamanho do passo
    end if
  end for
  return (Θ^*, Ω^*)

4.6 Considerações Finais

Neste capítulo apresentamos o conjunto das funções próprias (estritas) compostas, como descritas por Menon & Williamson [2016]. Propusemos variantes de uma função deste conjunto, a função assimétrica p-Classification, para o problema de *credit scoring*.

Para cada uma das funções propostas, AWT e AWF, mostramos que elas são próprias compostas, calibradas para classificação e aversas a palpites. Finalmente, propusemos um algoritmo alternado de treino para aprender os hiper-parâmetros das funções junto com os demais hiper-parâmetros do modelo profundo.

Capítulo 5

Experimentos e resultados

Neste trabalho, estamos interessados em avaliar o impacto das estratégias de amostragem em modelos profundos de pontuação de crédito e introduzir uma nova abordagem sensível ao custo para lidar com o problema do desbalanceamento de classes. Para isso, neste capítulo, apresentamos o protocolo experimental que seguimos, as coleções de dados utilizadas e os resultados dos experimentos realizados.

5.1 Protocolo experimental

Nesta seção, descrevemos o planejamento e objetivos de cada experimento realizado. Para tanto, organizamos os experimentos em cinco cenários:

1. Avaliação de modelos com distribuição real de classes. Neste cenário, avaliamos o desempenho de modelos rasos e profundos que não usam estratégia de pré-processamento para balanceamento do conjunto de dados de treinamento. O melhor modelo deste cenário foi adotado como *baseline* para os próximos experimentos.

2. Avaliação de modelos com estratégias de amostragem. Neste cenário, nosso objetivo foi avaliar o impacto de modelos que lidam com o desbalanceamento de classes a partir de estratégias balanceamento baseadas em dados. Para isso, cada modelo raso citado na seção anterior foi combinado às quatro estratégias de amostragem encontradas na literatura. Além desses, modelos profundos projetados para resolver *credit scoring* usando alguma das estratégias de amostragem também foram avaliados. O melhor modelo deste cenário foi usado como *baseline* do quarto cenário, onde são avaliados métodos que usam abordagens baseadas em algoritmos.

3. Obtenção manual do melhor valor para o hiperparâmetro p para a função de perda p-Classification. Essa função é a base para a função AWT, avaliada no próximo cenário de experimentos. Para esse experimento, utilizamos uma abordagem heurística para a seleção do melhor p.

4. Avaliação de modelos com aprendizagem sensível a custo. Neste cenário, nosso objetivo é avaliar o desempenho de modelos que lidam com desbalanceamento de classes a partir de abordagens assimétricas baseadas em algoritmos. Para isso, o desempenho de funções assimétricas foi avaliado a partir da arquitetura profunda de uma DFN. O desempenho dessas abordagens foi comparado com o desempenho dos nossos dois modelos propostos e com o desempenho do melhor modelo do segundo cenário.

5. Avaliação de tempo de treinamento. Neste cenário, avaliamos o custo de treinamento dos modelos que foram avaliados no cenário quatro. Para isso, avaliamos o número médio de épocas necessárias para conclusão do treino e o tempo médio por época de treino. Os experimentos foram realizados na mesma máquina e sob as mesmas condições de concorrência com outros processos.

Na seção a seguir, apresentamos as duas coleções de dados utilizadas em nossa pesquisa.

5.2 Coleções

Neste trabalho, utilizamos duas bases de dados, uma coletada de um grande varejista brasileiro e outra coleta de um site de competições públicas, conforme descrito nas seções a seguir.

5.2.1 Coleção de transações do VAREJO

Nossa primeira base de dados possui 665.428 instâncias que representam clientes que compraram um ou mais produtos a crédito em uma grande rede de varejo brasileira, entre dezembro de 2016 e maio de 2018. A fim de garantir a confidencialidade das informações sobre transações e clientes, estes dados estão anonimizados. Como conjunto típicos de dados em problemas de pontuação de crédito, esta coleção apresenta alto grau de desbalanceamento da classe alvo, onde 90,55% dos clientes são rotulados como bons pagadores contra apenas 9,45% maus pagadores. Cada cliente é descrito por 83 atributos, onde 75 são numéricos e 8 são categóricos. O varejista adota um

período de 12 meses para observação, 6 meses para o desempenho e um cliente torna-se inadimplente após 60 dias de atraso. Como estratégia de pré-processamento, todos os atributos categóricos foram convertidos em atributos binários numéricos usando a abordagem de um atributo por valor [Breiman, 2017], em que um atributo categórico com k valores distintos é transformado em k atributos binários. Após a transformação, o banco de dados passou a ser composto por 116 atributos. Os atributos numéricos foram padronizados para apresentar média zero e desvio padrão unitário.

5.2.2 Coleção de competição do KAGGLE

O segundo conjunto de dados usado é disponibilizado publicamente pelo Kaggle, originalmente usado em uma das suas competições[1] criada para melhorar o estado da arte em *credit scoring*. Nesta competição, o objetivo era prever a probabilidade de que um cliente de um banco tivesse problemas financeiros em uma janela de dois anos. Este conjunto de dados contém 150.000 instâncias de candidatos, com 139.974 bons e 10.026 pagadores ruins. Cada candidato é representado por 11 atributos numéricos iniciais. Como estratégia de pré-processamento, colunas foram excluídas quando seus valores eram nulos para mais que 50% das instâncias. Colunas com valores nulos para menos que 50% das instâncias foram preenchidas com o valor da mediana. Além disso, criamos cinco novas colunas numéricas derivadas das iniciais e, por fim, normalizamos toda a coleção de dados. Após as transformações, o conjunto de dados passou a contar com 146.067 instâncias de clientes, com 136.222 bons e 9.845 maus pagadores. Cada candidato passou a ser representado por 16 atributos numéricos.

5.3 Arquitetura Profunda

Neste trabalho, usamos uma DFN para aprender a representação dos atributos dos conjuntos de dados desbalanceados e de alta dimensão para tarefas de classificação. Escolhemos essa arquitetura em função de sua simplicidade. A arquitetura profunda usada neste trabalho é semelhante à DFN típica usada em problemas de classificação com desbalanceamento de dados [Chung et al., 2015; Wang et al., 2016] e em problemas de pontuação de crédito [Addo et al., 2018; Ha & Nguyen, 2016], exceto pelo algoritmo de aprendizado e pelas funções de perda usadas. Nossa DFN foi treinada usando a abordagem de aprendizado alternado, mostrada no Algoritmo 1, utilizando nossas funções de perda propostas, AWT (Eq. 4.9) e AWF (Eq. 4.15).

[1] Give me some credit - `https://www.kaggle.com/c/GiveMeSomeCredit/`

Embora muitas arquiteturas pudessem ser usadas, não é o foco deste trabalho estudar diferentes configurações de modelos profundos. Assim, optamos por uma DFN especificada como segue. A camada de entrada possui U neurônios, onde U é o número de atributos usados para representar as instâncias da coleção. Usamos cinco camadas ocultas, sendo quatro com 500 neurônios, enquanto a última camada oculta têm cem neurônios. As camadas ocultas usam a função de ativação ELU enquanto a camada de saída usa uma SOFTMAX para classificação. Como estratégia de regularização, aplicamos *batch normalization* e *dropout* de 60% em cada camada, exceto na camada de saída. O treino é configurado para iterar por, no máximo, mil épocas usando a abordagem de parada antecipada para evitar *overfiting*. A parada ocorre quando o otimizador não consegue melhorar o resultado obtido em uma janela de dez iterações. Como otimizador usamos o algoritmo adaptativo com momento *adam* com *batches* de tamanho 2048.

5.4 Metodologia de Avaliação

Os experimentos realizados neste trabalho seguiram a estratégia de validação cruzada de dez partições, comumente utilizada na literatura, onde cada partição é composta por amostras aleatórias de dados de clientes da nossa base de dados de treinamento. Os conjuntos de dados em cada rodada da validação cruzada foram divididos em amostras de treinamento (50 %), validação (20 %) e teste (30 %). Os resultados foram mensurados usando as medidas AUC, KS e a taxa de Verdadeiro Positivo (VP), descritas na Seção 2.1.2. Para cada abordagem, mostramos o ganho da métrica AUC em relação ao *baseline*. Além disso, marcamos com "*" resultados que diferem do *baseline* com pelo menos 95% de confiança, usando o teste não paramétrico de Wilcoxon [Wilcoxon, 1992].

Todos os experimentos foram realizados em um computador dedicado, com sistema operacional Linux 18.04, 12 CPUs Intel(R) Xeon(R) E5-2690 v3 2.60GHz, 116 GB de RAM, 1 TB de disco rígido e GPU NVIDIA Tesla K80 para computação vetorial. Todos os arquivos necessários estavam no disco rígido, sem necessidade de download durante o processamento. Para tomadas de tempo, todos os dados necessários foram carregados para a memória RAM e esse tempo de carregamento não foi adicionado ao tempo total de treinamento. Os modelos foram feitos em Python, versão 3.6, e Tensorflow, versão 1.15.

5.5 Resultados dos cenários de experimentos

Nesta seção, apresentamos os detalhes de cada um dos cinco cenários de experimentos implementados em nosso trabalho.

