GABRIEL VITA SILVA FRANCO

ESTRATÉGIAS PARA SELEÇÃO DE HIPERPARÂMETROS EM APRENDIZADO COM PROPORÇÕES DE RÓTULOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

Orientador: Giovanni Ventorim Comarela

VIÇOSA - MINAS GERAIS
2021

Ficha catalográfica elaborada pela Biblioteca Central da Universidade Federal de Viçosa - Campus Viçosa

T
F825e
2021

Franco, Gabriel Vita Silva, 1995-
Estratégias para seleção de hiperparâmetros em aprendizado com proporções de rótulos / Gabriel Vita Silva Franco. – Viçosa, MG, 2021.
66 f. : il. (algumas color.) ; 29 cm.

Orientador: Giovanni Ventorim Comarela.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 64-66.

1. Aprendizado do computador. 2. Mineração de dados. 3. Aprendizado supervisionado (Aprendizado do computador). I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-Graduação em Ciência da Computação. II. Título.

CDD 22. ed. 006.31

GABRIEL VITA SILVA FRANCO

ESTRATÉGIAS PARA SELEÇÃO DE HIPERPARÂMETROS EM APRENDIZADO COM PROPORÇÕES DE RÓTULOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae.*

APROVADA: 08 de junho de 2021.

Assentimento:

Gabriel V. S. Franco

Gabriel Vita Silva Franco
Autor

Giovanni Ventorim Comarela

Giovanni Ventorim Comarela
Orientador

Agradecimentos

Aos meus pais e ao meu irmão Herbert por todo o apoio.
À minha companheira Maria Eduarda pelos conselhos e paciência durante todo o período do mestrado.
Ao meu orientador, professor Giovanni Comarela, por ter aceitado me orientar neste trabalho. Obrigado por todos os ensinamentos e conselhos, que me ajudaram muito a evoluir como cientista da computação, pesquisador e também como pessoa durante esses anos.
Ao professor Mark Crovella por toda sua fundamental ajuda para a realização deste trabalho.
A todos os professores do Departamento de Informática da Universidade Federal de Viçosa que passaram pela minha trajetória acadêmica durantes esses anos, especialmente aos professores Levi Lelis e André dos Santos.
Aos grandes amigos que fiz no DPI e na UFV. Em especial, agradeço ao Walysson, Felipe, Marcelo, Marcos, Jansen, Vavá, a galera da química e a do karaokê por todos os ótimos momentos durante esses anos.
Aos meu grandes amigos do bairro Floresta por estarem comigo desde sempre.
O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001.

Resumo

FRANCO, Gabriel, M.Sc., Universidade Federal de Viçosa, junho de 2021. **Estratégias para seleção de hiperparâmetros em Aprendizado com Proporções de Rótulos**. Orientador: Giovanni Ventorim Comarela.

Esta dissertação propõe estratégias para o problema de seleção de hiperparâmetros no problema de Aprendizado com Proporções de Rótulos, conhecido como LLP. Neste problema, os dados estão divididos em conjuntos, chamados *bags*, e apenas a proporção dos rótulos em cada *bag* é conhecida. Primeiro, o problema de LLP é formalmente definido. São apresentadas duas definições: uma que condiz com os cenários apresentados na literatura e outra que abrange cenários que fogem da definição padrão da literatura. Com o problema formalizado, uma estratégia genérica de seleção de hiperparâmetros para LLP é proposta. Esta estratégia divide o problema de seleção de hiperparâmetros em LLP em duas partes: divisão do conjunto de dados entre conjuntos de treino e validação e computação do erro do modelo no conjunto de validação. Para a primeira parte, são propostos três algoritmos que fazem a divisão dos dados por *bag*. Já para a segunda parte, uma nova função de erro que utiliza pesos para as *bags* baseados na informação de Fisher é proposta. Com os métodos definidos, todo o cenário de avaliação dos métodos é apresentado, incluindo novos conjuntos de dados sintéticos que incorporam aspectos práticos do problema de LLP. Os resultados mostraram que utilizar amostras com repetição e um conjunto de validação maior na seleção de hiperparâmetros traz ganhos em relação ao estado da arte em cenários de LLP Geral.

Palavras-chave: Aprendizado com Proporções de Rótulos. Seleção de hiperparâmetros. Aprendizado semi-supervisionado.

Abstract

FRANCO, Gabriel, M.Sc., Universidade Federal de Viçosa, June, 2021. **Towards a general hyperparameter selection strategy in Learning with Label Proportions**. Advisor: Giovanni Ventorim Comarela.

In this work, we study the hyperparameter selection in the Learning with Label Proportions (LLP) context. In LLP, the data is provided in bags and only the label proportion of each bag is known. The LLP problem was first formalized. Two definitions were introduced: the first one is consistent with the scenarios studied in literature and the second one covers scenarios that are not included in the first definition. Then, a generic strategy for hyperparameter selection in LLP was defined. This strategy can be divided into two parts: split the data into training and validation sets and compute the model error in the validation set given a hyperparameter combination. We proposed three algorithms that divide the data per bag to address the first part. For the second part, a new error function that gives weights based on Fisher information for bags was introduced. Thereafter, the evaluation scenario for these methods was presented, including the new synthetic datasets which include some LLP practical aspects. The results showed that using the proposed methods for hyperparameter selection in LLP general scenarios improves the performance when compared with the state-of-art method.

Keywords: Learning with Label Proportions. Hyperparameter selection. Semi-supervised learning.

Lista de ilustrações

Lista de tabelas

Sumário

1 Introdução

Aprendizado de máquina é o estudo de algoritmos de computador que aprendem automaticamente através de experiencia através do uso de dados (MITCHELL, 1997). Algoritmos de aprendizado de máquina são utilizados em diversas aplicações, que vão desde a detecção de objetos em imagens (SZEGEDY; TOSHEV; ERHAN, 2013) até o reconhecimento de *e-mails* que são *spam* (METSIS; ANDROUTSOPOULOS; PALIOURAS, 2006).

Dentro da área de aprendizado de máquina, existem três grandes categorias de problemas de aprendizado, que são: aprendizado supervisionado, não supervisionado e por reforço. A seguir, essas três categorias serão brevemente apresentadas baseadas nas definições de RUSSELL; NORVIG (2009).

No aprendizado não supervisionado, a entrada é fornecida sem nenhum tipo de supervisão, sendo que o objetivo é aprender algum padrão importante dos dados. No aprendizado supervisionado a entrada é fornecida em pares de instância e saída, sendo o objetivo aprender uma função que mapeia uma instância para uma saída. Já no aprendizado por reforço, existe um agente que aprende o objetivo a partir de reforços, chamados de recompensas e penalidades.

Além dessas três categorias, existe uma categoria intermediária entre o aprendizado supervisionado e o aprendizado não supervisionado chamada de aprendizado semi-supervisionado. Muitas vezes o processo de obter rótulos é oneroso. Também existem aplicações nas quais não é possível obter os rótulos, mas existe alguma informação relacionada as instâncias ou a grupos de instâncias. O objetivo do aprendizado semi-supervisionado é, geralmente, aprender uma função que mapeia uma instância para uma saída utilizando menos informação em relação ao aprendizado supervisionado.

Um dos problemas de aprendizado semi-supervisionado que vem chamando bastante atenção da comunidade de aprendizado de máquina e que será abordado nesse trabalho é o Aprendizado com Proporções de Rótulos. A seguir, o problema de Aprendizado com Proporções de Rótulos será apresentado, sendo sua importância motivada e justificada. Então, a hipótese deste trabalho será apresentada, bem como seus objetivos. Por fim, a forma como o restante do trabalho está organizado será apresentada.

1.1 O problema e sua importância

Segundo YU et al. (2013), Aprendizado com Proporções de Rótulos (do inglês *Learning with Label Proportions*, ou LLP) é um problema de aprendizado semi-supervisionado

em que as instâncias de treino são disponibilizadas em *bags*. Para cada *bag*, é conhecida apenas as proporções dos rótulos. A tarefa é aprender uma função que mapeia instâncias em rótulos, assim como no problema de classificação.

Existem inúmeras aplicações práticas interessantes desse problema. Prever o resultado das eleições dos EUA é uma delas, cenário explorado por COMARELA et al. (2018). Os estados americanos podem ser vistos como *bags*, e para cada estado, se conhece a proporção de votos entre republicanos e democratas utilizando pesquisas eleitorais. Outra aplicação de LLP, apresentada por ARDEHALY; CULOTTA (2016), é prever o gênero e a raça de um usuário do *Twitter* baseados em seus *tweets* e sua imagem de perfil.

De acordo com YU et al. (2014), muitas informações são disponibilizadas na forma de grupos com proporções. Além do exemplo das eleições, existem informações de diagnósticos de doenças por CEP, de histórico de compras por região e de preferência em algum assunto específico (e.g. se as pessoas concordam com a política externa do país).

Além disso, o problema de LLP também engloba questões de privacidade (YU et al., 2014). Muitas vezes, os dados utilizados para treinar modelos são extremamente sensíveis, e a disponibilização desses dados podem causar grande dano aos envolvidos. Um exemplo disso é o diagnóstico de doenças, como por exemplo o câncer. Para não comprometer dados sensíveis dos indivíduos, esses dados podem ser transformados em dados de LLP antes de serem utilizados para aprendizado.

Um aspecto importante do processo de treinamento de modelos de aprendizado é a seleção de hiperparâmetros, que é o processo que encontra a melhor combinação de hiperparâmetros para um modelo de aprendizado treinado utilizando os dados disponíveis. Nem sempre as soluções para esse problema são fáceis, e se tratando de algoritmos LLP, esse problema se torna ainda mais complicado, devido a estrutura dos dados, que são divididos em *bags*, e a falta de informação de rótulos, que dificulta o processo de medir o erro de um certo modelo.

Observando esse cenário, este trabalho propõe uma estratégia de seleção de hiperparâmetros para algoritmos de Aprendizado com Proporções de Rótulos, levando em consideração vários aspectos práticos do problema.

1.2 Justificativa

Não existe uma forma bem definida de selecionar hiperparâmetros em no contexto de Aprendizado com Proporções de Rótulos. Esse problema é abordado parcialmente por HERNÁNDEZ-GONZÁLEZ (2019), que propõe dois métodos de validação cruzada para LLP. Já no trabalho de PATRINI et al. (2014), a seleção de hiperparâmetros foi feita utilizando os rótulos reais do conjunto de validação, mostrando a falta de uma abordagem bem definida para esse problema.

A seleção de hiperparâmetros em LLP, levando em consideração aspectos práticos do problema, ainda não foi abordada de forma profunda na literatura, apesar de ser um aspecto importante do processo de treinamento de modelos de aprendizado.

Dada a importância da seleção de hiperparâmetros no processo de treinamento e relevância que o problema de Aprendizado com Proporções de Rótulos possui, este trabalho se torna relevante à medida que propõe uma metodologia de seleção de hiperparâmetros em LLP.

1.3 Hipótese

No contexto de Aprendizado com Proporções de Rótulos, métodos de seleção de hiperparâmetros que levam em conta aspectos práticos do problema obterão melhor desempenho em relação aos métodos da estado da arte da literatura.

1.4 Objetivos

O objetivo deste trabalho é apresentar uma estratégia de seleção de hiperparâmetros para algoritmos de Aprendizado com Proporções de Rótulos que funcione bem nos diversos cenários práticos do problema. Além do objetivo principal este trabalho propõe como objetivos específicos:

- Formalizar o problema de Aprendizado com Proporções de Rótulos na forma que o problema é abordado na literatura e também levando em conta aspectos que fogem dessa definição padrão.
- Propor formas de dividir os dados entre conjuntos de treino e validação para serem utilizados no processo de seleção de hiperparâmetros em LLP.
- Criar uma forma de medir o erro de validação de um modelo de LLP no processo de seleção de hiperparâmetros.
- Construir conjuntos de dados de LLP que possuam propriedades interessantes afim de avaliar os métodos propostos.

1.5 Organização do trabalho

O restante deste trabalho está organizado da seguinte forma:

- Capítulo 2: apresenta os fundamentos teóricos de aprendizagem, provê uma breve descrição do problema de Aprendizado com Proporções de Rótulos e mostra uma série de trabalhos relacionados com esse problema.

- Capítulo 3: formaliza o problema de Aprendizado com Proporções de Rótulos na forma que é comumente abordado na literatura. Além disso, motiva e formaliza uma definição formal de Aprendizado com Proporções de Rótulos que foge da definição utilizada na literatura.

- Capítulo 4: propõe uma nova metodologia de seleção de hiperparâmetros para os algoritmos de Aprendizado com Proporções de Rótulos. Novas formas de dividir os dados entre conjuntos de treino e validação e de medir o erro de validação de um modelo de LLP são apresentados.

- Capítulo 5: apresenta todo o arcabouço experimental do trabalho, com os conjuntos de dados e a metodologia experimental utilizada. Os resultados obtidos são apresentados, mostrando que os métodos propostos proveem uma melhoria significativa em cenários que fogem da definição padrão de LLP da literatura.

- Capítulo 6: apresenta as considerações finais do trabalho, além de enumerar possíveis trabalhos futuros.

2 Referencial Teórico

Neste capítulo será apresentada a fundamentação teórica de casos clássicos de aprendizado. Então, o problema de Aprendizado com Proporções de Rótulos será introduzido. Por fim, trabalhos da literatura que abordam o problema de Aprendizado com Proporções de Rótulos serão mostrados divididos em três categorias: métodos, resultados teóricos e avaliação e validação.

2.1 Casos Clássicos de Aprendizado

MITCHELL (1997, p.2) define aprendizado de máquina da seguinte forma: "Diz-se que um programa de computador aprende pela experiência E, com respeito a algum tipo de tarefa T e performance P, se sua performance P nas tarefas em T, na forma medida por P, melhoram com a experiência E.". Os tipos mais clássicos de aprendizado são o aprendizado supervisionado e o não supervisionado, os quais serão detalhados a seguir, e o aprendizado por reforço, que não será detalhado pois não está no escopo desse trabalho.

Segundo RUSSELL; NORVIG (2009), no aprendizado não supervisionado padrões são aprendidos nos dados sem o fornecimento de informação explícita. A tarefa mais comum desse tipo de aprendizado é o agrupamento, que tem como objetivo detectar grupos nos dados fornecidos como entrada.

No aprendizado supervisionado são utilizados exemplos de pares de entrada e saída para aprender uma função que mapeia as entradas para a saída (RUSSELL; NORVIG, 2009). Os problemas de aprendizado podem ser divididos em dois tipos dependendo do tipo de saída fornecida. Os problemas de classificação possuem saída discreta e finita, e os de regressão possuem saída contínua.

Também existe o aprendizado semi-supervisionado, que se situa entre o aprendizado supervisionado e o não supervisionado, onde estão disponíveis apenas alguns dados rotulados e uma grande quantidade de dados não rotulados (RUSSELL; NORVIG, 2009). O problema abordado nesse trabalho, chamado Aprendizado com Proporções de Rótulos, é um exemplo de problema de aprendizado semi-supervisionado. Esse problema se assemelha muito a classificação, mas utilizando informações mais fracas que os rótulos para classificar. Vários algoritmos da literatura de LLP se baseiam em algoritmos de classificação. Por isso será dado foco ao problema de classificação no decorrer dessa seção.

2.1.1 Problema de Classificação

Nesta seção o problema de classificação binária será formalizado. O cenário considerado por todo o trabalho é o binário, i.e., existem apenas dois rótulos possíveis, também chamados de rótulos positivos e negativos. Toda notação e formulação está baseada nos trabalhos de MOHRI; ROSTAMIZADEH; TALWALKAR (2018) e de MITCHELL (1997).

O conjunto X denota o conjunto de todas as instâncias possíveis. Já o conjunto de todos os rótulos possíveis é dado por Y. Como o problema é de classificação binária, $Y = \{0, 1\}$. Um conceito c é um mapeamento de X para Y, definido por $X \to Y$. Um exemplo de conceito pode ser "pessoas que ganham um salário anual maior do que \$50.00". Se uma instância x é um exemplo positivo de c, então $c(x) = 1$, caso contrário $c(x) = 0$. O conjunto de todos os conceitos que são podem ser aprendidos é chamado de classe de conceitos, denotada por C. Existe a presunção de que as instâncias são independentes e identicamente distribuídas (*i.i.d.*) de acordo com uma distribuição de probabilidade $\mathcal{D}$. O aprendiz L considera um conjunto de possíveis hipóteses dado por $\mathcal{H}$ na tentativa de aprender um conceito c. L recebe as instâncias de entrada e seus respectivos rótulos, dados pelo conceito c, e tem como tarefa selecionar a hipótese $h \in \mathcal{H}$ que minimiza o erro de generalização referente ao conceito c.

De acordo com MITCHELL (1997), dado uma hipótese $h \in \mathcal{H}$, um conceito $c \in C$ e uma distribuição de probabilidade $\mathcal{D}$, o erro de generalização de h pode ser definido como:

$$Error_{\mathcal{D}}(h) = \mathbb{P}_{x \sim \mathcal{D}}[h(x) \neq c(x)]$$

Existem vários algoritmos de LLP que utilizam ou se baseiam em algoritmos de classificação. A próxima seção detalhará um deles, chamado SVM, que é bastante utilizado por este trabalho.

2.1.2 Máquina de Vetores de Suporte

A Máquina de Vetores de Suporte (do inglês *support vector machine*, ou SVM) é um classificador bastante conhecido na literatura por conseguir um bom desempenho em uma grande gama de problemas, inclusive problemas não linearmente separáveis.

A ideia do SVM linear é encontrar o hiperplano que separa os dados e que possui a maior margem, sendo a margem definida como a menor distância entre o hiperplano de separação e os pontos (ZAKI; MEIRA JR, 2014). Os pontos que definem a localização da margem são chamados de vetores de suporte. Em problemas não linearmente separáveis, os vetores de suporte podem estar dentro da região entre a margem e o hiperplano de separação. A Figura 1 mostra o resultado de uma aplicação do SVM em dados linearmente separáveis.