5.5.1 Avaliação de modelos com distribuição real de classes

Neste primeiro cenário de experimentos, avaliamos os modelos rasos clássicos Gradient Boosting (GB), Random Forest (RF), Regressão logística (RLo) e Regressão linear (RLi). Esses algoritmos foram escolhidos porque representam os primeiros (RLo e RLi) e os mais recentes (GB e RF) métodos rasos usados na avaliação de risco de crédito. Além disso, escolhemos métodos profundos [Tran et al., 2016; Neagoe et al., 2018; Luo et al., 2017; Yifei, 2016] projetados para resolver diretamente problemas de *credit scoring* sem tratamento de desbalanceamento. A regressão linear foi escolhida como *baseline* deste cenário, por ter sido o primeiro método usado para lidar com pontuação de crédito automática.

A Tabela 5.1 apresenta o resultados obtidos com a coleção de dados de varejo, onde as colunas da tabela indicam o método avaliado, os valores alcançados para AUC, KS e VP, além do ganho em relação ao *baseline*. Nessa tabela podemos verificar que a CNN proposta por Yifei [2016] apresentou os melhores resultados em todas as métricas avaliadas, alcançando $AUC = 0.813, KS = 0.504, VP = 0.81$ e $Ganho = 16.62\%$. Em seguida, verificamos que o GB alcançou os melhores resultados entre os métodos rasos, atingindo $AUC = 0.810, KS = 0.465, VP = 0.05$ e $Ganho = 16.06\%$. Apesar dos resultados parecidos em AUC, os dois modelos são diferentes em KS e VP. Avaliando especificamente a VP, notamos que todos os modelos rasos classificam corretamente menos de 10% das classes positivas, onde os clientes são maus pagadores, indicando o viés dos modelos de classificação em favor da classe majoritária. Esse desempenho ruim pode ter ocorrido devido à alta taxa de desbalanceamento de dados. Além disso, a abordagem Neagoe et al. [2018] não obteve resultados significativamente diferentes daqueles alcançados pelo *baseline*.

A Tabela 5.2 apresenta os resultados alcançados na base de dados do KAGGLE, onde o método de Yifei [2016] e o método de Luo et al. [2017] atingiram o mesmo resultado em AUC, mas o segundo método teve melhor desempenho em KS, se tornando, assim, o melhor resultado desse *dataset*. Nesse caso, o método de Luo et al. [2017] alcançou $AUC = 0.848, KS = 0.58, VP = 0.62$ e $Ganho = 3.77$. Assim como ocorreu na base de dados de VAREJO, os modelos rasos falharam na tarefa de classificar a classe minoritária, atingindo $VP <= 15\%$. Novamente, o GB obteve o melhor resultado entre

Tabela 5.1: Desempenho de modelos sem abordagens de amostragem - VAREJO

Método	AUC	KS	VP	Ganho(%)
Regressão linear	0.698	0.362	0.06	-
Regressão logística	0.795*	0.446*	0.09	13.98*
Gradient Boosting	0.800*	0.465*	0.05	14.62*
Random Forest	0.799*	0.478*	0.02	14.49*
Luo et al. [2017]	0.801*	0.491*	0.72*	14.88*
Tran et al. [2016]	0.743*	0.371	0.76*	6.47*
Neagoe et al. [2018]	0.691	0.290*	0.68*	-0.92
Yifei [2016]	**0.813***	**0.504***	**0.81***	**16.62***

os métodos rasos.

Tabela 5.2: Desempenho de modelos sem abordagens de amostragem - KAGGLE

Método	AUC	KS	VP	Ganho(%)
Regressão Linear	0.817	0.502	0.03	-
Regressão Logística	0.833*	0.513*	0.15	1.97*
Gradient Boosting	0.839*	0.539*	0.13	2.73*
Random Forest	0.838*	0.511*	0.03	2.58*
Luo et al. [2017]	**0.848***	**0.580***	**0.62***	**3.77***
Tran et al. [2016]	0.787*	0.401*	0.62*	-3.67*
Neagoe et al. [2018]	0.742*	0.351*	0.61*	-9.12*
Yifei [2016]	0.848*	0.557*	0.73*	3.77*

Como resultado desse cenário de experimentos, usaremos o método de Yifei [2016] como *baseline* para a base de dados de VAREJO e o método de Luo et al. [2017] como *baseline* para a coleção do KAGGLE no próximo cenário de experimentos.

5.5.2 Avaliação de modelos com estratégias para lidar com desbalanceamento

Neste cenário de experimentos, apresentaremos os resultados dos modelos de *credit scoring* que usam abordagens de amostragem como etapa de pré-processamento para balancear o conjunto de treinamento. Entre os métodos de amostragem, avaliamos os métodos presentes na literatura de *credit scoring*: *oversampling* (Aleatório, ADASYN e SMOTE) e *undersampling* aleatório. Nesse caso, cada modelo raso foi combinado

a cada uma das abordagens de amostragem. Para modelos profundos, usamos apenas abordagens de amostragem definidas em seus trabalhos originais. Na Tabela 5.3 apresentamos os resultados dos experimentos, em que o primeiro método é o *baseline* de Yifei [2016], a coluna *Pré-processamento* indica os tipos de amostragem utilizados em cada método, onde *Oversampling* indica *oversampling aleatório* e *Undersampling* indica *undersampling aleatório*.

Tabela 5.3: Desempenho de modelos que usam abordagens de amostragem - VAREJO

Método	Pré-processamento	AUC	KS	VP	Ganho(%)
Yifei [2016]	**-**	**0.813**	**0.504**	**0.81**	**-**
Regressão linear	ADASYN	0.665*	0.321*	0.72*	-18.29*
	SMOTE	0.697*	0.359*	0.78*	-14.35*
	Oversampling	0.699*	0.361*	0.80*	-14.11*
	Undersampling	0.697*	0.359*	0.81	-14.29*
Regressão logística	ADASYN	0.793*	0.451*	0.72*	-2.47*
	SMOTE	0.803*	0.462*	0.75*	-1.25*
	Oversampling	0.804*	0.465*	0.77*	-1.14*
	Undersampling	0.792*	0.440*	0.78*	-2.66*
Gradient Boosting	ADASYN	0.761*	0.388*	0.73*	-6.58*
	SMOTE	0.763*	0.393*	0.64*	-6.17*
	Oversampling	0.810	0.467*	0.78*	-0.39
	Undersampling	0.810	0.466*	0.79*	-0.45
Random Forest	ADASYN	0.741*	0.365*	0.68*	-8.94*
	SMOTE	0.760*	0.375*	0.66*	-6.61*
	Oversampling	0.789*	0.435*	0.79*	-3.04*
	Undersampling	0.786*	0.433*	0.81	-3.34*
Ha & Nguyen [2016]	New FS	0.751*	0.384*	0.78*	-7.63*
Yu et al. [2018]	Oversampling	0.754*	0.372*	0.77*	-7.28*
Kvamme et al. [2018]	Oversampling	0.686*	0.371*	0.71*	-15.70*
Zhu et al. [2018]	Relief	0.676*	0.284*	0.69*	-16.94*
Addo et al. [2018]	SMOTE	0.748*	0.375*	0.71*	-8.09*

Como resultado inicial da Tabela 5.3, observamos que nenhum dos métodos superou o *baseline*, o que indica que a abordagem profunda proposta por Yifei [2016] tem desempenho superior ou similar ao desempenho de métodos rasos e profundos que utilizam abordagens de amostragem. Também observamos que GB apresentou novamente

os melhores resultados entre os algoritmos rasos, ocorrendo empate no desempenho dos modelos de GB que utilizaram *oversampling* e *undersampling*. Este foi o único método a apresentar um desempenho estatisticamente similar ao baseline (em AUC). Em nossos estudos, técnicas de amostragem aleatória superaram o desempenho de técnicas sintéticas, sendo que o *oversampling* Aleatório apresentou o melhor resultado entre elas. Além disso, ao analisarmos os valores apresentados na coluna VP, verificamos que o balanceamento em nível de dados melhorou o desempenho dos modelos rasos na tarefa de classificação da classe minoritária.

A Tabela 5.4 apresenta os resultados dos experimento com balanceamento de dados da coleção de competição KAGGLE, onde pudemos notar resultados similares ao resultados alcançados na coleção de VAREJO. Nenhum método que usou balanceamento de dados obteve desempenho melhor do que o alcançado pelo *basline* de Luo et al. [2017]. Além disso, a tabela nos permite verificar que, novamente, GB com *oversampling* alcançou o melhor resultado entre os modelo rasos.

Ao analisarmos o desempenho de modelos rasos em relação ao uso ou não de abordagens de amostragem, verificamos que, conforme apresentado na Figura 5.1, os modelos alcançam, em média, valores equivalentes de AUC e KS. Contudo, ao analisarmos a taxa de Verdadeiro Positivo (VP), verificamos que modelos rasos que usam estratégias de amostragem conseguem classificar melhor a classe minoritária, do que modelos que lidam com o desbalanceamento real do problema de pontuação de crédito.