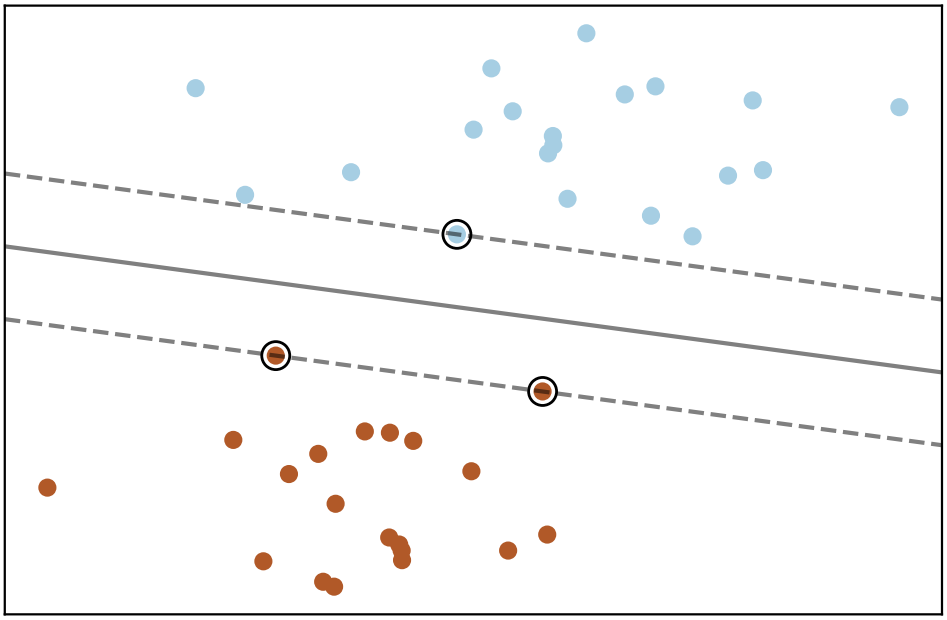


Figura 1 – Resultado da aplicação do SVM em dados linearmente separáveis. As margens são as linhas tracejadas, e os vetores de suporte são os pontos circulados. **Fonte**: Scikit-Learn. Disponível em: <https://scikit-learn.org/stable/auto_examples/svm/plot_separating_hyperplane.html>

Para resolver problemas não lineares, o SVM aplica o truque do kernel dos dados. Segundo ZAKI; MEIRA JR (2014), a ideia é mapear os pontos $\mathbf{x}_i$ d-dimensionais no espaço de entrada para pontos $\phi(\mathbf{x}_i)$ no espaço dos atributos, que possui uma alta dimensionalidade, utilizando uma transformação não linear ϕ. Com isso, espera-se que os pontos $\phi(\mathbf{x}_i)$ sejam mais fáceis de se separar. O truque do *kernel* possibilita a realização de todas as operações utilizando a função *kernel* no espaço de entrada, não sendo necessário realizar efetivamente o mapeamento dos pontos para o espaço dos atributos.

Seja $\mathbf{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ o conjunto de entradas, com n pontos no espaço d-dimensional. Considere o caso de classificação binário, i.e., $y_i \in \{-1, 1\}$. Segundo ZAKI; MEIRA JR (2014), o problema de otimização do SVM pode ser definido como:

$$\min_{\mathbf{w},b,\zeta} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{n}\zeta_i,$$

s.t. $y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \zeta_i$ e $\zeta_i \geq 0,\ i = 1, \ldots, n.$

Essa equação tenta maximizar a margem ao mesmo tempo que penaliza uma instância classificada incorretamente ou que está dentro da margem. O termo ζ_i é utilizado para flexibilizar a restrição de separabilidade, pois nem sempre os dados são totalmente

separáveis por um hiperplano. O termo C controla o tamanho dessa penalidade.

Escolher os melhores hiperparâmetros de um certo algoritmo, como o parâmetro C do SVM, é uma das tarefas mais importantes do processo de treinamento de modelos de aprendizado de máquina. Na próxima seção, o problema de seleção de hiperparâmetros será introduzido.

2.1.3 Seleção de Hiperparâmetros

Geralmente os algoritmos de classificação possuem hiperparâmetros que precisam ser otimizados para a solução de um problema, como por exemplo o parâmetro C do SVM. A forma mais comum de se fazer isso é, primeiro, particionando o conjunto de dados em conjuntos de treino e teste. Então, para cada combinação de hiperparâmetros que será testada, um processo de validação cruzada é feito utilizando o conjunto de dados de treino. Esse processo fornece uma estimativa do erro de predição do modelo para cada combinação de hiperparâmetros. Após todas as combinações de hiperparâmetros serem testadas, o modelo com o menor erro de predição é selecionado. Por fim, este modelo pode ser avaliado no conjunto de teste, que fornece uma melhor estimativa do desempenho desse modelo em dados nunca vistos.

O processo de validação cruzada consiste em dividir os dados em conjuntos de treino e validação. Então, o modelo é treinado no conjunto de treino e avaliado no conjunto de validação. Esse processo pode ser repetido várias vezes, sendo que a estimativa do erro de predição do modelo é dado pela combinação dos erros calculados em cada iteração. Existem várias maneiras de fazer validação cruzada, e uma das mais conhecidas é o k-fold.

Segundo HASTIE; TIBSHIRANI; FRIEDMAN (2009), o k-fold funciona da seguinte forma: os dados são divididos em K partes iguais. Para a k-ésima parte, o modelo é treinado com as outras $K - 1$ partes e o erro de predição desse modelo é calculado utilizando a k-ésima parte. Isso é repetido para $k = 1, 2, \ldots, K$. O erro final é a combinação do erro dos K modelos treinados (e.g. utilizando a média do erro dos K modelos). A Figura 2 mostra em detalhes o funcionamento do k-fold.

Após essa introdução aos casos de aprendizado clássicos, será introduzido na próxima seção o problema principal abordado por este trabalho, chamado Aprendizado com Proporções de Rótulos.

2.2 Aprendizado com Proporções de Rótulos

Segundo YU et al. (2013), Aprendizado com Proporções de Rótulos (LLP) é um problema de aprendizado semi-supervisionado em que as instâncias de treino são disponibilizadas em *bags*. Para cada *bag*, é conhecida apenas as proporções dos rótulos. A tarefa é aprender uma função que mapeia instâncias em rótulos, assim como no problema de

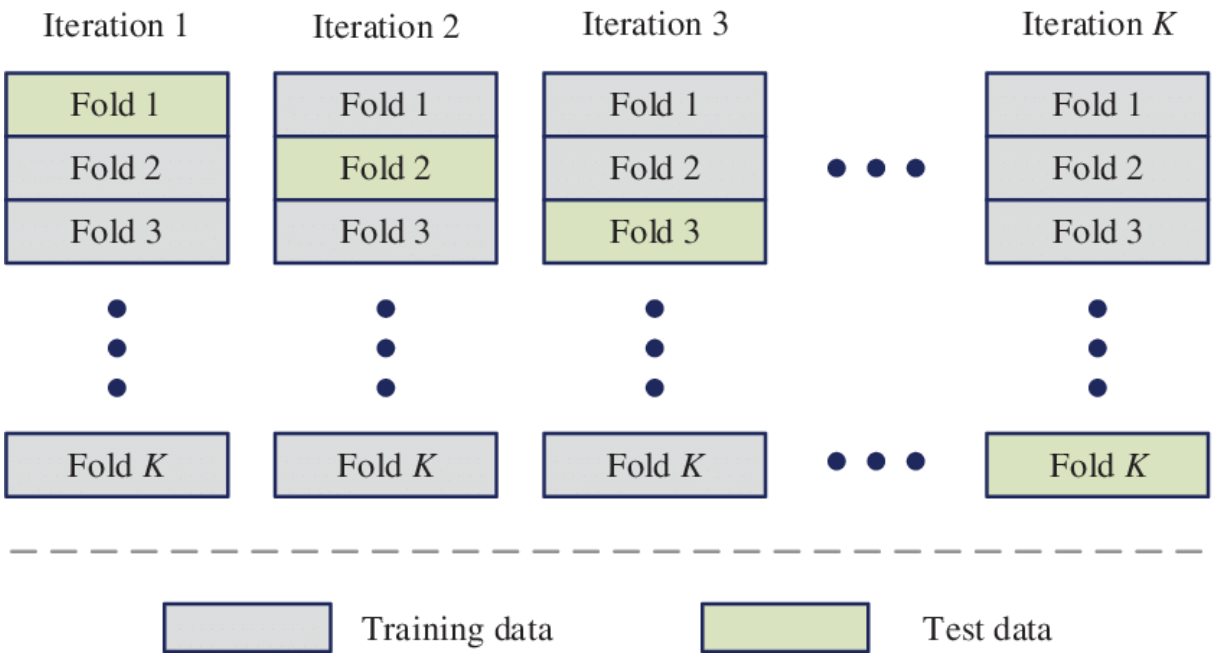


Figura 2 – Funcionamento do k-fold **Fonte**: REN; LI; HAN (2019)

classificação. A Figura 3 mostra uma comparação entre tarefas de aprendizado clássicas e LLP.

Um exemplo simples de LLP é mostrado na Figura 4. O foco deste trabalho será em problemas binários, portanto, a proporção de uma *bag* será dada pela proporção dos rótulos positivos dessa *bag*. Além disso, as proporções estão sempre no intervalo $[0, 1]$. Nesse exemplo, só existe uma *bag*, com proporção de $\frac{1}{3}$, *i.e.*, $\frac{1}{3}$ dos pontos dessa *bag* possuem rótulos positivos e $\frac{2}{3}$ possuem rótulos negativos (ver Figura 4a). O objetivo do problema é encontrar uma função que mapeia instâncias em rótulos, respeitando ao máximo as proporções, como mostrado na Figura 4b.

Existem inúmeras aplicações práticas para o problema de Aprendizado com Proporções de Rótulos. Em muitas situações práticas, as informações só existem na forma de grupos com as proporções de rótulo. Já em outros casos, as informações individuais existem, mas por questões de privacidade elas são omitidas e disponibilizadas na forma de LLP. A eleição dos EUA são um bom exemplo prático de LLP, sendo explorada por COMARELA et al. (2018). Neste trabalho, foram utilizados os históricos de navegação na web de usuários para prever a preferência de candidato nas eleições presidenciais norte americanas de 2018. Nesse caso, as pesquisas eleitorais forneceram a proporção de republicanos e democratas por estado. Utilizando essa técnica, foi possível extrair a preferência de candidato por dia e em nível estadual utilizando dados da Internet, além de entender o impacto de evento "*Comey Letter*" nas eleições presidenciais dos EUA de 2018. Outro exemplo de LLP na prática, mostrado por YU et al. (2014), é diagnóstico de doenças por CEP. Neste caso, dados sobre a incidência de certas doenças por CEP são disponibilizados publicamente. Os dados sobre as doenças de cada pessoa são extremamente sensíveis, por

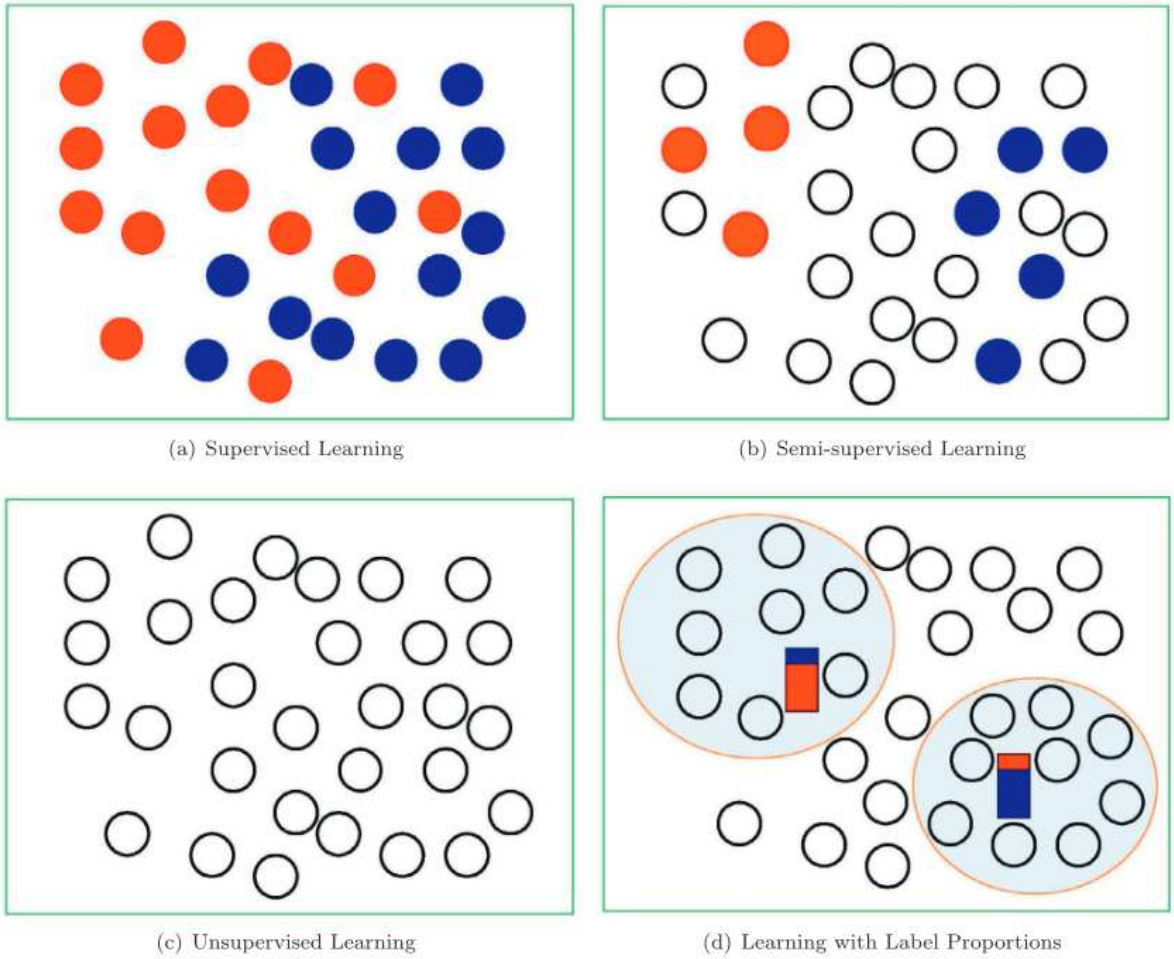
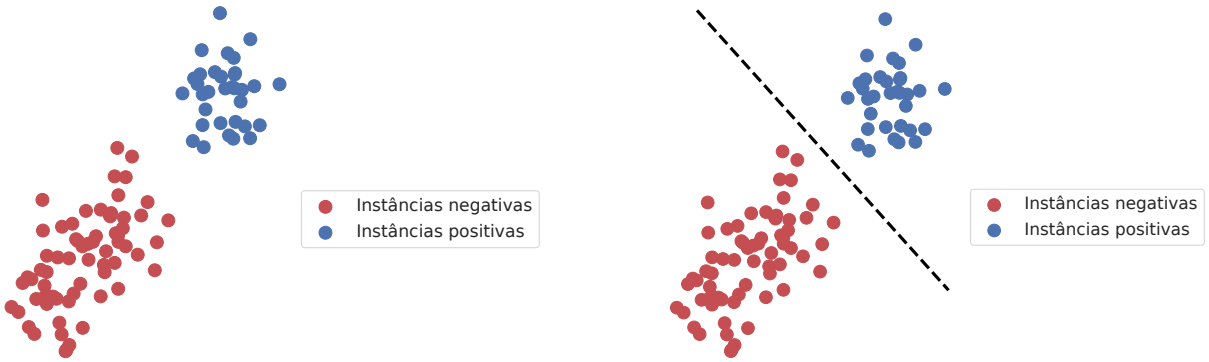


Figura 3 – Comparação entre aprendizado supervisionado, não supervisionado, semi-supervisionado e LLP. **Fonte**: CHEN et al. (2017)

isso eles são fornecidos dessa forma.

Apesar das inúmeras aplicações e relevância do problema, existe uma grande dificuldade em resolvê-lo. Em geral, o conjunto de hipóteses que satisfazem as restrições de proporções pode ser muito grande, o que aumenta a complexidade para resolver problemas de LLP. O cenário hipotético mostrado na Figura 5 será utilizado para ilustrar essa dificuldade. Neste cenário, existem duas *bags*, cada uma com 50 elementos e com proporção de $\frac{1}{2}$. Considerando um modelo de aprendizado supervisionado, uma simples reta separa os elementos positivos (em azul) dos negativos (em vermelho). Mas se tratando de um modelo de LLP, existem duas opções de classificação. A primeira é classificar todos os elementos abaixo da reta como negativo e acima como positivo. Neste caso, as proporções das *bags* serão respeitadas, i.e., a previsão do modelo para essas *bags* também possui metade de rótulos positivos para cada uma delas, e todos os rótulos são previstos corretamente. A segunda é classificar de forma oposta a primeira, ou seja, os elementos abaixo da reta são classificados como positivos e acima como negativos. As proporções das *bags* também são respeitadas nesse caso, mas todos os rótulos são previstos incorretamente. Neste caso, existem duas soluções que respeitam as proporções, mas apenas



(a) Bag com proporção de $\frac{1}{3}$ (b) Objetivo de LLP

Figura 4 – Exemplo de Aprendizado com Proporções de Rótulos com uma *bag*. (a) Dados da *bag*. (b) Hiperplano de separação dos dados que respeita as proporções.

uma que classifica os rótulos individuais corretamente. Como os rótulos reais não estão disponíveis, não é possível diferenciar as duas soluções. Este é um caso extremo que pode acontecer em LLP, mas mostra a dificuldade em resolver o problema.

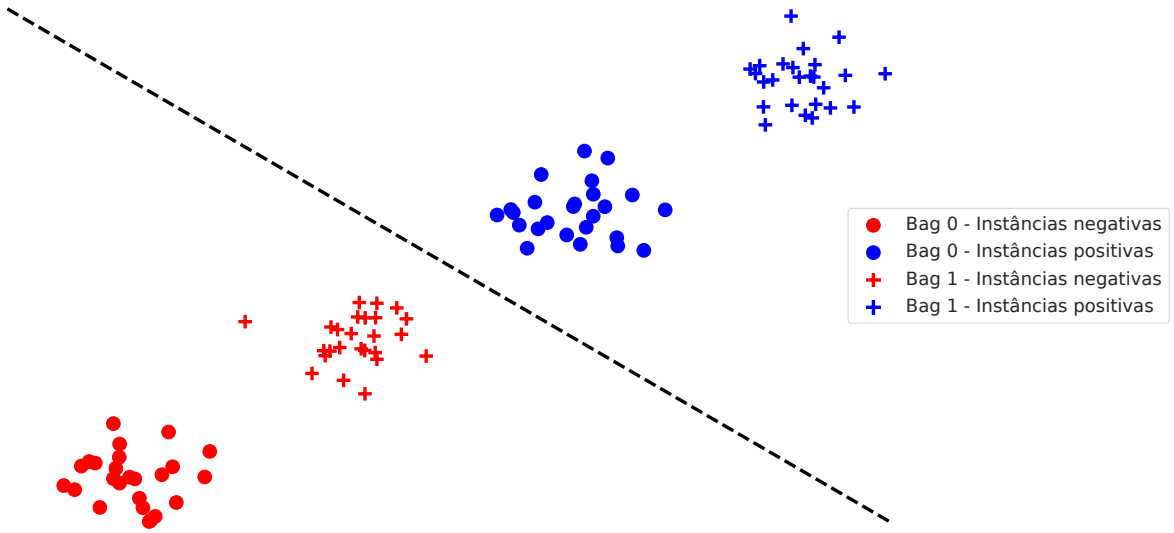


Figura 5 – Cenário hipotético de Aprendizado com Proporções de Rótulos.