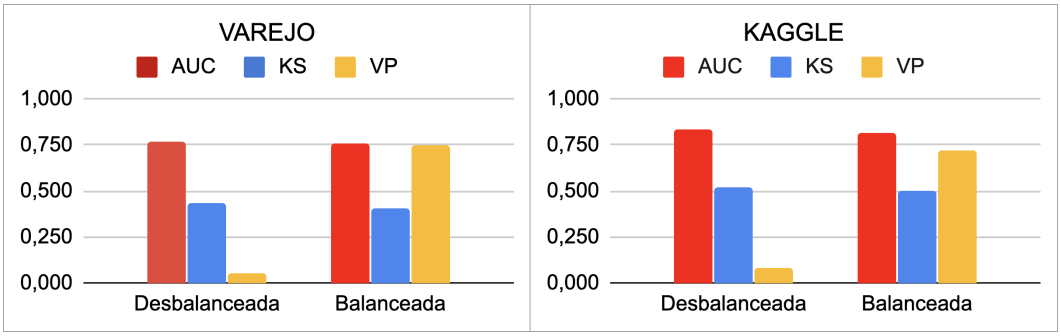

Figura 5.1: Desempenho de modelos rasos por distribuição de classes

A fim de avaliarmos o desempenho das abordagens de amostragem de dados utilizadas para o balanceamento de classes, agrupamos na Tabela 5.5 os valores médios de AUC, KS e VP alcançados por cada amostragem. Como resultado, podemos verificar que modelos que utilizam *Undersampling*, em média, alcançaram os melhores resultados para as métricas avaliadas. Além disso, verificamos que modelos que utilizam dados sintéticos gerados por ADASYN obtiveram os piores resultados desse cenário de experimentos.

Tabela 5.4: Desempenho de modelos que usam abordagens de amostragem - KAGGLE

Método	Pré-processamento	AUC	KS	VP	Ganho(%)
Luo et al. [2017]	**-**	**0.848**	**0.580**	**0.62**	-
Regressão linear	ADASYN	0.772*	0.449*	0.70*	-8.96*
	SMOTE	0.782*	0.464*	0.71*	-7.69*
	Oversampling	0.783*	0.466*	0.71*	-7.56*
	Undersampling	0.782*	0.450*	0.71*	-7.78*
Regressão logística	ADASYN	0.809*	0.476*	0.71*	-4.55*
	SMOTE	0.819*	0.489*	0.73*	-3.34*
	Oversampling	0.820*	0.489*	0.72*	-3.22*
	Undersampling	0.820*	0.503*	0.72*	-3.27*
Gradient Boosting	ADASYN	0.832*	0.527*	0.68*	-1.88*
	SMOTE	0.836*	0.533*	0.66*	-1.42*
	Oversampling	0.841	0.552*	0.77*	-0.76
	Undersampling	0.840	0.536*	0.77*	-0.90
Random Forest	ADASYN	0.820*	0.498*	0.69*	-3.28*
	SMOTE	0.829*	0.509*	0.76*	-2.24*
	Oversampling	0.838*	0.518*	0.76*	-1.14*
	Undersampling	0.828*	0.502*	0.77*	-2.34*
Ha & Nguyen [2016]	H2O score	0.793*	0.403*	0.67*	-6.42*
Yu et al. [2018]	Oversampling	0.797*	0.411*	0.66*	-5.99*
Kvamme et al. [2018]	Oversampling	0.736*	0.341*	0.64*	-13.14*
Zhu et al. [2018]	Relief	0.730*	0.336*	0.61	-13.58*
Addo et al. [2018]	SMOTE	0.770*	0.383*	0.65*	-9.12*

Tabela 5.5: Avaliação de desempenho por tipo de amostragem de dados

TIPO DE	VAREJO			KAGGLE		
AMOSTRAGEM	AUC	KS	VP	AUC	KS	VP
ADASYN	0,740	0,381	0,71	0,808	0,488	0,70
SMOTE	0,754	0,393	0,71	0,807	0,476	0,70
Oversampling	0,757	0,412	0,77	0,803	0,463	0,71
Undersampling	**0,771**	**0,425**	**0,80**	**0,818**	**0,498**	**0,74**

Ao analisarmos apenas os métodos profundos avaliados neste cenário de experimentos, verificamos que, em média, os métodos que lidam com o desbalanceamento real de classes alcançaram resultados superiores aos resultados dos métodos que utilizaram abordagens de amostragem de dados, em ambos os conjuntos de dados, como apresentado na tabela 5.6. Além disso, podemos verificar que as duas abordagens lidaram bem com a classificação da classe minoritária, alcançando em média $VP = 0,73$ na base VAREJO e $VP = 0,64$ na base KAGGLE, indicando que métodos profundos foram naturalmente mais robustos a desbalanceamento de classes que os métodos rasos.

Tabela 5.6: Avaliação de métodos profundos por distribuição de classes

Distribuição	VAREJO			KAGGLE		
de classes	AUC	KS	VP	AUC	KS	VP
Balanceada	0,723	0,357	0,732	0,765	0,375	0,646
Desbalanceada	**0,762**	**0,414**	**0,743**	**0,806**	**0,472**	**0,645**

Resumindo a análise dos métodos rasos e profundos avaliados neste cenário de experimentos, podemos afirmar que:

1. Confirmando a intuição comum, métodos profundos apresentaram melhores resultados que os métodos rasos, sendo que a CNN proposta por Yifei [2016] alcançou os melhores resultados na base VAREJO e a DBN proposta por Luo et al. [2017] alcançou o melhor resultado na base KAGGLE. Dentre os métodos rasos, o Gradient Boosting foi o que apresentou os melhores resultados em ambas as coleções de dados;

2. Métodos rasos que utilizam balanceamento em nível de dados têm melhor desempenho na classificação da classe minoritária do que métodos rasos que lidam com o desbalanceamento real do problema de pontuação de crédito;

3. Dentre as abordagens de amostragem de dados, a amostragem baseada em *Undersampling* aleatório apresentou os melhores resultados deste cenário de experimento. Por outro lado, a amostragem baseada em amostras sintéticas criadas por ADASYN apresentou o pior desempenho;

4. Métodos profundos se mostram mais robustos aos desbalanceamento de classes do que métodos rasos, quando consideramos a métrica de acerto na classificação da classe minoritária;

5. Os métodos profundos que não usam estratégia de pré-processamento para balanceamento de classes obtiveram o melhor desempenho deste cenário de experimentos, superando métodos rasos e profundos que utilizam amostragem de dados no pré-processamento das bases;

5.5.3 Inferência do melhor valor para o hiperparâmetro *p* em *p*-Classification

Uma vez que a AWT (Eq.4.9) representa uma variante da função usada no trabalho de Ertekin & Rudin [2011], avaliamos o desempenho da DFN proposta na Seção 5.3 usando a p-Classification original. A p-Classification é uma função assimétrica que usa o parâmetro p para penalizar casos em que instâncias negativas que são classificadas como positivas. Adotando uma abordagem empírica para encontrar o melhor valor de p, geramos um conjunto aleatório de 5 números inteiros entre 2 e 20, obtendo como resultado o conjunto {3, 5, 9, 11, 18}.

A Tabela 5.7 mostra os resultados para cada valor de p, em cada uma das nossas duas coleções de dados. Na coleção de VAREJO, $p = 3$ obteve o melhor resultado, com $AUC = 0.821, KS = 0.490$ e $VP = 0.78$. Por sua vez, na coleção KAGGLE, $p = 5$ atingiu os melhores resultados, com $AUC = 0.860, KS = 0.563$ e $VP = 0.81$.

Tabela 5.7: Desempenho de diferentes valores de p

	VAREJO			KAGGLE		
p	AUC	KS	VP	AUC	KS	VP
3	**0.821**	**0.490**	**0.78**	0.851	0.542	0.80
5	0.812*	0.479*	0.78	**0.860***	**0.563***	**0.81**
9	0.803*	0.459*	0.77	0.851	0.554*	0.80
11	0.800*	0.454*	0.77	0.842*	0.539*	0.79
18	0.755*	0.390*	0.76*	0.844*	0.537*	0.79

Como resultado deste cenário de experimentos, a DFN que usa p-Classification como função de perda será avaliada na próxima seção utilizando $p = 3$ na coleção VAREJO e $p = 5$ na coleção KAGGLE.

5.5.4 Avaliação de modelos com aprendizagem sensível a custo

Nesta seção, apresentamos os resultados dos experimentos para abordagens baseadas em algoritmos, onde os modelos lidam com o problema de desbalanceamento de dados modificando as funções de perda ou o processo de aprendizado.

Neste cenário, utilizamos os método de Yifei [2016] e Luo et al. [2017] como *baselines* das coleções de VAREJO e KAGGLE, respectivamente, por terem apresentado os melhores resultados entre métodos avaliados nas seções anteriores. Para padronizar a execução dos demais experimentos, utilizamos a arquitetura de DFN explicada na Seção 5.3 para avaliar o uso das nossas funções de custo e da nossa estratégia de aprendizado alternado. Além disso, usamos essa DFN para avaliar (i) o uso de entropia cruzada como função de custo, por ser uma função comumente usada em modelos profundos de classificação; (ii) a função proposta por Yan et al. [2003], por otimizar a AUC, através de uma aproximação; (iii) a estratégia de aprendizagem proposta por Khan et al. [2018], com função de perda ponderada dada pelas Equações 3.19 e 3.20, por representar uma técnica ponderada com descoberta automática dos pesos; (iv) a função de perda p-Classification, proposta por Ertekin & Rudin [2011], com seus melhores valores de p, por ter servido de inspiração para a AWT; e (v) a Ranknet proposta por Burges et al. [2005], usando uma rede siamesa, por representar uma ténica baseada em pares que otimiza a definição direta da AUC (cf. Equação 3.22).

A Tabela 5.8 apresenta os resultados alcançados na coleção do VAREJO. Como observado, nossas funções assimétricas de perda, associadas à nossa abordagem de aprendizagem alternada, obtiveram os melhores resultados de AUC e de KS neste cenário de experimentos. Em particular, a AWF, que considera 4 fatores de ponderação, apresentou o melhor resultado geral, com $AUC = 0,831, KS = 0,539, VP = 0,78, Ganho = 2,214$. Contudo, é possível verificar que o método proposto por Yifei [2016] obteve o melhor resultado na tarefa de classificação da classe minoritária, alcançando $VP = 0,81$, indicando que esse método capturou melhor certas características da população inadimplente do que as nossas abordagens.

A Tabela 5.9 apresenta os resultados alcançados na coleção do KAGGLE, onde podemos verificar que as funções de Ertekin & Rudin [2011], Entropia cruzada e AWF superaram o desempenho do *baseline* e atingiram o mesmo valor de $AUC = 0.86$. Contudo, nossa abordagem que utiliza AWF como função de perda obteve os melhores resultados nas outras métricas, apresentando $KS = 0.597$ e $VP = 0.78$. Nesse sentido, é possível perceber que, de forma geral, os métodos deste cenário alcançaram valores similares de AUC, apresentando diferenças maiores nas métricas de KS e VP, refor-

Tabela 5.8: Abordagens de balanceamento em nível de algoritmo - VAREJO

Método	AUC	KS	VP	Ganho(%)
Yifei [2016]	0,813	0,504	0,81	-
Ertekin & Rudin [2011]	0,821	0,490	0,78*	0,984
Entropia cruzada	0,812	0,474*	0,77*	-0,123
Yan et al. [2003]	0,814	0,479*	0,77*	0,123
Khan et al. [2018]	0,814	0,511	0,75*	0,123
Burges et al. [2005]	0,787*	0,432*	0,78*	-3,235*
AWT	0,829*	0,527*	0,78*	1,968*
AWF	**0,831***	**0,539***	**0,78***	**2,214***

çando a importância em utilizarmos mais de uma medida de desempenho. Note que, diferente do que aconteceu na coleção VAREJO, na coleção KAGGLE todos os métodos apresentaram taxa de verdadeiro positivo (VP) superior à taxa apresentada pelo *baseline*, mostrando que, nessa coleção, os métodos que foram avaliados conseguiram caracterizar a amostra de clientes inadimplente com maior precisão.

Tabela 5.9: Abordagens de balanceamento em nível de algoritmo - KAGGLE

Método	AUC	KS	VP	Ganho(%)
Luo et al. [2017]	0,841	0,580	0,62	**-**
Ertekin & Rudin [2011]	**0,860***	**0,563***	**0,76***	**2,259***
Entropia cruzada	**0,860***	**0,563***	**0,76***	**2,259***
Yan et al. [2003]	0,859*	0,562*	0,77*	2,140*
Khan et al. [2018]	0,851*	0,583	0,77*	1,189*
Burges et al. [2005]	0,835	0,529*	0,75*	-0,773
AWT	0,855*	0,594*	0,71*	1,665*
AWF	**0,860***	**0,597***	**0,78***	**2,259***

Ao final da execução do cenário atual de experimentos, avaliamos o desempenho médio dos modelos dos modelos profundos por nível de balanceamento utilizado. Para isso, os modelos foram agrupados conforme descrito na Tabela 5.10, onde podemos verificar que os métodos que realizaram balanceamento de classes em nível de algoritmos

obtiveram os melhores resultados em todas as métricas avaliadas, nas duas coleções de dados.

Tabela 5.10: Desempenho por nível de balanceamento

Nível de	VAREJO			KAGGLE		
Balanceamento	AUC	KS	VP	AUC	KS	VP
Sem abordagem	0,762	0,414	0,743	0,805	0,472	0,645
Dados	0,729	0,373	0,730	0,768	0,378	0,650
Atributos	0,714	0,334	0,735	0,762	0,370	0,640
Algoritmo	**0,815**	**0,498**	**0,772**	**0,852**	**0,573**	**0,760**

As Figuras 5.2 e 5.3 apresentam a evolução dos pesos, responsáveis pela assimetria nas funções AWT e AWF, ao longo das épocas de treino alternado. Na Figura 5.2 observamos que a punição dos falso-positivos (e recompensa dos verdadeiros negativos) cresce à medida que o tempo avança, com $p \approx 2.2$ ao fim da época 25. Um efeito corresponde é observado na Figura 5.3, onde os pesos p (punição dos falso-positivos) e p' (recompensa dos verdadeiros negativos) crescem de forma bastante similar (e rápida). Contudo, note que a punição aos falso-negativos (peso q) também cresce em relação à recompensa aos verdadeiros-positivos (peso q'), resultando assim em mais assimetria. Como a função AWF leva aos melhores resultados, podemos dizer que estes estão associados à maior assimetria dos custos envolvidos.

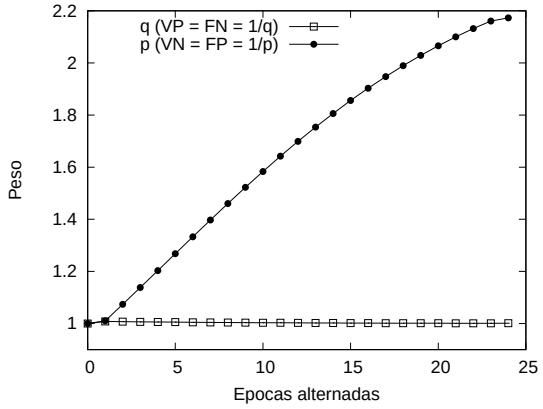

Figura 5.2: Evolução dos parâmetros p e q

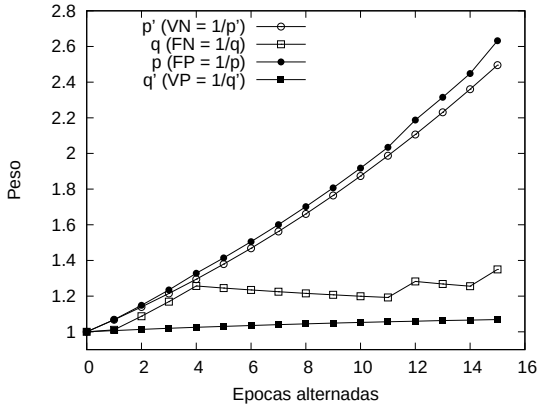

Figura 5.3: Evolução dos parâmetros p, p', q' e q ao longo do treinamento.

Nós agora observamos, nas Figuras 5.4 e 5.5, o impacto do desbalanceamento em nossas funções assimétricas, AWT e AWF, quando comparadas à sua versão simétrica,

p-Classification, $p = 1$. Para tanto, as coleções VAREJO e KAGGLE foram amostradas para criar nove outras coleções com grau de desbalanceamento crescente. Nas coleções derivadas de VAREJO, Figura 5.4, podemos observar que tanto AWT quanto AWF apresentam grande ganho sobre p-Classification. De forma geral, as três funções são robustas ao desbalanceamento, com AWF até melhorando o desempenho à medida que aumenta o desbalanceamento. No caso das coleções derivadas de KAGGLE, Figura 5.5, observamos que as três funções tem desempenho similar para o caso balanceado, com p-Classification piorando seu desempenho enquanto AWF, novamente, melhora o seu. Neste caso, AWT permaneceu com desempenho relativamente estável independente do grau de desbalanceamento. Estes resultados sugerem que as funções assimétricas usadas foram efetivamente melhores que a função simétrica para lidar com desbalanceamento.

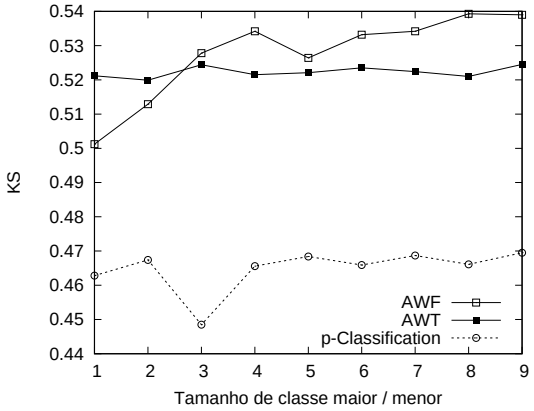

Figura 5.4: KS por grau de desbalanceamento em VAREJO.

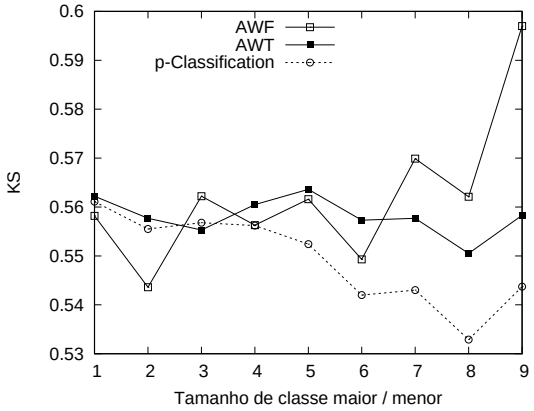

Figura 5.5: KS por grau de desbalanceamento em KAGGLE.

Como resultado deste cenário de experimentos, podemos concluir que:

1. Os métodos profundos que usam balanceamento em nível de algoritmos apresentam bons resultados na tarefa de classificação da classe minoritária, em bases de dados com alto grau de desbalanceamento.

2. Nossas duas abordagens propostas, AWF e AWT, apresentaram resultados competitivos nas duas coleções utilizadas, sendo que AWF obteve os melhores resultados de AUC e KS na coleção VAREJO, e foi uma das três melhores abordagens em AUC da coleção KAGGLE, superando as demais em KS e VP. Dado o desempenho de AWF, a maior assimetria entre os custos envolvidos parece contribuir para os melhores resultados.

3. Métodos que utilizam balanceamento em nível de algoritmo obtiveram os melhores resultados de AUC, KS e VP entre todos os métodos avaliados.