2.3 Abordagens existentes

O problema de Aprendizado com Proporções de Rótulos vem ganhando bastante relevância na comunidade de aprendizado de máquina desde que foi introduzido por KÜCK; FREITAS (2005). Além de introduzir pela primeira vez LLP e algumas aplicações práticas, este trabalho propôs um modelo probabilístico para resolver problemas de LLP no caso binário. A probabilidade condicional que o modelo visa aprender é dada por $Pr(y = 1|\mathbf{x}, \mathcal{D})$, onde $\mathbf{x}$ é uma instância, y um rótulo binário e $\mathcal{D}$ o conjunto de dados de treinamento. O trabalho utiliza o algoritmo de Monte Carlo via Cadeias de Markov

(MCMC) para aproximar o valor dessa probabilidade. Esse algoritmo foi testado em dados sintéticos e em um conjunto de dados de reconhecimento de imagens.

Desde então, muitos trabalhos em LLP foram desenvolvidos, os quais serão apresentados nesta seção. O resto da seção será organizada da seguinte maneira: serão apresentados primeiro os métodos para resolver LLP, depois os resultados teóricos e finalmente os trabalhos sobre avaliação e validação de modelos de LLP.

2.3.1 Métodos

Afim de apresentar de forma mais organizada os métodos para resolver problemas de LLP da literatura, esta subseção mostrará os métodos divididos em três grupos: baseados no SVM, baseados em redes neurais artificiais e outros métodos, que não necessariamente possuem relação entre si.

2.3.1.1 Métodos Baseados no SVM

RUEPING (2010) propôs o *InvCal*, um algoritmo que resolve o problema de LLP binário importante na literatura e o primeiro baseado no SVM. A ideia geral do algoritmo é utilizar calibração inversa (do inglês *Inverse Calibration*) para, dada as probabilidades, que neste caso são as proporções de cada *bag*, treinar um SVM que prevê os rótulos individuais. Para conseguir fazer isso, o trabalho trata a média de cada *bag* como uma instância, e rótulo dessa instância é a proporção dessa *bag*. Os resultados em diferentes conjuntos de dados mostraram uma boa efetividade dessa estratégia. Foi mostrando também a dependência dos resultados em relação ao tamanho das *bags* e ao *kernel* utilizado no algoritmo.

O principal algoritmo de LLP baseado no SVM da literatura foi proposto por YU et al. (2013), o qual realizou a seguinte modificação na formulação do algoritmo supervisionado para resolver o problema:

$$\min_{\mathbf{w},b,\mathbf{y}} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{N} L(y_i, \mathbf{w}^T\varphi(\mathbf{X}_i) + b) + C_p\sum_{k=1}^{K} L_p(\hat{p_k}, p_k), \tag{2.1}$$

onde C e C_p são hiper-parâmetros, a função $L(.)$ é a função de perda do aprendizado supervisionado e a função $L_p(.)$ é a função de perda proposta neste trabalho para penalizar soluções que não respeitam as proporções das *bags*. A função de perda *hinge*, dada por $max(0, 1 - y_i(\mathbf{w}^T\varphi(\mathbf{X}_i) + b))$, foi utilizada em $L(.)$. Já para $L_p(.)$, a função de perda utilizada pelo alter-SVM é dada por:

$$L_p = \sum_{i=i}^{K} |p_i - \hat{p}_i|, \tag{2.2}$$

onde K é o número de *bags* e $\hat{p}_i$ é a proporção predita na i-ésima *bag*.

A tarefa de otimização é, ao mesmo tempo, encontrar os parâmetros do modelo $\mathbf{w}$ e b e os rótulos $\mathbf{y}$ que minimizem a Equação 2.1.

Foram propostos dois algoritmos para resolver esse problema de otimização: alter-$\propto$SVM e conv-$\propto$SVM. Nesta dissertação, o alter-$\propto$SVM será utilizado como base para as comparações realizadas, seguindo o que é feito na literatura. Por esse motivo, este algoritmo será apresentado de forma mais detalhada. O Algoritmo 1 mostra o pseudocódigo do alter-$\propto$SVM.

Algoritmo 1: alter-$\propto$SVM

```
Inicialize aleatoriamente y_i ∈ {−1, 1}, ∀_{i=1}^{N}. C* = 10^{−5}C
enquanto C* < C faça
    C* = min{(1 + Δ)C*, C}
    repita
        Fixe y para resolver w e b
        Fixe w e b para resolver y
    até a redução do objetivo ser menor que um limiar (10^{−4});
```

O algoritmo começa inicializando aleatoriamente o vetor de rótulos $\mathbf{y}$. Então, é utilizada a ideia de otimização alternada em dois passos. O primeiro passo é fixar o termo $\mathbf{y}$ da Equação 2.1 e resolver um SVM. Com isso, os parâmetros do modelo $\mathbf{w}$ e b são utilizados para encontrar o $\mathbf{y}$. Para evitar mínimos locais, a parte de otimização alternada está dentro de um laço que utiliza a técnica de *annealing* no parâmetro C. O valor de $\Delta = 0.5$ é utilizado como padrão pelo algoritmo.

Já o segundo algoritmo apresentado neste trabalho, chamado conv-$\propto$SVM, faz uma relaxação convexa na Equação 2.1 e a resolve utilizando técnicas de *Multiple Kernel Learning* (MLK). Os experimentos foram feitos em um dados sintéticos e reais, e os resultados mostram que ambos alter-$\propto$SVM e conv-$\propto$SVM superam os outros algoritmos nos cenários testados.

Vários trabalhos propondo modificações no alter-$\propto$SVM foram propostos desde então, o que mostra a relevância desse algoritmo. QI et al. (2016) propôs um método baseado em SVM não paralelo junto com agrupamento k-plane chamado LLP-NPSVM. A ideia geral desse método é construir dois hiperplanos não paralelos para recuperar informações sobre a distribuição dos dados, e assim utilizar erros de proporção de *bag* para ajustar os rótulos individuais. Esses dois passos podem ser vistos como um algoritmo de *expectation-maximization* (EM) que utiliza um laço com *annealing* para evitar mínimos locais. É mostrado que o LLP-NPSVM converge rapidamente e tem boa acurácia quando comparado com outros métodos em vários cenários. CHEN et al. (2017) propôs uma formulação similar, mas utilizando um SVM não paralelo. O objetivo é encontrar dois hiperplanos não paralelos que dividem os dados respeitando as proporções. Foi mostrado que esse algoritmo se comporta bem em alguns cenários práticos. SHI et al. (2018)

utilizou uma floresta aleatória na estrutura do alter-∝SVM. Esse algoritmo possui vantagem principalmente em dados com alta dimensionalidade, já que florestas aleatórias são robustas em dados com várias dimensões. Os resultados mostram vantagem exatamente nesses cenários. SHI et al. (2019) propôs o uso da função de perda *pinball* no problema de otimização do alter-∝SVM, mostrando que essa função melhora o desempenho em alguns cenários práticos. Dois métodos que utilizam *boosting* e *bagging* juntamente com o alter-∝SVM foram propostos por CHEN; CHEN; SHI (2020), com o foco em cenários de predição de falência. Nesses cenários, esses dois métodos obtiveram melhor desempenho se comparados a três algoritmos: InvCal, alter-∝SVM e LLP-NPSVM.

2.3.1.2 Métodos Baseados em Redes Neurais Artificiais

Vários métodos baseados em redes neurais artificiais e aprendizado profundo também foram propostos para resolver problemas de LLP.

Redes neurais artificiais e aprendizado profundo são duas tendências em aprendizado supervisionado. Alguns trabalhos da literatura se basearam nessas técnicas para resolver problemas de LLP. ARDEHALY; CULOTTA (2017) propôs uma camada chamada *Batch Averager* que possibilita utilizar arquiteturas de redes neurais para aprender problemas de LLP. Para isso, cada instância têm como rótulo a proporção de sua respectiva *bag*. No treinamento da rede, cada *batch* é correspondente a uma *bag*. Quando a *bag* é muito grande, ela é dividida em alguns *batches* menores. Para cada *batch*, a camada de *Batch Averager* tira a média das probabilidades preditas pela rede neural e calcula o erro em relação a proporção da *bag* daquele *batch*. Essa camada só é aplicada no treinamento da rede, sendo removida da rede depois de treinada. Além disso, esse trabalho apresenta uma forma de fazer co-treinamento de redes neurais pra LLP, possibilitando o uso de duas visões condicionalmente independente dos dados (e.g. texto e imagem de um certo conjunto de dados) para treinar a rede neural.

SHI; LIU; QI (2018) propôs uma rede neural convolucional para LLP. Nessa arquitetura, a última camada da rede é um valor, que é utilizado para fazer uma estimativa dos valores de y, utilizando a média dos valores de cada *bag*. Assim, é possível calcular o erro entre os rótulos preditos e os rótulos estimados, possibilitando a utilização dos algoritmos usuais para treinamento de redes neurais supervisionadas, sendo que neste caso, foi utilizado o algoritmo ADAM. A arquitetura foi testada no conjunto de dados MNIST, e os resultados mostraram que esse arquitetura supera os outros algoritmos nesse contexto.

Já LIU et al. (2019) propôs um método que utiliza *Generative Adversarial Networks* (GANs) chamado LLP-GAN. Esse algoritmo é treinado como uma GAN supervisionada, mas assim como em outros trabalhos, a proporção de cada *bag* é utilizada como probabilidade a priori para as instâncias, representando um rótulo real. Além disso, o algoritmo possui várias propriedades teóricas parecidas com as de GANs. Os resultados mostram

que o LLP-GAN supera o estado da arte nos conjuntos de dados de imagem CIFAR-10, CIFAR-100, SVHN e MNIST.

2.3.1.3 Outros métodos

QUADRIANTO et al. (2009) introduziu um algoritmo chamado *MapMean*. Este trabalho se baseia em estimar operador de média utilizando as proporções ao invés dos rótulos reais, achar o melhor θ resolvendo um problema de otimização e utilizar esse valor para a estimar $p(y|x,\theta)$, que é utilizado para estimar o rótulo de x. O trabalho argumenta que a estimativa do operador de média é um bom *proxy* para o valor real devido a esta quantidade ser bem comportada e convergir sob poucas condições para seu valor esperado. Esse algoritmo assume que as *bags* e os dados são condicionalmente independentes dado os rótulos. O algoritmo foi testado em vários conjuntos de dados do UCI e do LibSVM.

PATRINI et al. (2014) propôs um método baseado no *MapMean* chamado *Laplacian Mean Map* (LMM) que estima o operador de média utilizando Regularização Manifold Laplaciana, mas sem as condições restritas do *MapMean*. Este trabalho também introduziu o *Alternating Mean Map* (AMM), que basicamente faz uma otimização em cima da solução do LMM. Os experimentos foram realizados em uma gama grande de domínios, utilizando vários conjuntos de dados supervisionados e várias formas de criar as *bags*. O trabalho apresentou versões lineares dos algoritmos, mas afirmaram que é possível utilizar funções *Kernel* nesses algoritmos para casos não linearmente separáveis.

Um algoritmo pra LLP baseado em Adaboost chamado Adaboost-LLP foi apresentado por QI et al. (2017). A ideia geral desse algoritmo é treinar vários classificadores lineares fracos e utilizar uma função logística para estimar a probabilidade de uma instância ser de uma classe, e então, é possível construir uma função de custo a ser otimizada por algum algoritmo. Além disso, esse trabalho apresentou uma função de perda de proporções que atribuía pesos para as *bags* de acordo com o tamanho e as proporções. *Bags* com proporções próximas de $\frac{1}{2}$ são mais difíceis de resolver, então o peso delas é menor. Além disso, *bags* pequenas contribuem menos para o erro, então o peso delas também é menor. As equações 2.3, 2.4, 2.5 e 2.6 mostram esse esquema de peso.

$$L_p(\hat{p_k}, p_k) = L_p^{MSE-weight}(\hat{p_k}, p_k) + L_p^{prior}(\hat{p_k}, p_k) \tag{2.3}$$

$$L_p^{MSE-weight}(\hat{p_k}, p_k) = w_k(p_k - \hat{p_k})^2 \tag{2.4}$$

$$w_k = \frac{|B_k|}{N} \frac{|p_k - 0.5| + \tau}{\sum_{i=1}^{K} |p_i - 0.5| + \tau}, \tag{2.5}$$

onde τ é uma constante.

$$L_p^{prior}(\hat{p_k}, p_k) = \left(\frac{|B_k|}{N} p_k - \frac{|B_k|}{N} \hat{p_k} \right)^2 \tag{2.6}$$

Um método baseado em propagação de rótulos (do inglês *Label Propagation*) foi proposto por POYIADZI; SANTOS-RODRIGUEZ; TWOMEY (2018). Esse algoritmo, chamado LP-LLP, utiliza a matriz similaridade entre as instâncias para extrair uma estrutura global dos dados. Após isso, utiliza uma otimização em dois passos: primeiro resolve um problema de propagação de rótulos, substituindo o rótulo pela proporção da *bag* a qual a instância pertence. Após isso, é utilizada a técnica de projeções alternadas (do inglês *Alternating Projections*) para determinar a solução que respeita as proporções. Esses dois passos são repetidos até a convergência. Os resultados mostram um ganho de desempenho em relação ao alter-$\propto$SVM e ao *InvCal*. Os autores também afirmam que esse método tende a dar menos superadaptação. Um método de adaptação de domínio foi introduzido por ARDEHALY; CULOTTA (2016). Nesse caso, as proporções do domínio de destino não eram conhecidas, então um modelo de LLP era treinado no domínio de origem utilizando *self-training* para gerar as proporções do domínio de destino. Foram utilizados dados do twitter, do IMDB e de fóruns de discussão para mostrar a efetividade do método.

COMARELA et al. (2018) desenvolveram um algoritmo de *expectation-maximization* utilizando regressão logística. A grande diferença desse algoritmo é usar um limiar de classificação variável para a regressão logística dependendo da *bag*. Se uma *bag* possuí uma proporção de 0.9, é mais provável que uma instância aleatória nesse estado seja positiva, e a variação do limiar de classificação da regressão logística consegue capturar essa ideia. A aplicação deste trabalho foi nas eleições americanas de 2016, como mostrado na Seção 2.2.

2.3.2 Resultados Teóricos

Também existem trabalhos sobre aspectos teóricos do Aprendizado com as Proporções de Rótulos. YU et al. (2014) mostram que, se as *bags* são independentes e identicamente distribuídas (ou *iid*), um modelo que prevê bem as proporções também prevê bem os rótulos. Esse trabalho também mostra um estudo de caso real utilizando dados do censo para prever renda, com algumas formas diferentes de gerar as *bags*. Já FISH; REYZIN (2017) utilizaram uma versão simplificada de LLP, com uma *bag* binária onde cada instância da mesma é *iid* e vem de uma distribuição arbitrária, para mostrar, além de outras coisas, que o que é aprendível em LLP é um pequeno subconjunto do que é aprendível em PAC. A extensão natural desse trabalho é mostrar essas propriedades pro caso geral, onde existem várias *bags* de tamanhos e proporções distintas.

2.3.3 Avaliação e Validação

HERNÁNDEZ-GONZÁLEZ (2019) introduziu uma nova estrutura para validação e avaliação de algoritmos de LLP. Para a parte de avaliação de algoritmos de LLP, foram propostas quatro formas distintas de estimar os verdadeiros positivos utilizando apenas as proporções. Considerando a notação utilizada pelo trabalho, dada uma *bag* i, m_{i+} é o número de instâncias positivas, $\hat{m}_{i+}$ é o número de instâncias preditas como positivas nesta *bag*, m_i é o número de instâncias. Além disso, considere K como o número de *bags* e p_i como a proporção de rótulos positivos da *bag* i. As estimativas dos verdadeiros positivos otimista (Equação 2.7), pessimista (Equação 2.8), aleatória (Equação 2.9) e ajustada (Equação 2.10) de uma *bag* i são dadas por:

$$\overline{TP}_i = min(m_{i+}, \hat{m}_{i+}) \tag{2.7}$$

$$\underline{TP}_i = max(0; \hat{m}_{i+} + m_{i+} - m_i) \tag{2.8}$$

$$\widetilde{TP}_i = \frac{\hat{m}_{i+} * m_{i+}}{m_i} \tag{2.9}$$

$$recall^* = \frac{\sum_i^K p_i * \overline{TP}_i + (1 - p_i) * \widetilde{TP}_i}{\sum_i^K m_{i+}}$$

$$TP_i^* = recall^* * m_i, \tag{2.10}$$

Utilizando uma dessas estimativas, é possível reconstruir a matriz de confusão, possibilitando o cálculo de todas as métricas usuais de avaliação de classificadores, como a revocação e a curva ROC. Foi avaliada a qualidade da aproximação da curva ROC para LLP em relação ao supervisionado utilizando classificadores supervisionados, e só utilizando as proporções na partição de validação. O problema dessa abordagem é que o algoritmo supervisionado nunca vai inverter os rótulos em casos como o da Figura 5, e assim, as métricas estimadas podem não ser robustas em relação a esses casos. Também foi mostrado que existem diferenças na função de densidade entre as métricas utilizando essas estimativas e a função de perda padrão de LLP.

A segunda parte desse trabalho propõe um k-fold e um k-fold estratificado para realizar validação cruzada em LLP. A ideia básica dos dois métodos é atribuir *bags* para partições de uma forma gulosa e probabilística. As Equações 2.11 e 2.12 mostram as

distribuições utilizadas pelo k-fold e k-fold estratificado, respectivamente.

$$i^* \sim Cat\left(\left\{\frac{m_i^2}{\sum_{i'=1}^{K} m_{i'}^2}\right\}_{i=1}^{K}\right) \tag{2.11}$$

$$i^* \sim Cat\left(\{c_i\}_{i=1}^{K}\right) \tag{2.12}$$

sendo

$$*c_i = \begin{cases} \frac{m_{i+}^2}{\sum_{i'=1}^{K} m_{i'+}^2} & \text{if } f_i > P \\ \frac{m_{i-}^2}{\sum_{i'=1}^{K} m_{i'-}^2} & \text{if } f_i < P \end{cases} \tag{2.13}$$

onde f_i é a proporção de rótulos positivos da partição i e P é a proporção de rótulos positivos de todo o conjunto de dados.

A ideia do k-fold é que a maior *bag* tenha a maior probabilidade de ser atribuída para a menor partição. Já a ideia do k-fold estratificado é criar partições que possuam a proporção de rótulos positivos similar a proporção globais de rótulos positivos. Os resultados mostram um ganho de desempenho dessas abordagens em relação a técnica *Leave-one-bag-out (LOBO)*, que consiste em utilizar uma *bag* para validação enquanto as outras são utilizadas para treino, e repetir esse procedimento K vezes, sendo utilizada uma *bag* diferente para validação em cada uma das vezes. O grande ponto fraco desses experimentos é que, como os anteriores, eles também foram feitos treinando classificadores supervisionados, o que deixa de fora casos difíceis para algoritmos de LLP (ver Figura 5).