4. Quando comparadas à sua versão simétrica, AWT e AWF obtiveram melhores resultados em quase todos os níveis de desbalanceamento.

Na seção a seguir, apresentamos nosso último cenário de experimentos, voltado à avaliação do tempo de treinamento dos modelos.

5.5.5 Avaliação de tempo de treinamento

Em nosso quinto cenário de experimentos, estudamos o tempo médio necessário para o treinamento dos modelos profundos estudados na seção anterior. Em todos os casos, os dados necessários foram carregados para a memória RAM e esse tempo de carregamento não foi adicionado ao tempo total de treinamento.

A base de treinamento da coleção VAREJO foi composta por 326.625 instâncias e a base da coleção KAGGLE foi composta por 102.246. Contudo, esses números foram diferentes para o método de Burges et al. [2005] pois, em modelos *pairwise*, cada instância de entrada corresponde a um par de entidades. Como o número de pares é quadrático em relação ao número de amostras na entrada, eles devem ser gerados à medida que são usados em uma otimização SGD. Contudo, este tipo de geração em tempo real é muito lenta pois, a cada lote, teríamos que selecionar amostras positivas e negativas e pareá-las. Assim, optamos por uma estratégia alternativa, onde amostramos apenas mil instâncias por coleção e geramos os pares antecipadamente para estas amostras. Com isso, viabilizamos o custo por lote. A amostragem, contudo, naturalmente diminui a diversidade de casos que o algoritmo vê, mesmo embora da amostra de mil instâncias seja obtida uma base de treinamento com 1 milhão de pares.

Nestes experimentos, 5% da base foi usada para validação. O controle de *overfitting* foi feito com parada antecipada usando janelas de 10 resultados, com lotes de tamanho 128 e validação cruzada de dez partições. O código foi instrumentado para que, durante o treinamento, fosse possível estimar a quantidade de épocas que o modelo leva para convergir (Total de épocas) e o tempo médio, em milissegundos, necessário para o processamento de um lote. A partir dos dados de configuração e dos dados coletados, calculamos a quantidade média de lotes processados por época (Batches / época), o tempo médio, em segundos, necessário por época e o tempo total de treinamento, em minutos (Tempo total).

A Tabela 5.11 apresenta os resultados alcançados na coleção VAREJO. Nessa tabela, podemos observar que, como esperado, o método de ranking de Burges et al.

[2005], que usa pares de instâncias para aprender a ordem do ranking, foi mais lento que os demais, precisando de 18,76 milissegundos por *batch*, 139,26 segundos por época e 51,39 minutos para concluir o treinamento do modelo. Todos os demais métodos apresentaram tempo de processamento por lote relativamente similar. Entre estes, os nossos métodos apresentam os maiores tempos de treino devido ao maior número de épocas necessárias para o processo de aprendizado alternado.

Como resultado, AWT precisou de 98,86 épocas e 45,44 minutos de treinamento, enquanto AWF precisou de 69,79 épocas e 32,68 minutos para completar o treinamento do modelo. A conversão mais rápida da AWF do que a da AWT foi inesperada porque, intuitivamente, esperávamos que a otimização de quatro variáveis fosse mais complexa que a otimização de duas. Com isso, a AWF, além de apresentar os melhores resultados para as métrica AUC, KS e VP, apresentou o menor tempo de processamento entre os nossos métodos.

Tabela 5.11: Avaliação do tempo de treinamento de modelos profundos - VAREJO

Método	Total de épocas	Total de Batches	Tempo / batch (ms)	Tempo / época (seg)	Tempo total (min)
Entropia cruzada	15,25	2.424,17	11,64	28,21	7,17
Ertekin & Rudin [2011] (p=5)	18,35	2.424,17	11,70	28,36	8,67
Ertekin & Rudin [2011] (p=3)	27,63	2.424,17	11,61	28,15	12,96
Yan et al. [2003]	44,37	2.424,17	11,68	28,30	20,93
Khan et al. [2018]	46,71	2.424,17	13,28	32,19	25,06
AWF	69,79	2.424,17	11,59	28,10	32,68
AWT	98,86	2.424,17	11,38	27,58	45,44
Burges et al. [2005]	22,14	7.421,88	18,76	139,26	51,39

A Tabela 5.12 apresenta os resultados dos experimentos realizados na coleção KAGGLE. Ao analisarmos o desempenho dos métodos, podemos notar que, assim como aconteceu na coleção VAREJO, o método de ranking de Burges et al. [2005] obteve o maior tempo de treinamento, precisando de 19 milissegundos por *batch*, 144,9 segundos por época e 47,12 minutos para concluir o treinamento do modelo. Como antes, nossos métodos demoraram mais para convergir, atingindo tempos maiores de treino quando comparados aos demais métodos não baseados em pares. Entre os dois, o AWF foi mais rápido novamente. Diferente de antes, o método proposto por Ertekin & Rudin [2011] ($p = 3$) também demorou a convergir.

Tabela 5.12: Avaliação do tempo de treinamento de modelos profundos - KAGGLE

Método	Total de épocas	Total de Batches	Tempo / batch (ms)	Tempo / época (seg)	Tempo total (min)
Entropia cruzada	15,23	758,86	11,26	8,54	2,17
Yan et al. [2003]	15,77	758,86	12,08	9,17	2,41
Ertekin & Rudin [2011] (p=5)	30,46	758,86	11,55	8,77	4,45
Khan et al. [2018]	49,81	758,86	11,07	8,40	6,98
AWF	57,73	758,86	11,64	8,83	8,50
Ertekin & Rudin [2011] (p=3)	62,43	758,86	11,65	8,84	9,20
AWT	78,54	758,86	11,32	8,59	11,24
Burges et al. [2005]	19,51	7.421,88	19,52	144,90	47,12

Como resultado deste cenário de experimentos, podemos concluir que:

- O alto custo no uso de técnicas baseadas em pares não é necessariamente compensador em termos de desempenho.
- Apesar da produção de rankings melhores e conveniência na obtenção de hiperparâmetros, nossos métodos precisam de maior número de épocas para convergir devido ao emprego de um treinamento alternado.

5.6 Considerações Finais

Neste capítulo, apresentamos os experimentos e resultados obtidos. Iniciamos com a metodologia de avaliação, cobrindo coleções, arquitetura, técnica experimental e ambientes usados. Em termos de experimentos, avaliamos o desempenho dos modelos sem estratégias para lidar com balanceamento, considerando estratégias de amostragem e sensibilidade a custos. Além disso, avaliamos a eficiência do modelo durante o treino.

Em nossos resultados, observamos que métodos profundos superam os rasos, mesmo quando estes últimos lidam com balanceamento e os primeiros, não. Em geral, os métodos profundos se mostraram mais robustos ao desbalanceamento, talvez pelo emprego de muitas técnicas para garantir generalização, tais como *dropout*. O uso de técnicas para lidar com desbalanceamento tem impacto no desempenho obtido para a classe minoritária, com a técnica de *undersampling* alcançando os melhores resultados.

Entre os métodos profundos, as técnicas que usam balanceamento em nível de algoritmo superaram as demais, com nossas funções AWT e AWF alcançando os melhores resultados, especialmente na coleção VAREJO, onde o desbalanceamento é maior. Quando comparadas à sua versão simétrica, AWT e AWF se saíram melhor, em geral, quando houve desbalanceamento. Embora nosso método seja mais rápido que uma abordagem baseada em pares de instâncias, ele converge mais lentamente devido ao uso da técnica de aprendizado alternado, quando comparado aos outros métodos, que não se baseiam em pares.

Capítulo 6

Conclusões

Desde a década de 40, métodos estatísticos são utilizados para ajudar em previsão de risco de crédito. Com o passar dos anos, abordagens baseadas em aprendizagem de máquina passaram a ser utilizadas com sucesso, para resolver esses problemas. Contudo, alguns avanços recentes observados na literatura de aprendizagem de máquina ainda não haviam sido devidamente explorados no contexto de risco de crédito.

O primeiro destes avanços é o emprego de abordagens neurais profundas com enorme sucesso em áreas como visão computacional, classificação de imagens e reconhecimento de fala. A adoção de tais técnicas em previsão de risco de crédito ainda é tímida, como observamos em nossa revisão de literatura.

O segundo é a caracterização de classes de funções de custo, as próprias estritas compostas, que podem ser usadas para resolver de forma equivalente problemas de classificação binária e ranqueamento bipartido. A previsão de risco em crédito é tipicamente tratada como um problema de classificação binária. Este, contudo, é um problema desbalanceado por natureza, com muito mais bons que maus pagadores. Essa característica pode influenciar modelos de classificação, necessitando o acréscimo de poder computacional para balanceamento das amostras. Ao mesmo tempo, na maioria das aplicações de risco, é necessário garantir que clientes com maior probabilidade de inadimplência recebam pontuação de risco maior que a pontuação de clientes com menor probabilidade. Dependendo do risco de um grupo de pessoas, diferentes políticas de crédito podem ser usadas. Ou seja, este é claramente um problema de ranqueamento bipartido, que deveria ser avaliado usando métricas de ranqueamento como KS e AUC. Contudo, como são tratados como métodos de classificação, a maioria das técnicas não é projetada para otimizar *ranking*.

Ao tirar proveito de classes de funções próprias estritas compostas, conseguimos projetar funções de custo assimétricas, para uso em arquiteturas profundas, que com-

binam as vantagens de métodos de ranquamento (robustez natural a desbalanceamento e ênfase em setores do ranking de maior interesse como o topo) com as vantagens de métodos de classificação (eficiência de processamento). Além disso, fomos capazes de propor funções em que todos os hiper-parâmetros envolvidos são aprendidos automaticamente no processo, dispensando fases preliminares de ajustes.