2.4 Discussões

Existem alguns aspectos experimentais de LLP que não foram devida e exaustivamente abordados na literatura. Um deles são os conjuntos de dados utilizados para teste de algoritmos. Grande parte dos trabalhos da literatura utilizam conjuntos de dados de aprendizado supervisionado que são transformados em dados de LLP. Existem três formas mais comuns de se fazer isso. A primeira é criar *bags i.i.d.* de tamanho fixo, que é a abordagem mais utilizada. Neste caso, não existe relação direta entre as *bags* e os pares de instância e rótulo, o que não representa bem um cenário real de LLP. Além disso, *bags* de tamanho fixo não são usuais em cenários práticos. A segunda forma é utilizar um algoritmo de agrupamento (e.g. k-means) para criar as *bags*. Neste caso, existe grande chance das *bags* serem convexas e homogêneas, o que não acontece na prática. Já a terceira forma é selecionar um atributo para servir como *bag*, e depois remove-lo do conjunto de dados. O problema dessa abordagem é que não existe um critério claro para a seleção do atributo que servirá como *bag*. Seria necessário um estudo mais aprofundado da escolha

de atributo para gerar cenários práticos interessantes.

Outro problema que será abordado neste trabalho e que é pouco abordado na literatura é como quantificar a informação de uma *bags*. Em cenários reais, *bags* possuem tamanhos e proporções diferentes. Grande parte da literatura não aborda isso convincentemente nos experimentos, sendo que a grande maioria dos trabalhos utilizaram *bags* de tamanho fixo.

Mais um ponto fundamental sobre LLP pouco abordado na literatura é a escolha de hiperparâmetros. HERNÁNDEZ-GONZÁLEZ (2019) propôs duas formas de k-fold que utilizam partições com *bags* inteiras. Em cenários práticos de LLP, abordagens que utilizam partições com *bags* inteiras podem não funcionar bem. Utilizar *bags* inteiras pode não oferecer uma amostra boa o suficiente dos dados completos, podendo diminuir o desempenho dos algoritmos de LLP.

Na próxima seção, o problema de Aprendizado com Proporções de Rótulos, da forma em que é abordado na literatura, será formalmente definido. Além disso, os cenários de LLP que fogem das definições utilizadas na literatura serão formalmente definidos.

3 Formalização do Problema de Aprendizado com Proporções de Rótulos

Neste capítulo será definida a notação que irá ser utilizada no decorrer do trabalho. Para isto, o problema de classificação binária será redefinido utilizando esta notação. Após isso, esta notação será estendida para definir diferentes versões de LLP. Então, as motivações para os problemas que serão abordados por este trabalho serão apresentadas. A formalização de duas versões diferentes de LLP, uma seguindo a intuição da literatura e a outra mais geral, já é uma contribuição deste trabalho. Essa formalização de LLP é importante para entender as diferenças entre elas e quais são os impactos dessas diferenças em aplicações reais do problema.

3.1 Definição do Problema de Classificação Binária

Seja a entrada do problema de classificação binária um conjunto de pares $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ com $\mathbf{x}_i \in \mathbb{R}^d$ e $y_i \in \{-1, 1\}$. Assumem-se duas suposições, que são as seguintes:

1. Cada par $(\mathbf{x}_i, y_i)$ é uma observação de um vetor aleatório $(\mathbf{X}, Y)$ com distribuição de probabilidade conjunta $F_{\mathbf{X},Y}(\mathbf{x}, y)$
2. $\mathbf{X}$ e Y não são independentes.

A tarefa de aprendizado desse problema é encontrar uma função f utilizando os dados D respeitando as suposições acima, tal que $P(f(\mathbf{X}) = Y)$ é alta. A suposição (1) se refere aos dados serem da mesma distribuição. Já a suposição (2) existe para o problema de classificação ser interessante, já que se $\mathbf{X}$ e Y forem independentes, não faz sentido aprender uma função que mapeia relação entre instâncias e rótulos.

3.2 Diferentes Definições para Aprendizado com Proporções de Rótulos

Nesta seção, serão definidas duas versões do problema de Aprendizado com Proporções de Rótulos: a versão padrão da literatura e uma nova definição para capturar cenários mais gerais, considerando alguns aspectos práticos que serão discutidos ao decorrer da seção.

3.2.1 Aprendizado com Proporções de Rótulos Padrão

A entrada de LLP é um conjunto de pares $D = \{(\mathbf{x}_i, b_i)\}_{i=1}^{N}$ e um vetor $\mathbf{p}$, tal que $\mathbf{x}_i \in \mathbb{R}^d$, $b_i \in \{1, \ldots, K\}$, cada $\mathbf{x}_i$ é associada a um rótulo $y_i \in \{-1, 1\}$ (desconhecido) e para cada $k \in \{1, \ldots, K\}$, pode-se definir p_k como:

$$p_k = \frac{|\{(\mathbf{x}_i, y_i, b_i), i = 1, \ldots, N : b_i = k \text{ e } y_i = 1\}|}{|\{(\mathbf{x}_i, y_i, b_i), i = 1, \ldots, N : b_i = k\}|}.$$

Em outras palavras, p_k é a proporção de 1's na *bag* k. O vetor $\mathbf{p}$ é conhecido. A notação B_i será utiliza durante o trabalho para referenciar as instâncias de D que estão na *bag* i, ou seja, $B_i = \{(\mathbf{x}_j, b_j) \in D \mid b_j = i\}$.

O problema de Aprendizado com Proporções de Rótulos Padrão será definido utilizando as suposições a seguir:

1. cada $(\mathbf{x}_i, y_i, b_i)$ é uma observação de um vetor aleatório $(\mathbf{X}, Y, Z)$ com distribuição de probabilidade conjunta $F_{\mathbf{X},Y,Z}(\mathbf{x}, y, b)$
2. Y e $\mathbf{X}$ não são independentes.
3. Z e $(\mathbf{X}, Y)$ não são independentes.
4. Dado Y, Z e $\mathbf{X}$ são independentes.

Esta definição de LLP, chamada de LLP Padrão neste trabalho, captura melhor a intuição sobre o problema de LLP na literatura.

As suposições (1) e (2) são as semelhantes a do aprendizado supervisionado, mas no contexto de LLP, adicionando a informação de *bag*. A suposição (3) é relacionada com as *bags* possuírem proporções diferentes. Já a (4) é definindo que as *bags* são amostras aleatórias estratificadas do conjunto de dados completo.

A vantagem da definição do Aprendizado com Proporções de Rótulos Padrão é poder, na validação cruzada, utilizar algumas *bags* no conjunto de treino e outras no conjunto de validação, como nos métodos propostos por HERNÁNDEZ-GONZÁLEZ (2019). Mas como já discutido, em cenários gerais de LLP, técnicas que utilizam essa metodologia podem não funcionar bem.

A seguir, será mostrado a aplicação da técnica de k-fold estratificado proposta por HERNÁNDEZ-GONZÁLEZ (2019) em um conjunto de dados sintético de LLP padrão para encontrar a melhor combinação de hiperparâmetros de um algoritmo de LLP nesses dados. Então, esta mesma técnica será aplicada em um cenário que viola a definição acima, motivando a utilização de outras técnicas de divisão dos dados entre treino e validação para a escolha de hiperparâmetros. Neste caso, será utilizado para medir o erro

do algoritmo no conjunto de validação a função mais utilizada na literatura de LLP, que será chamada neste trabalho de `erro-abs`, dada por:

$$\texttt{erro-abs}(\hat{\boldsymbol{p}}, \boldsymbol{p}) = \sum_{i=1}^{K} |p_i - \hat{p}_i|, \tag{3.1}$$

onde K é o número de *bags* e $\hat{p}_i$ é a proporção predita na i-ésima *bag*. Note que, técnicas que utilizam uma subconjunto de *bags* inteiros para o conjunto de validação, como o k-fold estratificado, não terão as K *bags* no conjunto de validação. Portanto, o erro das *bags* que não estão presentes no conjunto de validação é nulo.

A Figura 6 mostra o conjunto de dados de LLP padrão que será utilizado para mostrar a utilização do k-fold estratificado de HERNÁNDEZ-GONZÁLEZ (2019). Esse conjunto de dados será chamado de *default-toy-llp* no decorrer do texto. O *default-toy-llp* possui 500 instâncias, 5 *bags* com 100 instâncias em cada. As proporções de cada *bag* estão mostradas na Figura 6.

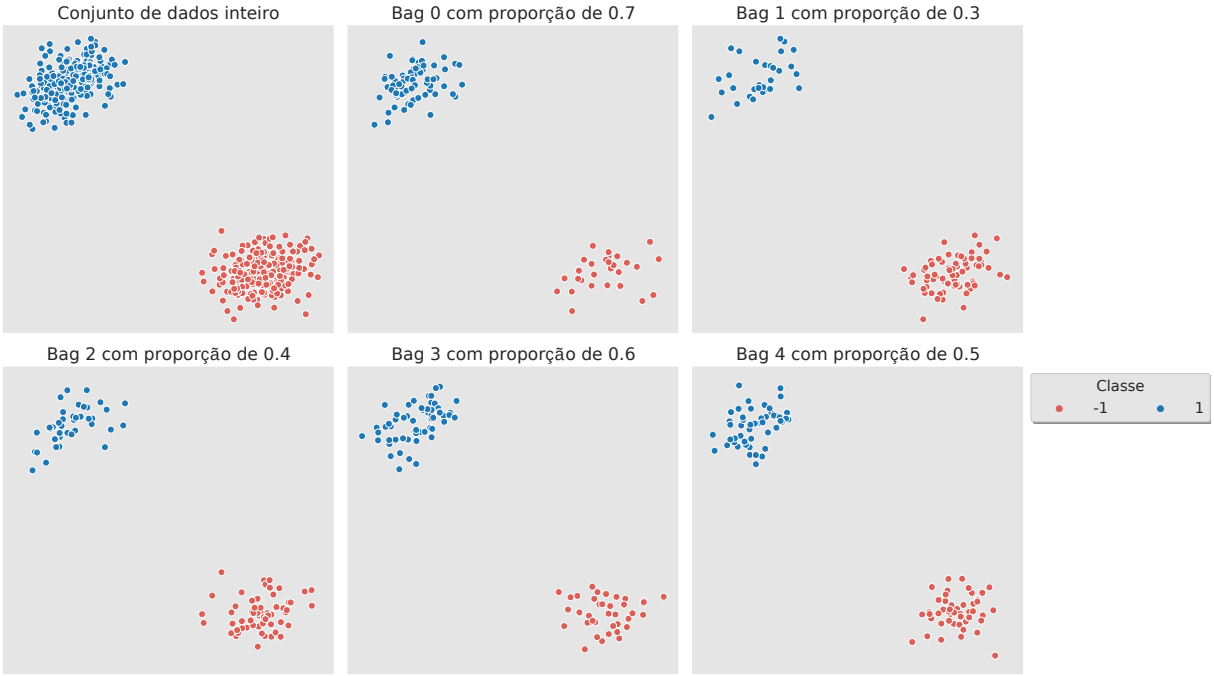


Figura 6 – Conjunto de Dados *default-toy-llp*.

Este conjunto de dados é linearmente separável e fácil de obter uma acurácia de 100% utilizando um SVM supervisionado com *kernel* linear. Note que este é um cenário comparativo, em que se assume conhecer os rótulos das instâncias para entender a complexidade do problema, sendo que os algoritmos de LLP não utilizam esses rótulos em nenhuma parte do processo de treinamento.

Se tratando de um cenário de LLP, utilizando o algoritmo alter-SVM com *kernel* linear em um processo de seleção de hiperparâmetros aplicando o k-fold estratificado com $k = 5$, i.e., cada partição conterá uma *bag*, uma acurácia de 100% também é obtida para o

conjunto de dados *default-toy-llp*. Portanto, em cenários de LLP padrão, é possível utilizar técnicas de validação cruzada que utilizam algumas *bags* para treinar e outras para testar sem comprometer o desempenho do algoritmo.

No entanto, este não é o caso cenários de LLP que violam a definição padrão. A Figura 7 mostra o conjunto de dados sintéticos *small-toy-llp-4k*. Este conjunto de dados foi desenvolvido para estudar cenários de LLP onde a definição padrão não é respeitada. A Figura 8 dá uma visão por *bag* do *small-toy-llp-4k*.

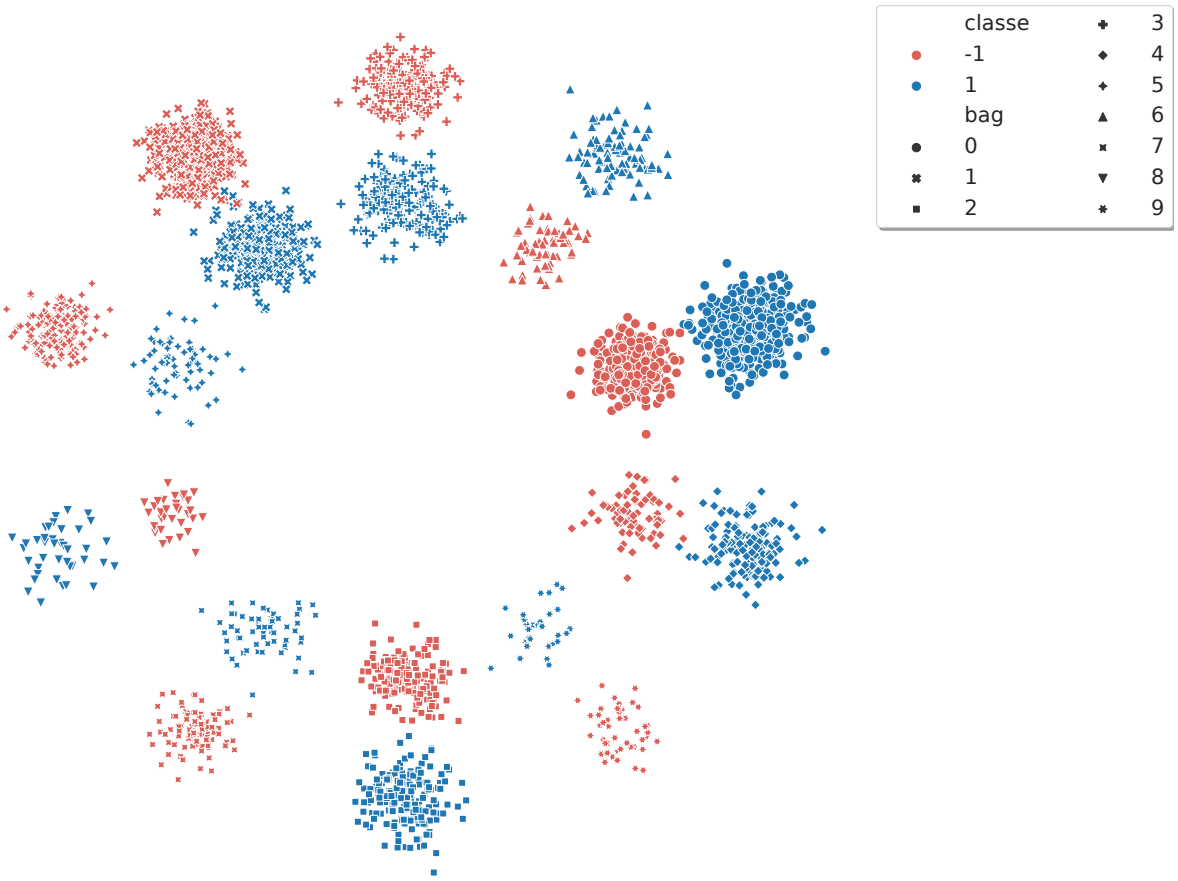


Figura 7 – Conjunto de Dados *small-toy-llp-4k*.

No caso do *small-toy-llp-4k*, que não não é linearmente separável, é necessário utilizar um SVM com *kernel* que não seja linear, neste caso foi utilizado o *kernel* RBF. No caso supervisionado, o SVM com kernel RBF consegue acurácia de 99%. Já em um cenário de LLP, utilizando $k = 5$ para o k-fold estratificado e um alter-SVM com *kernel* RBF, uma acurácia de 62% é obtida. Neste caso, o problema foi o modelo errado ter sido aprendido devido ao conjunto de dados utilizado para treinamento não ter sido representativo em relação ao conjunto de dados completo. Mais detalhes sobre os experimentos nesse conjunto de dados serão mostrados na seção de experimentos. O objetivo aqui é motivar a utilização de outras técnicas de validação cruzada em cenários que violam a definição padrão de LLP, os quais não são abordados na literatura e que serão formalmente

definidos na próxima seção.

3.2.2 Aprendizado com Proporções de Rótulos Geral

O problema de Aprendizado com Proporções de Rótulos Geral será definido removendo a suposição (4) da definição de Aprendizado com Proporções de Rótulos Padrão, ou seja, *bags* e instâncias podem ser dependentes dado os rótulos. Isto significa que instâncias podem mudar de rótulo dependendo da *bag* na qual elas estão. Portanto, esta definição de LLP captura conceitos práticos importantes, o que a torna mais adequada para problemas reais de LLP.

Por exemplo, podem existir duas instâncias parecidas em *bags* diferentes com os rótulos reais diferentes. Se uma dessas *bags* está no conjunto de treino e outra no conjunto de validação, o modelo pode aprender uma função que irá errar o rótulo de uma dessas instâncias no conjunto de validação. Isso dá uma ideia do porquê treinar em algumas *bags* e validar em outras é ruim no cenário de LLP geral. Esse tipo de comportamento pode ser visto no cenário de eleições americanas. Neste contexto, um democrata no Texas pode parecer mais um republicano na Califórnia do que um democrata na Califórnia.

3.3 Discussões

A principal questão que este trabalho visa responder é: “como escolher o melhor conjunto de hiperparâmetros em um modelo de LLP?". Essa questão pode ser dividida em duas, que são: “como dividir os dados em conjuntos de treino e validação para a seleção de hiperparâmetros?” e “como medir o erro de um modelo no conjunto de validação da melhor forma?”.

Como motivado na seção anterior, em cenários de LLP geral, treinar em algumas *bags* e testar em outras pode não funcionar bem. Além disso, a função de erro padrão de LLP (ver Equação 3.1) pode levar a medições equivocadas de erro, como discutido na Seção 2.2 e mostrado na Figura 5. As abordagens desse trabalho para solucionar esses dois problemas serão apresentadas na próxima seção.

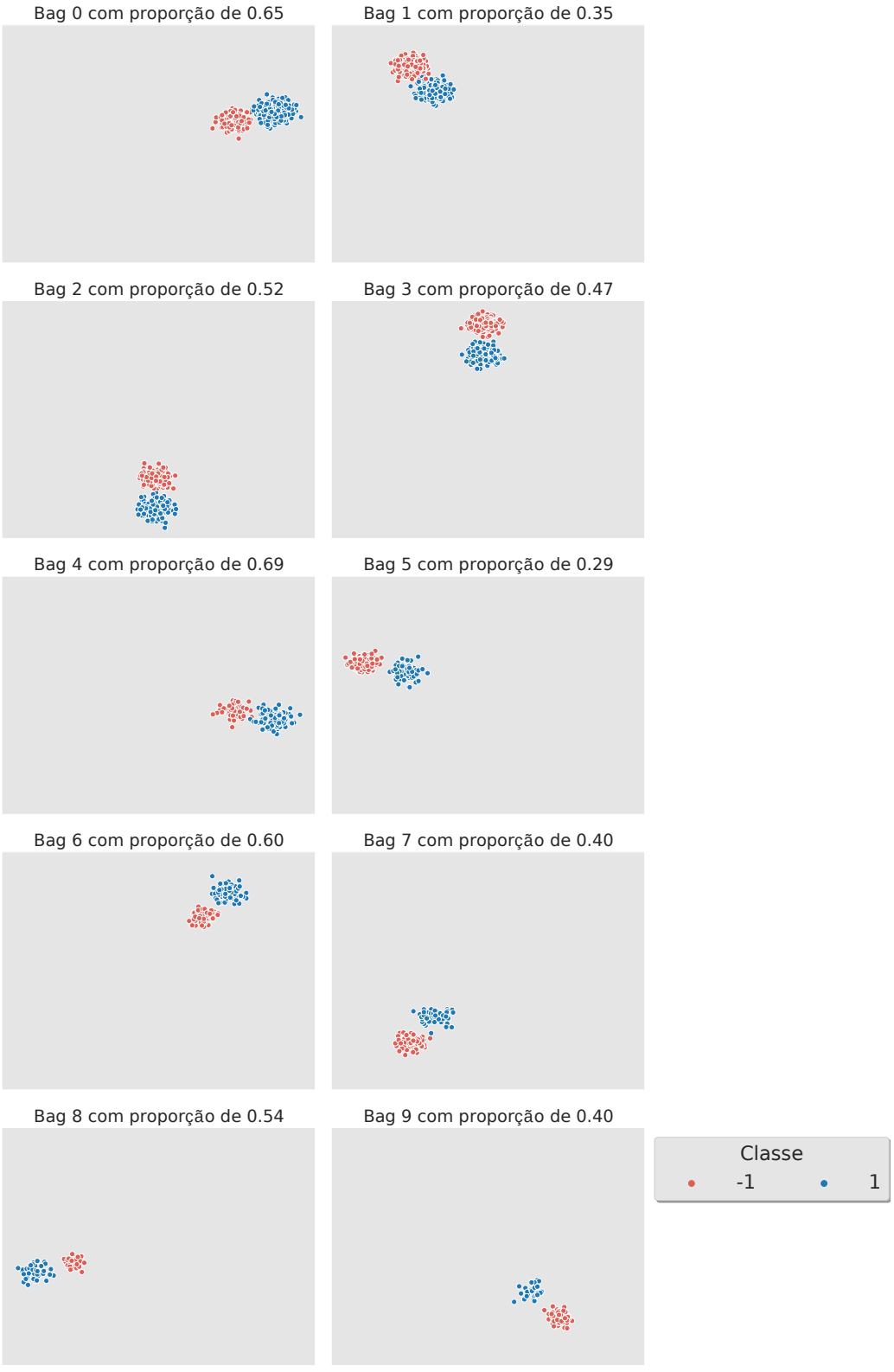


Figura 8 – Conjunto de Dados *small-toy-llp-4k* separado por *bag*.

4 Uma nova metodologia para seleção de hiperparâmetros em Aprendizado com Proporções de Rótulos

Neste capítulo serão apresentados os métodos propostos para selecionar hiperparâmetros de algoritmos de LLP. Primeiro será proposta uma estratégia de seleção de hiperparâmetros genérica para LLP. Após essa definição, os métodos de divisão dos dados entre treino e validação e de computação de erro na seleção de hiperparâmetros propostos por esse trabalho serão apresentados.

4.1 Estratégia de seleção de hiperparâmetros genérica para LLP

Como dito na Seção 3.3, é possível dividir a seleção de hiperparâmetros em duas partes: divisão dos dados entre treino e validação e computar o erro do modelo dada uma certa combinação de hiperparâmetros e os conjuntos de dados de treino e validação. Uma estratégia de seleção de hiperparâmetros genérica pode ser definida abstraindo essas duas partes.

Sejam $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$ F divisões dos dados de entrada D em treino e validação. Seja $\boldsymbol{p}$ o vetor de proporções das *bags* e Ω o espaço de hiperparâmetros que será testado, onde cada elemento de Ω é uma n-upla com hiperparâmetros do algoritmo que serão testados. Com esses dados de entrada, uma estratégia de seleção de hiperparâmetros genérica para LLP é mostrada no Algoritmo 2. O Algoritmo começa iterando no espaço de hiperparâmetros (linha 1). Então, para cada divisão dos dados entre treino e validação (linha 2), um modelo de LLP é treinado utilizando o conjunto de dados de treino e a combinação de hiperparâmetros corrente (linha 3). Então, o modelo treinado anteriormente é utilizado para prever os rótulos dos dados de validação (linha 4). Com posse desses rótulos, as proporções preditas de cada *bag* no conjunto de validação são calculadas (linha 5) para então o erro entre as proporções preditas no conjunto de validação e as proporções reais ser computado (linha 6). O erro de uma certa combinação de hiperparâmetros é dados pela média do erro entre proporções nas F execuções (linha 7). Por fim, o algoritmo retorna a combinação de hiperparâmetros que minimiza o erro (linha 8).

A contribuição que será dada no decorrer desse capítulo é propor estratégias para as duas partes genéricas do Algoritmo 2:

Algoritmo 2: Estratégia de seleção de hiperparâmetros genérica para LLP

Entrada: Dados divididos em F pares de treino e validação $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$, espaço de hiperparâmetros Ω, vetor de proporções das *bags* $\boldsymbol{p}$

Saída: Melhor combinação de hiperparâmetros

para cada $\omega \in \Omega$ **faça**
  para cada $i \leftarrow 1$ **até** F **faça**
    Treine um modelo $\mathcal{M}$ em $D_{i,\text{Treino}}$ utilizando ω
    $\hat{\boldsymbol{y}}_{i,\omega}$ = Use $\mathcal{M}$ para prever os rótulos de $D_{i,\text{Validação}}$
    Compute o vetor com as proporções preditas $\hat{\boldsymbol{p}}$ utilizando $\hat{\boldsymbol{y}}_{i,\omega}$
    $\text{erro}_{i,\omega}$ = `compute-erro`$(\hat{\boldsymbol{p}}, \boldsymbol{p})$
  $\text{erro}_{\omega} = \frac{1}{F}\sum_{i=1}^{F} \text{erro}_{i,\omega}$
retorna ω que minimiza erro_{ω}

- Metodologia de divisão dos dados entre conjuntos de treino e validação (Seção 4.2).
- Metodologia para o cálculo o erro do modelo de LLP dado uma certa combinação de hiperparâmetros e os conjuntos de dados de treino e validação (Seção 4.3).

4.2 Divisão dos dados entre os conjuntos de treino e validação

Se tratando de aprendizado supervisionado, existem várias formas de dividir os dados em F pares de treino e validação no contexto de seleção de hiperparâmetros. Uma dessas formas é o k-fold, que foi mostrado na Seção 2.1.3. Outra forma bastante utilizada para fazer esse divisão é amostrar aleatoriamente uma porcentagem grande dos dados para formar o conjunto de treino e utilizar os dados que sobrarem para o conjunto de validação, repetindo esse processo F vezes. No contexto de aprendizado supervisionado, esses métodos funcionam bem para representar a distribuição dos dados inteiros. No entanto, se tratando de LLP, a utilização direta dessas estratégias não é o ideal.

Como o foco desse trabalho são cenários de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2), é importante possuir uma amostra de cada *bag* no conjunto de treino aos invés de utilizar *bags* inteiras para os conjuntos de treino e validação, como mostra o último exemplo da Seção 3.2.1. Além disso, o ideal é fazer uma divisão na qual as proporções dos conjuntos de treino e validação são próximas das proporções globais do conjunto de dados, já que essas proporções que são utilizadas para calcular o erro do modelo na seleção de hiperparâmetros genérica apresentada pelo Algoritmo 2.

Uma forma de dividir os dados entre treino e validação é utilizando diretamente técnicas de aprendizado supervisionado, não levando em consideração a estrutura dos dados em *bags*. Neste caso, existem grandes chances de o erro amostral por *bag* ser imenso, já que esse processo não leva em consideração que os dados estão divididos em *bags* no problema de LLP. *Bags* de tamanho pequeno seriam as mais prejudicadas, pois além da

maior suscetibilidade a maiores erros amostrais, elas poderiam estar presentes no conjunto de dados de treino e não estarem no conjunto de validação ou vice-versa. Para resolver esse problema, a divisão dos dados entre treino e validação será feita por *bag* nas abordagens propostas por esse trabalho. Assim, o erro amostral é reduzido e todas as *bags* serão representadas em ambos conjuntos de treino e validação.

No decorrer dessa seção, serão apresentadas três algoritmos de divisão dos dados entre treino e validação adaptados para o contexto de LLP. São eles:

1. *k-fold-LLP*: Faz um k-fold considerando as *bags* para formar as partições.
2. *Shuffle-Split-LLP*: Faz uma amostragem sem repetição por *bag* para formar os conjuntos de treino e validação, repetindo esse processo F vezes.
3. *Bootstrapping-Split-LLP*: Faz uma amostragem com repetição por *bag* para formar os conjuntos de treino e validação, repetindo esse processo F vezes.

4.2.1 *k-fold-LLP*

O k-fold é uma técnica que divide os dados em F (utilizando a notação de número de partições F para não confundir com o número de *bags* K) partes iguais, utilizando $F-1$ partes para treino e uma parte para validação, rotacionando entre as partes F vezes (para mais detalhes, ver Seção 2.1.3). O *k-fold-LLP* funciona de uma forma similar, mas dividindo cada *bag* em F partes iguais. Dessa forma, espera-se que as proporções de cada uma das F partes seja próxima da proporção da *bag*.

A Figura 9 mostra o esquema de funcionamento do *k-fold-LLP* para LLP dividido em três passos. O passo 0 referencia os dados de LLP com K *bags*. No passo 1, para cada *bag* i, os dados dessa *bag* são embaralhados e divididos em F partes disjuntas de mesmo tamanho $T_{i,1}, \ldots, T_{i,F}$. Já no passo 2, as partições do conjunto de dados completo são formadas, sendo que a união das primeiras partições de cada uma das *bags* formam a primeira partição dos dados, e assim sucessivamente. Com as F partições do conjunto de dados, a mesma ideia do algoritmo k-fold de aprendizado supervisionado é utilizada, i.e., utilizar $F-1$ partes para treino e uma parte para validação, rotacionando as partes que serão utilizadas para a validação.

O Algoritmo 3 mostra o pseudocódigo do *k-fold-LLP*. As linhas 1-3 do algoritmo dividem os dados de cada *bag* em F partes de tamanhos iguais, como mostra o passo 1 da Figura 9. Já as linhas 4-5 se referem ao passo 2 da Figura 9, i.e., a i-ésima partição dos dados completos é formada pela união da i-ésima partição de cada *bag*. Por fim, as linhas 6-8 fazem a lógica do k-fold, ou seja, utilizar $F-1$ partições para formar o conjunto de treinamento e a partição restante para formar o conjunto de validação, rotacionando entre as partes utilizadas para a validação F vezes.

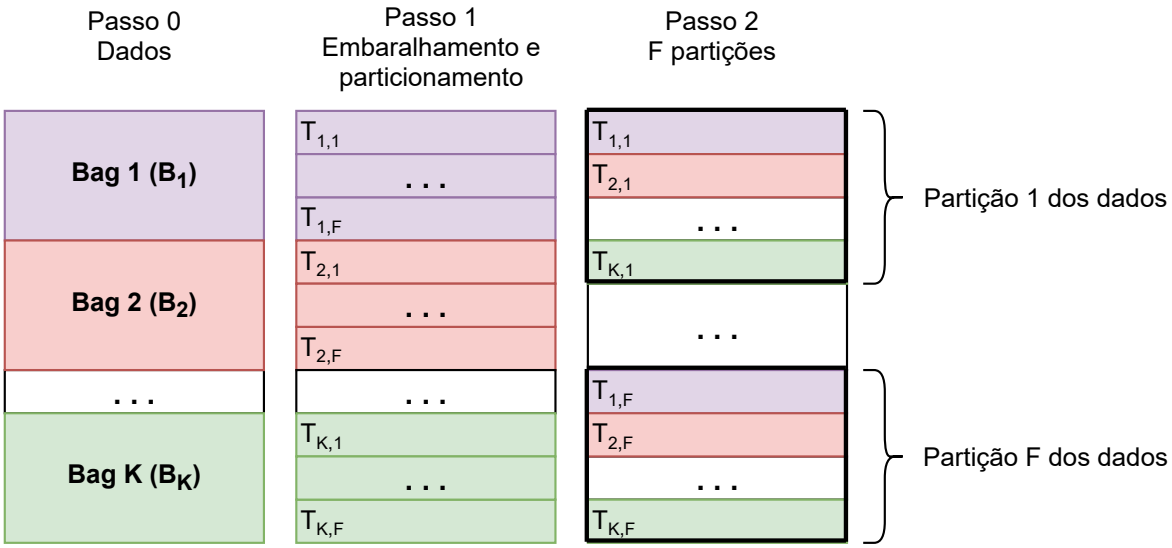


Figura 9 – Funcionamento do algoritmo *k-fold-LLP*.

Algoritmo 3: *k-fold-LLP*

Entrada: Conjunto de dados D, número de *bags* K e número de partições F
Saída: Dados divididos em F pares de treino e validação
$(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$

para cada $i \leftarrow 1$ **até** K **faça**
  Embaralhe e divida B_i em F partes $T_{i,1}, \ldots, T_{i,F}$ disjuntas de tamanhos iguais
para cada $f \leftarrow 1$ **até** F **faça**
  $U_f = \bigcup_{i=1}^{K} T_{i,f}$
para cada $f \leftarrow 1$ **até** F **faça**
  $D_{f,\text{Validação}} = U_f$
  $D_{f,\text{Treino}} = D - U_f$
retorna $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$

4.2.2 *Shuffle-Split-LLP*

O *Shuffle Split* é uma divisão aleatória dos dados entre treino e validação, com uma proporção β dos dados para validação e 1 - β para treino. Para utilizar essa técnica em seleção de hiperparâmetros, esse processo é repetido F vezes de forma independente, obtendo F pares de conjuntos de treino e validação distintos. Diferentemente do k-fold, não existe garantia que uma instância estará pelo menos uma vez no conjunto de validação utilizando esse método.

No *Shuffle-Split-LLP*, essa divisão aleatória dos dados entre treino e validação será feita por *bag*, utilizando a mesma proporção β dos dados para validação para todas as *bags*. A união dos conjuntos de validação das *bags* formará o conjunto de validação de todo o conjunto de dados. De forma análoga, o conjunto de treino é formado pela união dos conjuntos de treino das *bags*.

A Figura 10 mostra uma divisão dos dados entre treino e validação utilizando o *Shuffle-Split-LLP*. O passo 0 é referente aos dados de LLP com K *bags*, sendo B_i os dados da *bag* i. No passo 1, cada *bag* é dividida em dois conjuntos: treino e validação. Dado a proporção de dados no conjunto de validação β, o conjunto de validação $T_{i,\text{Validação}}$ da *bag* i é formado por uma amostra aleatória sem repetição de B_i de tamanho $|B_i|\cdot\beta$. Já o conjunto de treino $T_{i,\text{Treino}}$ da *bag* i é formado por todos os dados de B_i que não estão em $T_{i,\text{Validação}}$. No passo 2, a união dos conjuntos de treino das *bags* formam o conjunto de treino dos dados completos. De forma similar, o conjunto de validação é formado pela união dos conjuntos de validação das *bags*. Assim como no *Shuffle Split*, esse processo é repetido F vezes de forma independente, gerando F pares distintos de dados de treino e validação para a estratégia de seleção de hiperparâmetros genérica para LLP (ver Algoritmo 2).

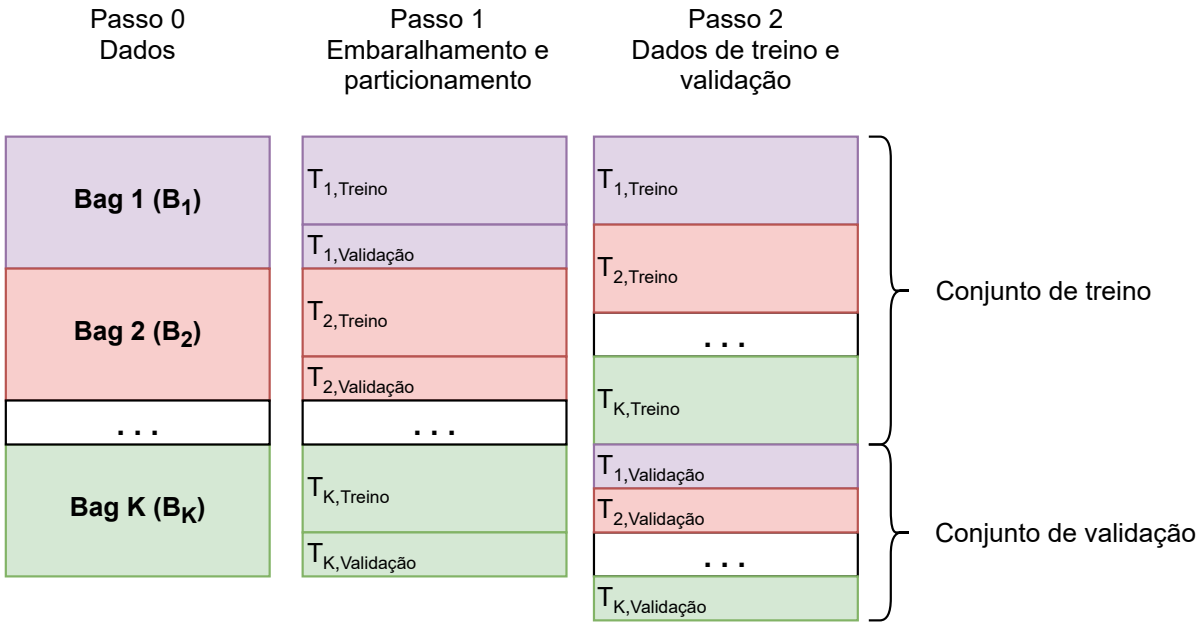


Figura 10 – Funcionamento do algoritmo *Shuffle-Split-LLP*.