Esta pesquisa foi motivada por uma série de questões, que respondemos nos parágrafos a seguir.

Qual é o impacto de técnicas de tratamento de desbalanceamento, tais como técnicas de sampling*, em modelos profundos em geral e naqueles aplicados a* credit scoring*, rasos ou profundos?*

Observamos que os métodos profundos avaliados são em geral mais robustos a desbalanceamento que os rasos, quando nenhuma técnica de pré-processamento para lidar com desbalanceamento é aplicada. O uso das técnicas de balanceamento, nos métodos rasos, claramente melhora o desempenho na classe minoritária, o que era esperado para o problema. Dentre as abordagens de amostragem de dados, a amostragem baseada em *undersampling* aleatório apresentou os melhores resultados deste cenário de experimento. Por outro lado, a amostragem baseada em amostras sintéticas criadas por ADASYN apresentou o pior desempenho.

Os modelos profundos que usam funções de custo assimétricas são mais robustos a desbalanceamento e mais adequados a diferentes políticas de crédito que modelos profundos que usam funções de custo simétricas?

De forma geral, os métodos profundos, quando aplicando técnicas de balanceamento em nível de algoritmo (técnicas sensíveis ao custo e assimétricas), apresentam os melhores resultados. Em um estudo específico, observamos que as funções assimétricas AWT e AWF superam consistentemente métodos baseados na sua versão simétrica, p-Classification ($p = 1$), independente do grau de desbalanceamento. Para AWF, estes resultados são especialmente relevantes quanto é maior o desbalanceamento.

As funções assimétricas usadas, por sua própria natureza, fornecem importância distinta para diferentes partes do ranking, como o topo. Contudo, dado o processo de parametrização automática via aprendizado alternado, não é possível manipular diretamente os parâmetros que controlam o impacto sobre diferentes pontos do ranking, de forma que o método proposto, de fato, não oferece tanta flexibilidade em termos de política de crédito.

Esses modelos são tão eficientes em termos de custo de treinamento e conveniência de parametrização quanto outras abordagens profundas encontradas na literatura?

Ao empregarmos uma técnica de aprendizado alternado, fomos capazes de criar um método que praticamente dispensa uma fase preliminar de refinamento e parâmetros. Contudo, ele acaba convergindo mais lentamente, uma vez que alterna múltiplas fases de aprendizado de parâmetros da função de custo com parâmetros da rede neural. Apesar disto, ele ainda é muito mais rápido que métodos baseados em pares de amostras de entrada, como o RankNet.

6.1 Limitações

Ao longo desta pesquisa, nós enfrentamos algumas dificuldades e limitações, entre as quais citamos:

1. Dificuldade em ter acesso a bases de dados públicas que fossem realmente grandes e adequadas ao treinamento de arquiteturas profundas. Embora estas bases sejam comuns no dia-a-dia de empresas credoras como grandes varejos, bancos e financeiras, pouca informação é disponibilizada publicamente, em especial, no Brasil. Além disso, a informação disponibilizada fora do Brasil nem sempre é adequada a cenários de crédito típicos de países em desenvolvimento, como os normalmente observados no varejo Brasileiro.

2. Embora estudos sobre arquiteturas profundas sejam comumente publicados com acesso a códigos, este não é o caso no contexto específico de *credit scoring*. Nenhum dos métodos que revisamos na literatura tinha código publicamente disponível, o que nos obrigou a implementação de vários dos métodos que usamos como referência nesta pesquisa. Embora tenhamos feito o melhor esforço para implementar o que foi descrito nos trabalhos, nem sempre estas descrições eram detalhadas o suficiente, o que nos obrigou, em alguns casos, à escolha de parâmetros e configurações que achamos mais adequadas;

3. Originalmente, pretendíamos realizar engenharia de atributos (incluindo o uso de dados adicionais de comportamento) para melhorar a representação dos clientes. Isto, contudo, é complexo quando tanto as bases públicas quanto as privadas são completamente anonimizadas, de forma que não é possível interpretar a semântica das variáveis;

6.2 Trabalhos Futuros

Ao longo desta pesquisa, observamos muitas outras oportunidades de trabalho que descrevemos a seguir.

Políticas de crédito podem ter objetivos muito diferentes. Por exemplo, um método poderia sacrificar a ordenação em geral dos clientes desde que fosse muito mais correto no topo e na parte mais baixa do ranking. Isso poderia ser feito, por exemplo, através da adoção de funções com penalidades/recompensas mais agressivas para diferentes tipos de erro. Embora as funções propostas neste trabalho permitam alguma variação de importância em segmentos do ranking, em geral, elas não são tão flexíveis.

Contudo, usando o método proposto por Menon & Williamson [2016] é realmente possível projetar funções com características mais flexíveis. Por exemplo, ao longo de nossa pesquisa, de fato, nós propusemos funções bem distintas dentro da classe das próprias estritas compostas. Um exemplo é baseado na função de Bezier dada pela Equação 6.1.

$$\ell(s,p,q,\lambda_1,\lambda_2) = \left(\frac{p(1+\lambda_1 - s\lambda_1 + \lambda_1^2 - (1+\lambda_1)\sqrt{1-2s\lambda_1+\lambda_1^2})}{2\lambda_1^2}, \right. \\ \left. \frac{q(1-\lambda_2 - s\lambda_2 + \lambda_2^2 + (-1+\lambda_2)\sqrt{1-2s\lambda_2+\lambda_2^2})}{2\lambda_2^2}\right) \tag{6.1}$$

onde s é o escore cujo custo é estimado, p e q são os pesos dos custos parciais e λ_1 e λ_2 podem ser usados para controlar a agressivade da recompensa ou punição produzida. Ao longo desta pesquisa decidimos não estudar esta função devido à sua complexidade e possíveis dificuldades com estabilidade numérica. As funções que adotamos, de fato, são baseadas em operações exponenciais simples de derivar e com implementações robustas e eficientes em bibliotecas vetoriais como tensorflow[1] e pytorch[2]. Ainda assim, seria interessante estudar o desempenho de funções assimétricas diferentes, como a de Bezier, que poderia ser mais útil em termos de política de crédito.

No caso específico de crédito concedido no varejo no Brasil, seria interessante estudar o impacto do uso de atributos associados à compra sendo realizada como, por exemplo, o tipo de produto sendo adquirido. A intuição aqui é que diferentes tipos de produtos (por exemplo, os de menor liquidez como sofás) podem estar menos associados com inadimplência que os de maior liquidez, como telefones celulares.

Outro problema importante de *credit scoring* é que as coleções de treino são

[1] `https://www.tensorflow.org/`
[2] `https://pytorch.org/`

tipicamente formadas por clientes para os quais se observou comportamento, já que eles foram rotulados como bons ou maus. Contudo, esses modelos devem ser usados sobre populações para os quais vai ou não se observar comportamento, já que o modelo é usado para selecionar os clientes que, mais tarde, serão observados. Em outras palavras, as populações de treino e teste normalmente diferem neste contexto. Seria interessante estudar formas de lidar com esta diferença de população para alcançar modelos que tem bom desempenho e boa estabilidade, apesar das diferenças.

Referências Bibliográficas

Abdallah, A.; Maarof, M. A. & Zainal, A. (2016). Fraud detection system: A survey. *Journal of Network and Computer Applications*, 68:90 – 113.

Addo, P. M.; Guegan, D. & Hassani, B. (2018). Credit risk analysis using machine and deep learning models. *Risks*, 6(2):38.

Altman, E. I. (1968). Financial ratios, discriminant analysis and the prediction of corporate bankruptcy. *The journal of finance*, 23(4):589--609.

Atiya, A. F. (2001). Bankruptcy prediction for credit risk using neural networks: A survey and new results. *IEEE Transactions on neural networks*, 12(4):929--935.

Baesens, B.; Van Gestel, T.; Viaene, S.; Stepanova, M.; Suykens, J. & Vanthienen, J. (2003). Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, 54(6):627--635.

Bartlett, P. L.; I., J. M. & McAuliffe, J. D. (2006). Convexity, classification, and risk bounds.

Batista, G. E. A. P. A.; Prati, R. C. & Monard, M. C. (2004). A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.*, 6(1):20--29.

Beijbom, O.; Saberian, M. J.; Kriegman, D. J. & Vasconcelos, N. (2014). Guess-Averse Loss Functions For Cost-Sensitive Multiclass Boosting. Em *Proceedings of the 31st International Conference on Machine Learning, Cycle 2*, volume 32 of *JMLR Proceedings*, pp. 586--594. JMLR.org.

Bellotti, T. & Crook, J. (2009). Support vector machines for credit scoring and discovery of significant features. *Expert Systems with Applications*, 36(2):3302--3308.

Boser, B. E.; Guyon, I. M. & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. Em *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144--152. ACM.

Branco, P.; Torgo, L. & Ribeiro, R. P. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys (CSUR)*, 49(2):1--50.

Breiman, L. (2017). *Classification and regression trees*. Routledge.

Brown, I. & Mues, C. (2012). An experimental comparison of classification algorithms for imbalanced credit scoring data sets. *Expert Systems with Applications*, 39(3):3446 – 3453.

Burges, C.; Shaked, T.; Renshaw, E.; Lazier, A.; Deeds, M.; Hamilton, N. & Hullender, G. (2005). Learning to rank using gradient descent. Em *Proceedings of the 22nd international conference on Machine learning*, ICML '05, pp. 89--96, New York, NY, USA. ACM.

Cao, Z.; Qin, T.; Liu, T.-Y.; Tsai, M.-F. & Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. Relatório técnico MSR-TR-2007-40.

Chawla, N. V. (2009). Data mining for imbalanced datasets: An overview. Em *Data mining and knowledge discovery handbook*, pp. 875--886. Springer.

Chawla, N. V.; Bowyer, K. W.; Hall, L. O. & Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321--357.

Chen, C.-C. & Li, S.-T. (2014). Credit rating with a monotonicity-constrained support vector machine model. *Expert Systems with Applications*, 41(16):7235--7247.

Chen, N.; Ribeiro, B. & Chen, A. (2016). Financial credit risk assessment: a recent review. *Artificial Intelligence Review*, 45(1):1--23.

Chen, X.-w. & Wasikowski, M. (2008). Fast: a roc-based feature selection metric for small samples and imbalanced data classification problems. Em *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 124--132. ACM.

Chen, Y.-S. & Cheng, C.-H. (2013). Hybrid models based on rough set classifiers for setting credit rating decision rules in the global banking industry. *Knowledge-Based Systems*, 39:224--239.

Chung, Y.-A.; Lin, H.-T. & Yang, S.-W. (2015). Cost-aware pre-training for multiclass cost-sensitive deep learning. *arXiv preprint arXiv:1511.09337*.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273--297.

Crook, J. N.; Edelman, D. B. & Thomas, L. C. (2007). Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, 183(3):1447--1465.

Departamento Intersindical de Estatísticas e Estudos Socioeconômicos - DIEESE (2018). Análise da evolução do crédito no período recente 2014 - 2017. [Online; accessed 14-June-2019].

di Basilea per la vigilanza bancaria, C. (2004). *International convergence of capital measurement and capital standards: a revised framework*. Bank for International Settlements.

Diniz, C. & Louzada, F. (2013). Métodos estatísticos para análise de dados de crédito. Em *6th Brazilian Conference on Statistical Modeling in Insurance and Finance, Maresias-SP*, p. 129.

Durand, D. (1941). *Risk Elements in Consumer Instalment Financing, Technical Edition*. National Bureau of Economic Research, Inc.

Ertekin, Ş. & Rudin, C. (2011). On equivalence relationships between classification and ranking algorithms. *Journal of Machine Learning Research*, 12(Oct):2905--2929.

Fernández, A.; García, S.; Galar, M.; Prati, R. C.; Krawczyk, B. & Herrera, F. (2018). *Learning from imbalanced data sets*. Springer.

Fernández, A.; García, S.; Herrera, F. & Chawla, N. V. (2018). Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary. *J. Artif. Int. Res.*, 61(1):863--905.

Ferri, C.; Hernández-Orallo, J. & Modroiu, R. (2009). An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27--38.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(7):179–188.

Freund, Y. & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139.

Goodfellow, I.; Bengio, Y. & Courville, A. (2016). *Deep learning*. MIT press.

Grunert, J.; Norden, L. & Weber, M. (2005). The role of non-financial factors in internal credit ratings. *Journal of Banking & Finance*, 29(2):509--531.

Guyon, I. & Elisseeff, A. (2003). An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157--1182.

Ha, V.-S. & Nguyen, H.-N. (2016). Credit scoring with a feature selection approach based deep learning. Em *MATEC Web of Conferences*, volume 54, p. 05004. EDP Sciences.

Haixiang, G.; Yijing, L.; Shang, J.; Mingyun, G.; Yuanyue, H. & Bing, G. (2017). Learning from class-imbalanced data: Review of methods and applications. *Expert Systems with Applications*, 73:220 – 239.

Han, C. E.; Arbib, M. A. & Schweighofer, N. (2008). Stroke rehabilitation reaches a threshold. *PLoS computational biology*, 4(8):e1000133.

Hand, D. J. & Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: a review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160(3):523--541.

Hand, D. J. & Kelly, M. G. (2002). Superscorecards. *IMA Journal of Management Mathematics*, 13(4):273--281.

He, H.; Bai, Y.; Garcia, E. A. & Li, S. (2008). Adasyn: Adaptive synthetic sampling approach for imbalanced learning. Em *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pp. 1322--1328. IEEE.

He, H. & Ma, Y. (2013). *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons.

He, H.; Zhang, W. & Zhang, S. (2018). A novel ensemble method for credit scoring: Adaption of different imbalance ratios. *Expert Systems with Applications*, 98:105--117.

Henley, W. & Hand, D. J. (1996). A k-nearest-neighbour classifier for assessing consumer credit risk. *The statistician*, pp. 77--95.

Hinton, G. E. (2012). A practical guide to training restricted boltzmann machines. Em *Neural networks: Tricks of the trade*, pp. 599--619. Springer.

Hinton, G. E.; Osindero, S. & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527--1554.

Hinton, G. E. & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *science*, 313(5786):504--507.

Horrigan, J. O. (1966). The determination of long-term credit standing with financial ratios. *Journal of Accounting Research*, pp. 44--62.

Huang, C.-L.; Chen, M.-C. & Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert systems with applications*, 33(4):847--856.

Huang, S.-C. & Day, M.-Y. (2013). A comparative study of data mining techniques for credit scoring in banking. Em *Information Reuse and Integration (IRI), 2013 IEEE 14th International Conference on*, pp. 684--691. IEEE.

Huang, Z.; Chen, H.; Hsu, C.-J.; Chen, W.-H. & Wu, S. (2004). Credit rating analysis with support vector machines and neural networks: a market comparative study. *Decision support systems*, 37(4):543--558.

Hwang, R.-C.; Cheng, K. & Lee, C.-F. (2009). On multiple-class prediction of issuer credit ratings. *Applied Stochastic Models in Business and Industry*, 25(5):535--550.

Kaplan, R. S. & Urwitz, G. (1979). Statistical models of bond ratings: A methodological inquiry. *Journal of business*, pp. 231--261.

Khan, S. H.; Hayat, M.; Bennamoun, M.; Sohel, F. A. & Togneri, R. (2018). Cost-sensitive learning of deep feature representations from imbalanced data. *IEEE transactions on neural networks and learning systems*, 29(8):3573--3587.

Khashman, A. (2010). Neural networks for credit risk evaluation: Investigation of different neural models and learning schemes. *Expert Systems with Applications*, 37(9):6233--6239.

Khashman, A. (2011). Credit risk evaluation using neural networks: Emotional versus conventional models. *Applied Soft Computing*, 11(8):5477--5484.

Kim, K.-j. & Ahn, H. (2012). A corporate credit rating model using multi-class support vector machines with an ordinal pairwise partitioning approach. *Computers & Operations Research*, 39(8):1800--1811.

Kim, Y.; Lee, H. & Provost, E. M. (2013). Deep learning for robust feature generation in audiovisual emotion recognition. Em *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 3687--3691. IEEE.

Kira, K. & Rendell, L. A. (1992). A practical approach to feature selection. Em *Machine Learning Proceedings 1992*, pp. 249--256. Elsevier.

Krizhevsky, A.; Sutskever, I. & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. Em *Advances in neural information processing systems*, pp. 1097--1105.

Kvamme, H.; Sellereite, N.; Aas, K. & Sjursen, S. (2018). Predicting mortgage default using convolutional neural networks. *Expert Systems with Applications*, 102:207--217.

Kvammea, H.; Sellereiteb, N.; Aas, K. & Sjursenc, S. (2018). Predicting mortgage default using convolutional neural networks. *Expert Syst. Appl.*, 102:207–2117.

Laha, A. (2007). Building contextual classifiers by integrating fuzzy rule based classification technique and k-nn method for credit scoring. *Advanced Engineering Informatics*, 21(3):281--291.

Lahmiri, S. (2011). A comparative study of backpropagation algorithms in financial prediction. *International Journal of Computer Science, Engineering and Applications (IJCSEA)*, 1(4):15--21.

Lango, M. & Stefanowski, J. (2018). Multi-class and feature selection extensions of roughly balanced bagging for imbalanced data. *Journal of Intelligent Information Systems*, 50(1):97--127.

Larochelle, H. & Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. Em *Proceedings of the 25th international conference on Machine learning*, pp. 536--543. ACM.

Le, D. & Provost, E. M. (2013). Emotion recognition from spontaneous speech using hidden markov models with deep belief networks. Em *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pp. 216--221. IEEE.

LeCun, Y.; Bottou, L.; Bengio, Y. & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278--2324.

Lessmann, S.; Baesens, B.; Seow, H.-V. & Thomas, L. C. (2015). Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124--136.

Lewis, E. M. (1992). *An introduction to credit scoring*. Fair, Isaac and Company.

Li, F.-C. (2009). The hybrid credit scoring strategies based on knn classifier. Em *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 1, pp. 330--334. IEEE.

Li, H.; Chung, F.-l. & Wang, S. (2015). A svm based classification method for homogeneous data. *Applied Soft Computing*, 36:228--235.

Li, S.-T.; Shiue, W. & Huang, M.-H. (2006). The evaluation of consumer loans using support vector machines. *Expert Systems with Applications*, 30(4):772--782.

Lin, S. L. (2009). A new two-stage hybrid approach of credit risk in banking industry. *Expert Systems with Applications*, 36(4):8333--8341.

Lin, W.-Y.; Hu, Y.-H. & Tsai, C.-F. (2012). Machine learning in financial crisis prediction: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4):421--436.