O Algoritmo 4 mostra a geração de F pares de dados de treino e validação utilizando o *Shuffle-Split-LLP*. As linhas 1-3 se referem a geração de F pares de dados de treino e validação independentes. As linhas 5-6 são referentes ao passo 1 da Figura 10, ou seja, criam os conjuntos de treino e validação de cada *bag*. Já as linhas 7-8 se referem ao passo 2 da Figura 10, no qual os dados de treino do conjunto de dados completo são formados pela união dos conjuntos de treino das *bags*, com o conjunto de validação dos dados completos sendo criado de forma análoga. Esse processo das linhas 5-9 é repetido para cada uma das *bags* (linha 4).

4.2.3 *Bootstrapping-Split-LLP*

Bootstrapping é um procedimento estatístico que utiliza amostras aleatórias com repetição e re-amostragem para fazer estimativas de estatísticas sobre a população (EFRON,

Algoritmo 4: *Shuffle-Split-LLP*

Entrada: Conjunto de dados D, número de *bags* K, número de pares de treino e validação F e a proporção de dados no conjunto de validação β

Saída: Dados divididos em F pares de treino e validação $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$

1 **para cada** $f \leftarrow 1$ **até** F **faça**
2 $D_{f,\text{Treino}} = \emptyset$
3 $D_{f,\text{Validação}} = \emptyset$
4 **para cada** $i \leftarrow 1$ **até** K **faça**
5 Seja $T_{i,\text{Validação}}$ uma amostra aleatória sem repetição de B_i de tamanho $|B_i| \cdot \beta$
6 $T_{i,\text{Treino}} = B_i - T_{i,\text{Validação}}$
7 $D_{f,\text{Validação}} = D_{f,\text{Validação}} \bigcup T_{i,\text{Validação}}$
8 $D_{f,\text{Treino}} = D_{f,\text{Treino}} \bigcup T_{i,\text{Treino}}$
9 **retorna** $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$

1992). Amostras aleatórias com repetição podem selecionar o mesmo elemento mais de uma vez para a mesma amostra. O algoritmo *Bootstrapping-Split-LLP* é semelhante ao *Shuffle-Split-LLP*, mas utilizando amostras aleatórias com repetição para dividir os dados nos conjuntos de treino e validação de cada *bag*. A ideia de usar amostras aleatórias com repetição nesse contexto é para tentar diminuir a variância das amostras, obtendo assim conjuntos de dados de treino e validação com menos erros amostrais, principalmente em casos com *bags* pequenas e com poucas instâncias de uma certa classe.

Por exemplo, suponha um caso extremo de uma *bag* com um único rótulo positivo. Utilizando o *k-fold-LLP*, a proporção da *bag* no conjunto de treino será diferente da proporção da *bag* no conjunto de validação, assim como no *Shuffle-Split-LLP*, já que a instância positiva só poderá ficar em uma das partições (treino ou validação). Já utilizando o *Bootstrapping-Split-LLP*, a instância positiva poderia aparecer em ambos os conjuntos de treino e validação, o que ocasiona um menor erro de amostragem nesse caso.

A Figura 11 mostra uma divisão dos dados entre treino e validação utilizando o *Bootstrapping Split*. O passo 0 é referente aos dados de LLP com K *bags*, sendo B_i os dados da *bag* i. No passo 1, os conjuntos de treino e validação de cada *bag* são formados. Dado a proporção de dados no conjunto de validação β, o conjunto de treino $T_{i,\text{Treino}}$ da *bag* i é formado por uma amostra aleatória com repetição de B_i de tamanho $|B_i| \cdot (1 - \beta)$ e o conjunto de validação $T_{i,\text{Validação}}$ da *bag* i é formado por uma amostra aleatória com repetição de B_i de tamanho $|B_i| \cdot \beta$. No passo 2, a união dos conjuntos de treino das *bags* formam o conjunto de treino dos dados completos. De forma similar, o conjunto de validação é formado pela união dos conjuntos de validação das *bags*. Os F pares de treino e validação utilizados pela estratégia de seleção de hiperparâmetros genérica para LLP (ver Algoritmo 2) são obtidos repetindo esse processo F vezes de forma independente.

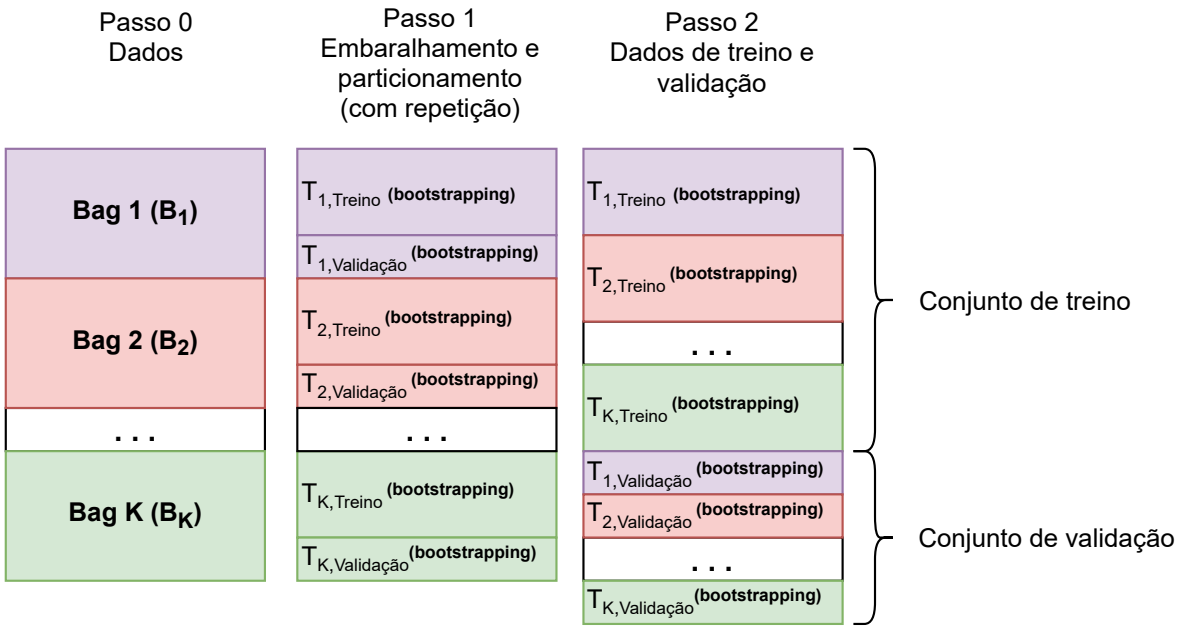


Figura 11 – Funcionamento do algoritmo *Bootstrapping-Split-LLP*.

O Algoritmo 5 mostra o processo de geração dos F pares de dados de treino e validação utilizando o *Bootstrapping-Split-LLP*. As linhas 1-3 se referem a geração de F pares de dados de treino e validação independentes. As linhas 5-6 são referentes ao passo 1 da Figura 11, i.e., criam os conjuntos de treino e validação de cada *bag* utilizando amostragem com repetição. Já as linhas 7-8 se referem ao passo 2 da Figura 11, no qual a união dos conjuntos de treino das *bags* formam o conjunto de treinamento e a união dos conjuntos de validação das *bags* formam o conjunto de validação. Esse processo das linhas 5-9 é repetido para cada uma das *bags* (linha 4).

Algoritmo 5: *Bootstrapping-Split-LLP*

Entrada: Conjunto de dados D, número de *bags* K, número de pares de treino e validação F e a proporção de dados no conjunto de validação β

Saída: Dados divididos em F pares de treino e validação $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$

1 **para cada** $f \leftarrow 1$ **até** F **faça**
2 $\quad D_{f,\text{Treino}} = \emptyset$
3 $\quad D_{f,\text{Validação}} = \emptyset$
4 $\quad$ **para cada** $i \leftarrow 1$ **até** K **faça**
5 $\quad\quad$ Seja $T_{i,\text{Treino}}$ uma amostra aleatória **com** repetição de B_i de tamanho $|B_i| \cdot (1 - \beta)$
6 $\quad\quad$ Seja $T_{i,\text{Validação}}$ uma amostra aleatória **com** repetição de B_i de tamanho $|B_i| \cdot \beta$
7 $\quad\quad D_{f,\text{Treino}} = D_{f,\text{Treino}} \bigcup T_{i,\text{Treino}}$
8 $\quad\quad D_{f,\text{Validação}} = D_{f,\text{Validação}} \bigcup T_{i,\text{Validação}}$
9 **retorna** $(D_{1,\text{Treino}}, D_{1,\text{Validação}}), \ldots, (D_{F,\text{Treino}}, D_{F,\text{Validação}})$

Nesta seção, as abordagens para dividir os dados entre treino e validação foram definidas. Formas para computar o erro do modelo na estratégia de seleção de hiperparâmetros genérica para LLP serão discutidas na próxima seção.

4.3 Medição do erro do modelo de LLP na seleção de hiperparâmetros

Para utilizar a estratégia genérica de seleção de hiperparâmetros para LLP (ver Algoritmo 2) é necessário definir, além do método de divisão dos dados entre treino e validação, a forma de computar o erro de predição no conjunto de validação do modelo de LLP treinado dado uma certa combinação de hiperparâmetros (ver a função `compute-erro` no Algoritmo 2). O cálculo desse erro é de suma importância para a escolha de hiperparâmetros em LLP, já que a combinação de hiperparâmetros escolhida no processo de seleção de hiperparâmetros é a que minimiza esse erro.

Definir essa função de erro no contexto de aprendizado supervisionado é uma tarefa mais simples, já que todos os rótulos reais individuais estão disponíveis. Já se tratando de LLP, computar esse erro não é uma tarefa simples. As principais informações disponíveis são o tamanho das *bags*, as proporções das *bags* do conjunto completo e as proporções das *bags* preditas pelo modelo de LLP no conjunto de validação. A função de erro mais utilizada na literatura para esse fim é a soma das diferenças em módulo entre as proporções preditas e as reais, chamada de `erro-abs` (ver Equação 3.1).

Como já dito na Seção 2.2 e exemplificado pela Figura 5, em conjuntos de dados que possuem *bags* com proporções próximas de 0.5, os modelos de LLP podem conseguir um erro de proporções igual a zero nessas *bags* e mesmo assim obter uma acurácia de 0.0 nas mesmas. Portanto, *bags* com proporção próxima de 0.5 proveem menos informação para o cálculo do erro em relação a *bags* com proporção mais distante de 0.5. Além disso, *bags* menores são mais sensíveis a erro de amostragem na divisão dos dados entre treino e validação, sendo esse erro de amostragem bastante prejudicial no cálculo do erro. Ou seja, quanto maior a *bag*, menor a chance de um erro grande de amostragem, e consequentemente, maior é a quantidade de informação que essa *bag* provê para o cálculo do erro.

A função de erro `erro-abs` não utiliza a informação de cada uma das *bags* para o cálculo do erro. Uma forma simples de resolver esse problema é, no cálculo do erro, dar pesos distintos para as *bags* baseados na quantidade de informação das mesmas. Na próxima seção, será apresentado um novo esquema de pesos proposto por este trabalho para levar em consideração a informação das *bags* será apresentado.

4.3.1 Um novo esquema de pesos para funções de erro de LLP

O esquema de pesos para funções de erro de LLP que será proposto nessa seção tem como objetivo capturar a quantidade de informação que uma *bag* possui. Quanto mais elementos uma *bag* possui, mais informativa ela é, por ser menor sensível a erros de amostragem. Além disso, quanto mais distante de 0.5 a proporção de uma *bag* é, mais informação ela possui. Se uma *bag* tem proporção de 0.0 ou 1.0, ela possuí a maior quantidade de informação possível, já que os rótulos de todos os elementos dessa *bag* são conhecidos, assim como no aprendizado supervisionado.

Para capturar essa intuição de informação de uma *bag*, será usado um esquema de pesos baseados na informação de Fisher de uma variável aleatória de Bernoulli. Segundo FRIEDEN (2004), a informação de Fisher quantifica a informação que os dados possuem sobre um parâmetro θ desconhecido da distribuição dos dados. Já uma variável aleatória de Bernoulli de parâmetro θ possui probabilidade igual a θ de sucesso e $1 - \theta$ de falha. A informação de Fisher de uma variável aleatória de Bernoulli está sendo utilizada nesse contexto pois o parâmetro desconhecido da distribuição dessa variável é a probabilidade de sucesso, o que pode ser interpretado como um valor de proporção de uma *bag* no contexto de LLP, já que ambas assumes valores no intervalo $[0.0, 1.0]$. A informação de Fisher de n variáveis aleatórias de Bernoulli independentes de parâmetro θ, é dada por:

$$\mathcal{I}(\theta) = \frac{n}{\theta(1-\theta)} \tag{4.1}$$

A Figura 12 mostra o comportamento da informação de Fisher de uma variável aleatória de Bernoulli. A informação de Fisher tem seu mínimo quando $\theta = 0.5$ e tende a infinito quando $\theta = 0.0$ ou $\theta = 1.0$. Além disso, de acordo com a Equação 4.1, quanto maior o número de tentativas n, maior é a informação de Fisher.

O esquema de pesos proposto nesta seção irá utilizar a informação de Fisher de várias variáveis aleatórias de Bernoulli independentes como inspiração. Como dito anteriormente, quanto maior o tamanho de uma *bag*, maior a informação dela. Além disso, a informação da *bag* aumenta quanto mais distante a proporção dela for de 0.5. Substituindo o parâmetro θ pela proporção da *bag* e o número de tentativas pelo tamanho da *bag*, é obtida uma função de informação para *bag* que possui essas duas propriedades. Portanto, o peso que quantifica a quantidade de informação de uma *bag* pode ser definido como:

$$w_i = \frac{|B_i|}{p_i(1-p_i)} \tag{4.2}$$

Uma nova função para computar o erro do modelo de LLP na seleção de hiperparâmetros será definida utilizando o esquema de pesos definido na Equação 4.2 combinado com a função de erro `erro-abs` (ver Equação 3.1). A função de erro `erro-fisher` é

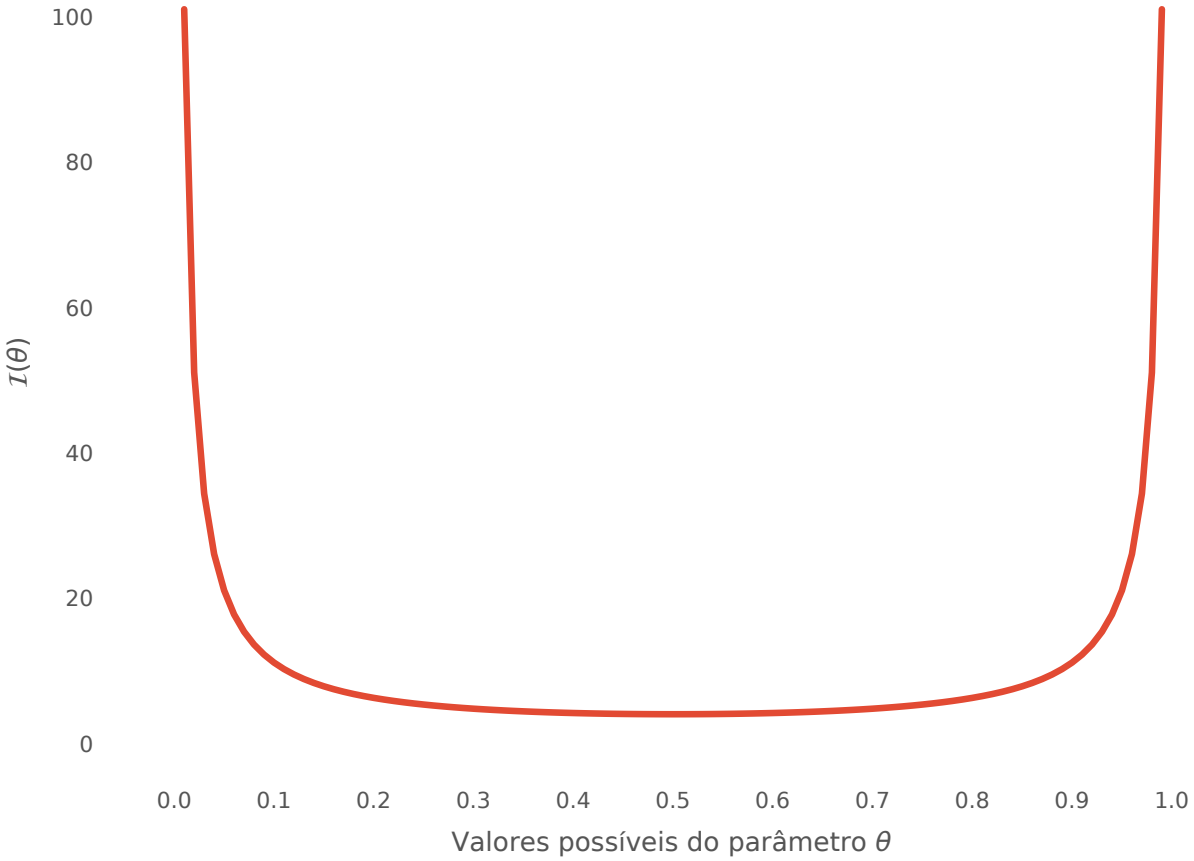


Figura 12 – Variação da informação de Fisher de uma variável aleatória de Bernoulli de acordo com o parâmetro θ.

definida como:

$$\texttt{erro-fisher}(\hat{\boldsymbol{p}}, \boldsymbol{p}) = \sum_{i=1}^{K} w_i \cdot |p_i - \hat{p}_i| \tag{4.3}$$

A ideia de utilizar essa função de erro no processo de seleção de hiperparâmetros para LLP é de que, priorizando *bags* com mais informações, modelos mais assertivos serão obtidos neste processo.

4.4 Discussões

Nesta capítulo, foram apresentadas formas de dividir os dados entre conjuntos de treino e validação e de computar o erro do modelo dada uma certa combinação de hiperparâmetros e os conjuntos de dados de treino e validação. Com isso, é possível utilizar a estratégia de seleção de hiperparâmetros genérica para LLP definida no início do capítulo para escolher os melhores hiperparâmetros para um modelo de LLP. No próximo capítulo, os resultados de experimentos serão apresentados com o intuito de comparar essas abordagens, especialmente em cenários de Aprendizado com Proporções de Rótulos

Geral.

5 Resultados e Discussão

Neste capítulo serão apresentados os resultados de experimentos comparando as abordagens apresentadas no capítulo anterior para a seleção de hiperparâmetros em algoritmos de LLP, com o foco em cenários de Aprendizado com Proporções de Rótulos Geral. No decorrer do capítulo serão mostrados os conjuntos de dados utilizados, o ambiente experimental, os resultados e as discussões acerca desses conjuntos de dados.

5.1 Conjuntos de dados

Neste seção os conjuntos de dados utilizados nos experimentos serão apresentados. Primeiro, serão mostrados os três conjuntos de dados sintéticos propostos por esse trabalho para simular cenários de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2). Então, serão apresentados dois conjuntos de dados que se adaptam a definição de Aprendizado com Proporções de Rótulos Padrão, com o intuito de avaliar o desempenho dos métodos propostos por esse trabalho em um contexto diferente.

5.1.1 Dados sintéticos

Para avaliar o desempenho dos métodos propostos por esse trabalho em cenários de Aprendizado com Proporções de Rótulos Geral foram propostos três conjuntos de dados sintéticos que satisfazem as suposições definidas na Seção 3.2.2. A forma de geração dos dados desses três conjuntos de dados sintéticos segue o mesmo padrão. Cada *bag* é formada por duas nuvens de pontos, uma de classe positiva e outra de classe negativa. Essas nuvens de pontos são amostradas de uma distribuição normal multivariada com a mesma covariância, mas a média dessa distribuição é ligeiramente deslocada para formar duas nuvens de pontos separadas. As *bags* são dispostas em um circulo maior de forma aleatória. Para cada *bag*, a classe que fica para dentro do círculo também é escolhida de forma aleatória.

Nesses três conjuntos de dados as *bags* e instâncias são dependentes. Como as *bags* estão em seções separadas do círculo que forma o conjunto de dados, a posição da instância no plano depende da *bag* que ela está. Além disso, como a disposição das classes muda em cada *bag*, existem casos de que um elemento de classe positiva de uma *bag* está mais próximo de um elemento de classe negativa de outra *bag* do que de um elemento de classe positiva, simulando o efeito já mostrado neste trabalho sobre as eleições americanas (um democrata no Texas pode parecer mais um republicano na Califórnia do que um democrata na Califórnia). No decorrer da seção os três conjuntos de dados sintéticos serão apresentados com mais detalhes.

5.1.1.1 Conjunto de dados *large-toy-llp-10k*

O primeiro conjunto de dados, chamado de *large-toy-llp-10k*, tem como objetivo simular um cenário de LLP geral, com muitas *bags* e que não seja linearmente separável. Para isso, foi utilizado o cenário das eleições americanas de 2016 como inspiração. Esse conjunto de dados possui 51 *bags*, cada uma representando um estado dos EUA, considerando Washington D.C. Os tamanhos das *bags* são proporcionais ao tamanho da população do estado, e a proporção de cada *bag* é a porcentagem de votos democratas no estado respectivo a *bag* nas eleições de 2016[1]. A Figura 13 mostra o *large-toy-llp-10k*.

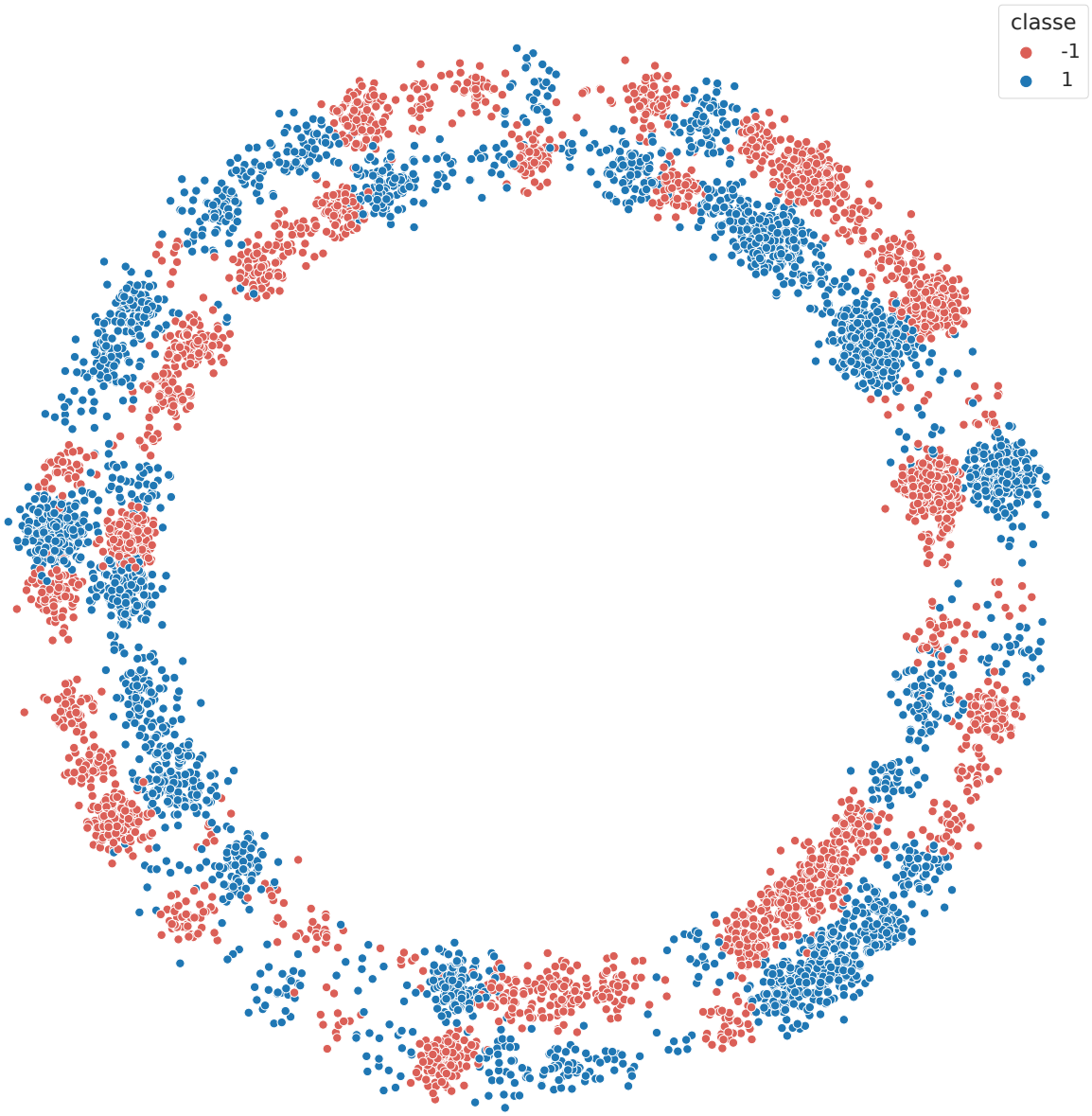


Figura 13 – Conjunto de Dados *large-toy-llp-10k*.

[1] Para mais detalhes dos resultados das eleições americanas de 2016, ver <https://www.nytimes.com/elections/2016/results/president>

5.1.1.2 Conjunto de dados *small-toy-llp-4k*

O segundo conjunto de dados é o *small-toy-llp-4k*, já mostrado na Seção 3.2.1 (ver Figuras 7 e 8). A ideia desse conjunto de dados também é simular um cenário de LLP geral mais simples, com menos *bags* em relação ao *large-toy-llp-10k*, mas mantendo o mesmo processo de geração dos dados.

5.1.1.3 Conjunto de dados *small-toy-llp-12k*

O terceiro conjunto de dados, chamado *small-toy-llp-12k*, é gerado pelo mesmo processo do *small-toy-llp-4k*, mantendo as proporções das *bags*, mas com um maior número de instâncias. Um efeito desse aumento no número de instâncias é a redução no tamanho da separação entre as *bags* e entre as classes dentro das *bags*. A Figura 14 mostra esse conjunto de dados.

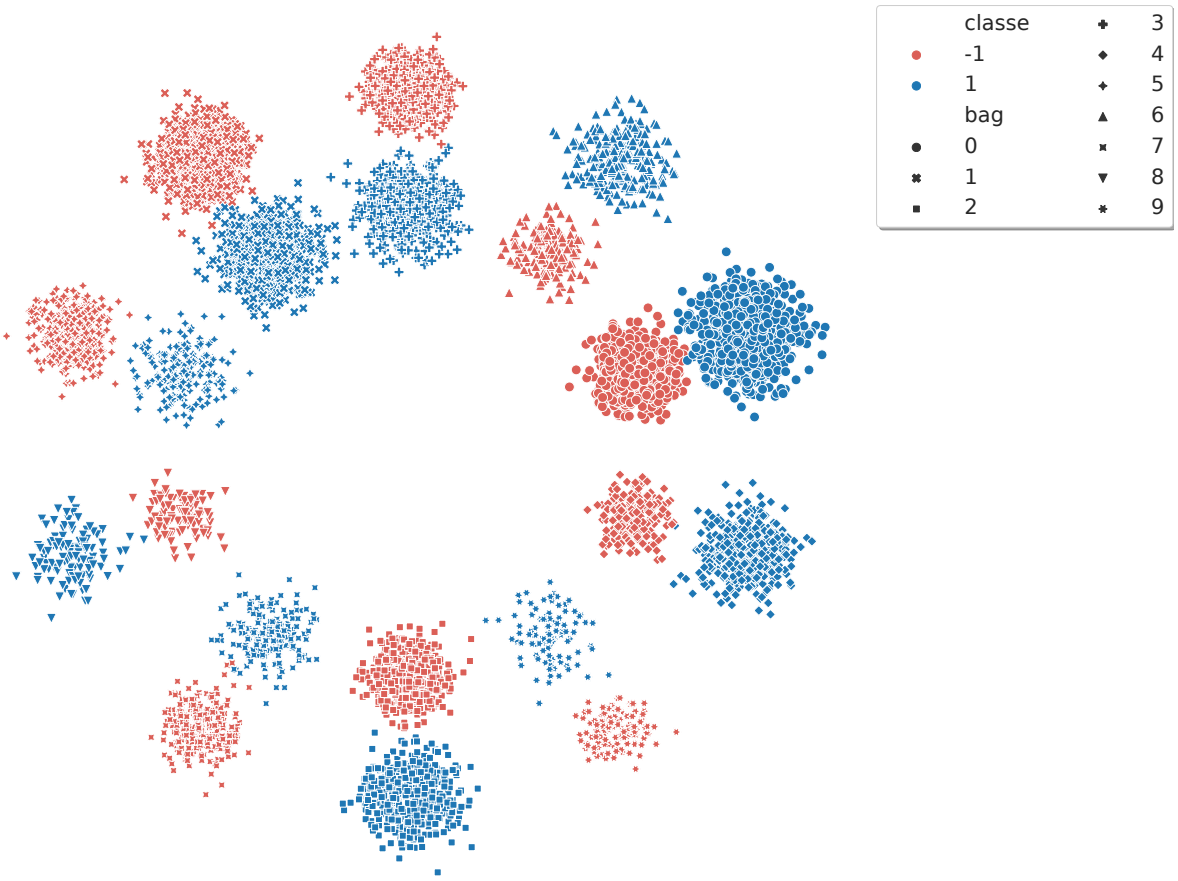


Figura 14 – Conjunto de Dados *small-toy-llp-12k*.

5.1.2 Conjuntos de dados *mrp-sim-llp-3k*

Multilevel regression and post-stratification (MRP) é uma técnica estatística bastante utilizada para estimar preferencias sub-regionais utilizando grandes dados nacionais (HANRETTY, 2020). Um exemplo clássico de uso de MRP é dado por PARK; GELMAN;

BAFUMI (2004), que utilizou pesquisas eleitorais em nível nacional para estimar o suporte a George W. Bush em nível estadual. Apesar de serem problemas distintos, existe uma relação entre MRP e LLP, já que as duas técnicas se baseiam em dados que possuem divisões regionais. Por esse motivo, um conjunto de dados de MRP adaptado para LLP será utilizado nesse trabalho.

Existe um gerador de dados sintéticos para experimentos de MRP implementado pelo pacote `rstanarm` da linguagem R (GOODRICH et al., 2020) [2]. Esse pacote possui um método chamado `simulate_mrp_data` para gerar dados simulados de MRP, com o tamanho da amostra sendo parâmetro do gerador. A Tabela 1 mostra os atributos do conjunto de dados e seus respectivos tipos. O rótulo das instâncias é a preferência por gatos, que é do tipo binário.

Atributo	**Tipo**
Sexo	Binário
Idade	Categórico (7 categorias)
Etnia	Categórico (3 categorias)
Renda	Categórico (3 categorias)
Estado	Categórico (50 categorias)

Tabela 1 – Atributos do conjunto de dados de MRP gerado pelo método `simulate_mrp_data` do pacote `rstanarm`

Foi gerado um conjunto de dados com 3000 instâncias utilizando o `simulate_mrp_data`. Para transformar esses dados em dados de LLP, o atributo estado foi utilizado como *bag*. Então, as proporções foram computadas e os rótulos removidos. Será usado o nome *mrp-sim-llp-3k* para se referir a esse conjunto de dados. Esse conjunto de dados foi utilizado pois já é bem estabelecido na literatura de MRP e possui o atributo estado, que naturalmente faz a divisão dos dados em *bags*. Além disso, os atributos categóricos foram transformados em atributos binários. Isto é, se um atributo possui três categorias, este atributo se transforma em três atributos novos, cada um destes respectivo a uma categoria. Uma instância que pertence a certa categoria possui o valor de 1 nesse novo atributo correspondente a categoria e 0 nos outros.

5.1.3 Conjuntos de dados *census-2015-llp*

O último conjunto de dados utilizado é o *census-2015-llp*, do censo americano de 2015 do *US Census Bureau*. Os dados foram retirados das tabelas DP03 e DP05 do *American Community Survey*[3] de 2015. Foi utilizado o conjunto de dados já processado disponível no Kaggle[4]. Esses dados possuem várias estatísticas sobre as micro regiões chamadas *Census Track*, que são utilizadas no censo, nível de condado e de estado. Para

[2] <https://cran.r-project.org/web/packages/rstanarm/vignettes/mrp.html>
[3] <https://www.census.gov/programs-surveys/acs>
[4] <https://www.kaggle.com/muonneutrino/us-census-demographic-data>

transformar esses dados em dados de LLP, o atributo estado foi utilizado como *bag*. A tarefa de aprendizado utilizada foi prever se a renda da instância é maior ou igual a 50000 dólares. O valor de renda de cada instância foi multiplicado pelo valor do dólar no seu estado calculado pelo *Bureau of Economic Analysis*[5]. Atributos relacionados com o estado e renda foram removidos.

A Tabela 2 mostra o número de instâncias e de atributos de cada conjunto de dados utilizado neste trabalho após todo o processamento.

Conjunto de dados	**Número de instâncias**	**Número de atributos**
large-toy-llp-10k	10002	2
small-toy-llp-4k	4200	2
small-toy-llp-12k	12600	2
mrp-sim-llp-3k	3000	13
`census-2015-llp`	71678	30

Tabela 2 – Número de instâncias e de atributos de cada conjunto de dados utilizado.

5.2 Ambiente experimental

Nesta seção, o ambiente experimental será apresentado. Primeiro, será apresentada uma descrição detalhada da forma como os experimentos foram conduzidos. Na sequência, uma discussão sobre os desafios experimentais será feita.

5.2.1 Descrição do ambiente experimental

Foram comparados as três abordagens de divisão dos dados entre treino e validação propostas na Seção 4.2 com a linha de base da literatura, que é o k-fold estratificado proposto por HERNÁNDEZ-GONZÁLEZ (2019), nos cinco conjuntos de dados mostrados na seção anterior. Além disso, função `erro-fisher` (ver Equação 4.3) proposta na Seção 4.3 foi comparada com a função `erro-abs` (ver Equação 3.1) e com $1 -$ acurácia do modelo no conjunto de validação utilizando os rótulos reais, que será chamada no decorrer deste trabalho de `oráculo`. O valor de acurácia foi utilizado desta forma afim de seguir a estratégia de seleção de hiperparâmetros genérica para LLP (ver Algoritmo 2), que retorna a combinação de hiperparâmetros que minimiza o erro.

Os dados foram divididos entre treino e teste. Todo o processo da escolha de hiperparâmetros utilizando o Algoritmo 2 foi feito no conjunto de treino. Então, a acurácia do melhor modelo vindo desse processo é calculada no conjunto de teste. Seguindo YU et al. (2013), esse processo foi repetido cinco vezes e foram reportados as acurácia médias e seus respectivos desvios padrões. Nos conjuntos de dados propostos por esse trabalho,

[5] <https://www.usatoday.com/story/money/economy/2018/05/10/cost-of-living-value-of-dollar-in-every-state/34567549/>

são gerados cinco conjuntos de teste, um para cada execução, de forma independente, utilizando a mesma distribuição. Do número de instâncias total dos conjuntos de dados *large-toy-llp-10k*, *small-toy-llp-4k* e *small-toy-llp-12k*, 25% delas são referentes ao conjunto de teste.

Já no *mrp-sim-llp-3k* e no *census-2015-llp*, em cada execução foi feito uma divisão aleatória dos dados entre os conjuntos de treino e teste, com 25% dos dados no conjunto de teste. Só que essa divisão aleatória foi feita por *bag*, para garantir que, para cada *bag*, as proporções não mudem entre o conjunto de treino e o conjunto de teste. A ideia é tentar garantir que os dados de treino e de teste tenham aproximadamente a mesma distribuição, assim como nos três conjuntos de dados de referência.

Como já dito anteriormente, o algoritmo utilizado para os experimentos será o alter-$\propto$SVM, por ser um dos estado da arte da literatura de LLP e por ter a facilidade no uso de *kernels* para problemas que não são linearmente separáveis. Seguindo YU et al. (2013), o alter-$\propto$SVM é inicializado aleatoriamente 10 vezes, sendo o resultado do algoritmo a execução com o menor valor da função objetivo (ver Algoritmo 1). Foi utilizada uma implementação própria do algoritmo em Python3 utilizando a implementação do SVM da biblioteca Scikit-Learn (PEDREGOSA et al., 2011).

Vários parâmetros foram testados nos algoritmos de divisão dos dados entre treino e validação, como mostra a lista a seguir:

- *k-fold-LLP* e o k-fold estratificado:
  - Número de partições $F = \{5, 10\}$.
- *Shuffle-Split-LLP* e no *Bootstrapping-Split-LLP*:
  - Número de pares de treino e validação $F = \{5, 10\}$ (Note que o parâmetro F desses algoritmos possui semântica diferente do parâmetro F dos k-folds. Para mais detalhes, ver Algoritmos 3, 4 e 5).
  - Proporção de dados no conjunto de validação $\beta = \{0.2\}$ para o *Shuffle-Split-LLP* e $\beta = \{0.2, 0.5, 0.8\}$ para o *Bootstrapping-Split-LLP*

Já no alter-$\propto$SVM, os parâmetros testados variam por conjunto de dados. Os parâmetros testados são:

- *census-2015-llp*:
  - C e C_p variando de 10^{-1} até 10^5, com incremento de um no expoente.
  - O *kernel* utilizado no SVM foi o linear.
- Demais conjuntos de dados:

- C e C_p variando de 10^{-3} até 10^7, com incremento de um no expoente.
- O *kernel* utilizado no SVM foi o RBF. O valor de γ utilizado foi de $\frac{1}{d \cdot \text{var}(X)}$, onde d é o número de dimensões dos dados e $\text{var}(X)$ é a variância dos dados.