Liu, H. & Motoda, H. (2012). *Feature selection for knowledge discovery and data mining*, volume 454. Springer Science & Business Media.

Luo, C.; Wu, D. & Wu, D. (2017). A deep learning approach for credit scoring using credit default swaps. *Engineering Applications of Artificial Intelligence*, 65:465--470.

Lyn, T.; David, E. & Jonathan, C. (2002). Credit scoring and its applications. *Philadelphia: Society for Industrial and Applied Mathematics*.

Mangasarian, O. L. (1965). Linear and nonlinear separation of patterns by linear programming. *Operations research*, 13(3):444--452.

Martin, D. (1977). Early warning of bank failure: A logit regression approach. *Journal of banking & finance*, 1(3):249--276.

Matthews, P. (1996). Relationship of firing intervals of human motor units to the trajectory of post-spike after-hyperpolarization and synaptic noise. *The Journal of physiology*, 492(2):597--628.

Menon, A. & Elkan, C. (2011). Link prediction via matrix factorization. Em Gunopulos, D.; Hofmann, T.; Malerba, D. & Vazirgiannis, M., editores, *Machine Learning and Knowledge Discovery in Databases*, volume 6912 of *Lecture Notes in Computer Science*, pp. 437–452. Springer Berlin / Heidelberg. 10.1007/978-3-642-23783-$6_2$8.

Menon, A. K. & Williamson, R. C. (2016). Bipartite ranking: A risk-theoretic perspective. *J. Mach. Learn. Res.*, 17(1):6766--6867.

Michel, A. J. (1977). Municipal bond ratings: a discriminant analysis approach. *Journal of Financial and Quantitative Analysis*, 12(4):587--598.

Mohamed, A.-r.; Dahl, G. E.; Hinton, G. et al. (2012). Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech & Language Processing*, 20(1):14--22.

Narasimhan, H. & Agarwal, S. (2013). On the relationship between binary classification, bipartite ranking, and binary class probability estimation. Em Burges, C. J. C.; Bottou, L.; Welling, M.; Ghahramani, Z. & Weinberger, K. Q., editores, *Advances in Neural Information Processing Systems 26*, pp. 2913--2921. Curran Associates, Inc.

Nasrabadi, N. M. (2007). Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901.

Neagoe, V.-E.; Ciotec, A.-D. & Cucu, G.-S. (2018). Deep convolutional neural networks versus multilayer perceptron for financial prediction. Em *2018 International Conference on Communications (COMM)*, pp. 201--206. IEEE.

Pant, H. & Srivastava, R. (2015). A survey on feature selection methods for imbalanced datasets. *International Journal of Computer Engineering and Applications*, 9(2).

Pogue, T. F. & Soldofsky, R. M. (1969). What's in a bond rating. *Journal of financial and quantitative analysis*, 4(2):201--228.

Qian, Y.; Liang, Y.; Li, M.; Feng, G. & Shi, X. (2014). A resampling ensemble algorithm for classification of imbalance problems. *Neurocomputing*, 143:57--67.

Raj, V.; Magg, S. & Wermter, S. (2016). Towards effective classification of imbalanced data with convolutional neural networks. Em *IAPR Workshop on Artificial Neural Networks in Pattern Recognition*, pp. 150--162. Springer.

Razali, N. M.; Wah, Y. B. et al. (2011). Power comparisons of shapiro-wilk, kolmogorov-smirnov, lilliefors and anderson-darling tests. *Journal of statistical modeling and analytics*, 2(1):21--33.

Ribeiro, B. & Lopes, N. (2011). Deep belief networks for financial prediction. Em *International Conference on Neural Information Processing*, pp. 766--773. Springer.

Saberi, M.; Mirtalaie, M. S.; Hussain, F. K.; Azadeh, A.; Hussain, O. K. & Ashjari, B. (2013). A granular computing-based approach to credit scoring modeling. *Neurocomputing*, 122:100--115.

Salakhutdinov, R.; Mnih, A. & Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. Em *Proceedings of the 24th international conference on Machine learning*, pp. 791--798. ACM.

Schraudolph, N. N.; Yu, J. & Günter, S. (2007). A stochastic quasi-newton method for online convex optimization. Em *Artificial Intelligence and Statistics*, pp. 436--443.

Schreiner, M. (2000). Credit scoring for microfinance: Can it work? *Journal of Microfinance/ESR Review*, 2(2):6.

Scott, C. (2012). Calibrated asymmetric surrogate losses. *Electron. J. Statist.*, 6:958--992.

Shakeel, F.; Sabhitha, A. S. & Sharma, S. (2017). Exploratory review on class imbalance problem: An overview. Em *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–8.

Shi, Y.; Peng, Y.; Xu, W. & Tang, X. (2002). Data mining via multiple criteria linear programming: applications in credit card portfolio management. *International Journal of Information Technology & Decision Making*, 1(01):131--151.

Stein, R. M. (2007). Benchmarking default prediction models: pitfalls and remedies in model validation. *Journal of Risk Model Validation*, 1(1):77--113.

Sun, Y.; Wong, A. K. & Kamel, M. S. (2009). Classification of imbalanced data: A review. *International journal of pattern recognition and artificial intelligence*, 23(04):687--719.

SUPERVISION, B. C. O. B. (2010). Basel iii: A global regulatory framework for more resilient banks and banking systems. *Basel, Bank for International Settlements*.

Tamilselvan, P. & Wang, P. (2013). Failure diagnosis using deep belief learning based health state classification. *Reliability Engineering & System Safety*, 115:124--135.

Tang, J.; Alelyani, S. & Liu, H. (2014). Feature selection for classification: A review. *Data classification: algorithms and applications*, p. 37.

Thomas, L.; Crook, J. & Edelman, D. (2017). *Credit scoring and its applications*, volume 2. Siam.

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International journal of forecasting*, 16(2):149--172.

Tomczak, J. M. & Zieba, M. (2015). Classification restricted boltzmann machine for comprehensible credit scoring model. *Expert Syst. Appl.*, 42(4):1789–1796.

Tran, K.; Duong, T. & Ho, Q. (2016). Credit scoring model: A combination of genetic programming and deep learning. Em *Future Technologies Conference (FTC)*, pp. 145--149. IEEE.

Tsai, C.-F. & Wu, J.-W. (2008). Using neural network ensembles for bankruptcy prediction and credit scoring. *Expert systems with applications*, 34(4):2639--2649.

Vellido, A.; Lisboa, P. J. & Vaughan, J. (1999). Neural networks in business: a survey of applications (1992–1998). *Expert Systems with applications*, 17(1):51--70.

Wang, S.; Liu, W.; Wu, J.; Cao, L.; Meng, Q. & Kennedy, P. J. (2016). Training deep neural networks on imbalanced data sets. Em *2016 international joint conference on neural networks (IJCNN)*, pp. 4368--4374. IEEE.

Wang, X.; Xu, M. & Pusatli, O. T. (2015). A survey of applying machine learning techniques for credit rating: Existing models and open issues. Em *Proceedings, Part II, of the 22Nd International Conference on Neural Information Processing - Volume 9490*, ICONIP 2015, pp. 122--132, New York, NY, USA. Springer-Verlag New York, Inc.

West, D. (2000). Neural network credit scoring models. *Computers & Operations Research*, 27(11-12):1131--1152.

West, R. R. (1970). An alternative approach to predicting corporate bond ratings. *Journal of Accounting Research*, pp. 118--125.

Wiginton, J. C. (1980). A note on the comparison of logit and discriminant models of consumer credit behavior. *Journal of Financial and Quantitative Analysis*, 15(3):757--770.

Wilcoxon, F. (1992). *Individual Comparisons by Ranking Methods*, pp. 196--202. Springer New York, New York, NY.

Wilson, D. R. & Martinez, T. R. (1997). Improved heterogeneous distance functions. *Journal of artificial intelligence research*, 6:1--34.

Yan, L.; Dodier, R.; Mozer, M. C. & Wolniewicz, R. (2003). Optimizing classifier performance via an approximation to the wilcoxon-mann-whitney statistic. Em *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, pp. 848--855. AAAI Press.

Yeh, C.-C.; Lin, F. & Hsu, C.-Y. (2012). A hybrid kmv model, random forests and rough set theory approach for credit rating. *Knowledge-Based Systems*, 33:166--172.

Yeh, I.-C. & Lien, C.-h. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert Syst. Appl.*, 36(2):2473--2480.

Yifei, R. (2016). Big data algorithm applied to credit risk assessment model. *International Journal of Simulation–Systems, Science & Technology*, 17(42).

Yu, L.; Zhou, R.; Tang, L. & Chen, R. (2018). A dbn-based resampling svm ensemble learning paradigm for credit classification with imbalanced data. *Applied Soft Computing*, 69:192--202.

Zhao, P.; Hoi, S. C.; Jin, R. & YANG, T. (2011). Online auc maximization. *Research Collection School of Information Systems*.

Zhong, H.; Miao, C.; Shen, Z. & Feng, Y. (2014). Comparing the learning effectiveness of bp, elm, i-elm, and svm for corporate credit ratings. *Neurocomputing*, 128:285--295.

Zhou, L.; Lai, K. K. & Yu, L. (2010). Least squares support vector machines ensemble models for credit scoring. *Expert Systems with Applications*, 37(1):127--133.

Zhu, B.; Yang, W.; Wang, H. & Yuan, Y. (2018). A hybrid deep learning model for consumer credit scoring. Em *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pp. 205--208. IEEE.

Zweig, M. H. & Campbell, G. (1993). Receiver-operating characteristic (roc) plots: a fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39(4):561--577.