Essa diferença no *census-2015-llp* se dá pelo conjunto de dados ser grande e linearmente separável. Por esses motivos, foram testados menos combinações de hiperparâmetros utilizando o *kernel* linear no SVM.

5.2.2 Desafios experimentais

Durante este trabalho, vários desafios foram enfrentados na parte experimental, principalmente relacionados ao custo computacional envolvido nos experimentos. O problema $\propto$SVM, que é resolvido pelo algoritmo alter-$\propto$SVM, é NP-difícil (YU et al., 2013). Internamente, são resolvidos vários SVMs, que é o passo dentro do laço interno quando o vetor de rótulos é fixado (ver Algoritmo 1). Além disso, o algoritmo é executado 10 vezes para retornar uma solução. Somado a isso, já que todo o processo é não determinístico, todos os experimentos são executados 5 vezes para obter uma acurácia média. Além desses pontos, foram testados vários parâmetros nas variadas técnicas utilizadas nos experimentos.

Para executar os experimentos, foi utilizado o *cluster* de computadores *Shared Computing Cluster* (SCC) da Universidade de Boston[6]. Foram utilizados nós do SCC com 28 cores, que possuem dois processadores de 14 cores. No total, foram utilizadas aproximadamente 20 mil horas de processamento em nós com 28 cores para a execução dos experimentos.

5.3 Resultados

Nesta seção, os resultados dos experimentos seguindo a metodologia mostrada na Seção 5.2.1 serão apresentados. Para facilitar a apresentação e discussão dos resultados, estes serão apresentados separados por conjunto de dados. Foram utilizados gráficos de barras reportando a acurácia média das 5 execuções para cada um dos métodos de divisão dos dados em treino e validação testados. Nos eixo x dos gráficos, `abs` se refere a função `erro-abs` (ver Equação 3.1) e `abs com Fisher` se trata da função `erro-fisher` (ver Equação 4.3). Além disso, foram utilizadas tabelas com as acurácias médias com os respectivos desvios padrões para cada um dos métodos testados.

[6] <http://www.bu.edu/tech/support/research/computing-resources/scc/>

5.3.1 Conjunto de dados *large-toy-llp-10k*

O *large-toy-llp-10k* é um conjunto de dados sintético que emula o cenário de eleições americanas e que satisfaz as propriedades da definição de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2). A Figura 15 e a Tabela 3 mostram os resultados nesse conjunto de dados. É possível notar que nesse conjunto de dados, todos os métodos de divisão dos dados entre treino e validação se saíram melhor em comparação ao k-fold Estratificado, que é a linha de base do trabalho. Note que, mesmo utilizando o `oráculo` o k-fold Estratificado é capaz de superar os outros métodos.

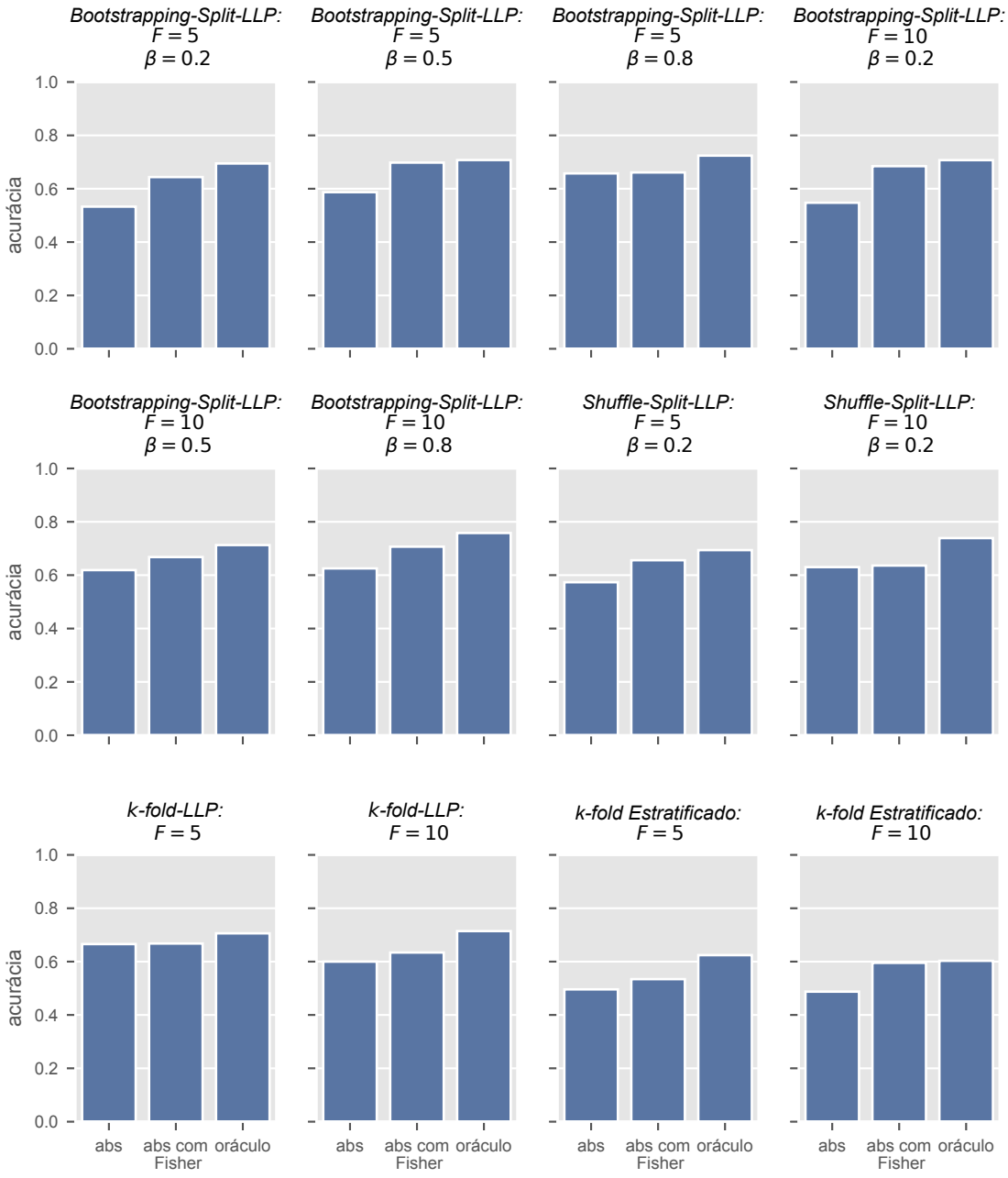


Figura 15 – Resultados no conjunto de dados *large-toy-llp-10k*.

Nesse conjunto de dados, as técnicas de *Bootstrapping-Split-LLP* com $F = 5$, $\beta = 0.5$ e $F = 10$, $\beta = 0.8$ utilizando a função de erro `erro-fisher` obtiveram uma

			`erro-abs`	`erro-fisher`	`oráculo`
Bootstrapping-Split-LLP	$F = 5$	$\beta = 0.2$	0.53 (0.08)	0.64 (0.08)	0.70 (0.03)
		$\beta = 0.5$	0.59 (0.09)	0.70 (0.06)	0.71 (0.02)
		$\beta = 0.8$	0.66 (0.06)	0.66 (0.05)	0.72 (0.05)
	$F = 10$	$\beta = 0.2$	0.55 (0.11)	0.68 (0.03)	0.71 (0.05)
		$\beta = 0.5$	0.62 (0.07)	0.67 (0.09)	0.71 (0.04)
		$\beta = 0.8$	0.53 (0.09)	0.71 (0.07)	0.76 (0.03)
Shuffle-Split-LLP	$F = 5$	$\beta = 0.2$	0.57 (0.10)	0.66 (0.04)	0.70 (0.05)
	$F = 10$	$\beta = 0.2$	0.63 (0.07)	0.63 (0.06)	0.73 (0.05)
k-fold-LLP	$F = 5$		0.67 (0.04)	0.67 (0.05)	0.71 (0.03)
	$F = 10$		0.60 (0.07)	0.64 (0.06)	0.71 (0.06)
k-fold Estratificado	$F = 5$		0.50 (0.03)	0.53 (0.12)	0.62 (0.08)
	$F = 10$		0.49 (0.02)	0.60 (0.11)	0.60 (0.03)

Tabela 3 – Resultados no conjunto de dados *large-toy-llp-10k*.

vantagem nas acurácias médias. Ou seja, nesse caso, amostrar com repetição e utilizar um conjunto de validação maior mostra uma certa vantagem em relação aos outros métodos. A intuição para esse resultado é que amostrar com repetição diminui o erro amostral das *bags* nos conjuntos de treino e validação. Além disso, como os métodos propostos utilizam amostras de todas as *bags*, o modelo é treinado com uma amostra mais representativa do conjunto de dados completo em relação a linha de base, assim obtendo um modelo mais generalizável.

Já se tratando de função de erro, a `erro-fisher` melhora em todos os casos em comparação com a `erro-abs`. Usar a informação de Fisher nesse cenário é interessante pela grande quantidade de *bags* com proporções e tamanhos muito distintos. Neste caso, a função `erro-fisher` pode admitir soluções que erram um pouco mais em *bags* com menos informação, mas que são mais precisar em *bags* com mais informação. Como o número de *bags* é grande, errar em uma *bag* menos informativa não tem um impacto muito grande na acurácia global, mas acertar em uma *bag* mais informativa tem grande impacto.

5.3.2 Conjuntos de dados *small-toy-llp-4k*

O *small-toy-llp-4k* é formado de dados sintético que emula um cenário de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2), mas com menos *bags* e instâncias que o *large-toy-llp-10k*. Os resultados desse conjunto de dados são mostrados na Figura 16 e na Tabela 4. Assim como no *large-toy-llp-10k*, todos os métodos de divisão dos dados entre treino e validação se saíram melhor em comparação ao k-fold Estratificado. Novamente o `oráculo` no k-fold Estratificado teve um desempenho ruim, mostrando que essa forma de dividir os dados não é boa para esse conjunto de dados nem no melhor caso possível.

As técnicas de *Bootstrapping-Split-LLP* com $F = 5$, $\beta = 0.5$ utilizando a função

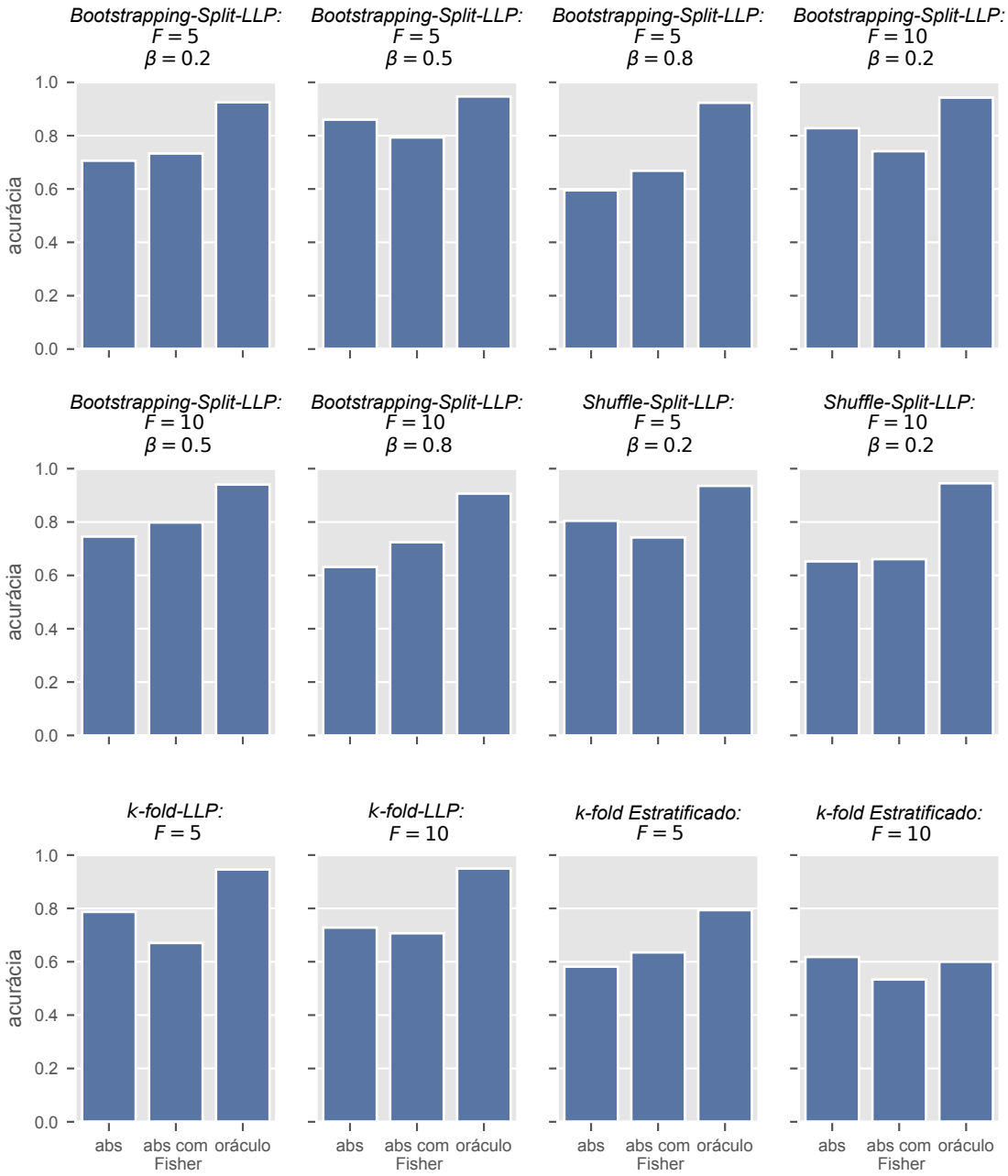


Figura 16 – Resultados no conjunto de dados *small-toy-llp-4k*.

de erro `erro-abs` e $F = 10$, $\beta = 0.5$ utilizando a função de erro `erro-fisher` obtiveram uma vantagem nas acurácias médias. Amostrar com repetição e utilizar um conjunto de validação maior também se mostrou uma estratégia interessante nesse conjunto de dados. A intuição para esse resultado neste conjunto de dados é a mesma do *large-toy-llp-10k*, ou seja, amostrar com repetição diminui o erro amostral das *bags* nos conjuntos de treino e validação e os métodos propostos fornecem uma amostra mais representativa do conjunto de dados completo em relação a linha de base, o que ajuda o modelo a ser mais generalizável.

Na comparação entre funções de erro, olhando a acurácia média, em alguns casos a função `erro-fisher` fica melhor que a função `erro-abs` e em outros não. Uma das

			`erro-abs`	`erro-fisher`	`oráculo`
Bootstrapping-Split-LLP	$F = 5$	$\beta = 0.2$	0.71 (0.13)	0.73 (0.17)	0.93 (0.02)
		$\beta = 0.5$	0.86 (0.15)	0.80 (0.14)	0.94 (0.01)
		$\beta = 0.8$	0.59 (0.01)	0.67 (0.11)	0.92 (0.03)
	$F = 10$	$\beta = 0.2$	0.83 (0.13)	0.74 (0.13)	0.94 (0.01)
		$\beta = 0.5$	0.75 (0.19)	0.80 (0.17)	0.94 (0.01)
		$\beta = 0.8$	0.63 (0.06)	0.72 (0.12)	0.91 (0.04)
Shuffle-Split-LLP	$F = 5$	$\beta = 0.2$	0.80 (0.19)	0.74 (0.19)	0.94 (0.02)
	$F = 10$	$\beta = 0.2$	0.66 (0.17)	0.66 (0.16)	0.95 (0.01)
k-fold-LLP	$F = 5$		0.72 (0.17)	0.67 (0.15)	0.95 (0.01)
	$F = 10$		0.73 (0.17)	0.71 (0.13)	0.95 (0.01)
k-fold Estratificado	$F = 5$		0.58 (0.05)	0.63 (0.13)	0.79 (0.08)
	$F = 10$		0.62 (0.05)	0.53 (0.00)	0.60 (0.18)

Tabela 4 – Resultados no conjunto de dados *small-toy-llp-4k*.

hipóteses para essa oscilação de desempenho da função `erro-fisher` é o número de instâncias e *bags* reduzidos nesse conjunto de dados. Nesse cenário, errar em uma *bag* menos informativa pode ter um impacto relevante na acurácia global.

5.3.3 Conjuntos de dados *small-toy-llp-12k*

O conjunto de dados *small-toy-llp-12k* é sintético e tem como objetivo emular um cenário de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2), sendo gerado da mesma forma que o *small-toy-llp-4k*, com o mesmo número de *bags* mas com mais instâncias. Um efeito que o aumento de instâncias ocasiona é a redução da separação entre rótulos positivos e negativos nas *bags*. A Figura 17 e a Tabela 5 mostram os resultados desse conjunto de dados.

			`erro-abs`	`erro-fisher`	`oráculo`
Bootstrapping-Split-LLP	$F = 5$	$\beta = 0.2$	0.70 (0.14)	0.71 (0.09)	0.93 (0.01)
		$\beta = 0.5$	0.68 (0.15)	0.67 (0.10)	0.92 (0.03)
		$\beta = 0.8$	0.60 (0.01)	0.80 (0.18)	0.92 (0.03)
	$F = 10$	$\beta = 0.2$	0.68 (0.15)	0.71 (0.14)	0.94 (0.01)
		$\beta = 0.5$	0.65 (0.15)	0.68 (0.20)	0.94 (0.01)
		$\beta = 0.8$	0.61 (0.01)	0.88 (0.15)	0.93 (0.02)
Shuffle-Split-LLP	$F = 5$	$\beta = 0.2$	0.71 (0.13)	0.75 (0.13)	0.93 (0.03)
	$F = 10$	$\beta = 0.2$	0.82 (0.14)	0.82 (0.13)	0.94 (0.02)
k-fold-LLP	$F = 5$		0.78 (0.16)	0.80 (0.13)	0.92 (0.04)
	$F = 10$		0.75 (0.13)	0.77 (0.19)	0.94 (0.02)
k-fold Estratificado	$F = 5$		0.66 (0.19)	0.79 (0.19)	0.80 (0.07)
	$F = 10$		0.60 (0.01)	0.57 (0.14)	0.75 (0.15)

Tabela 5 – Resultados no conjunto de dados *small-toy-llp-12k*.

No caso deste conjunto de dados, a função de erro `erro-fisher` melhorou bastante o resultado do k-fold Estratificado com $F = 5$, o deixando com o desempenho muito

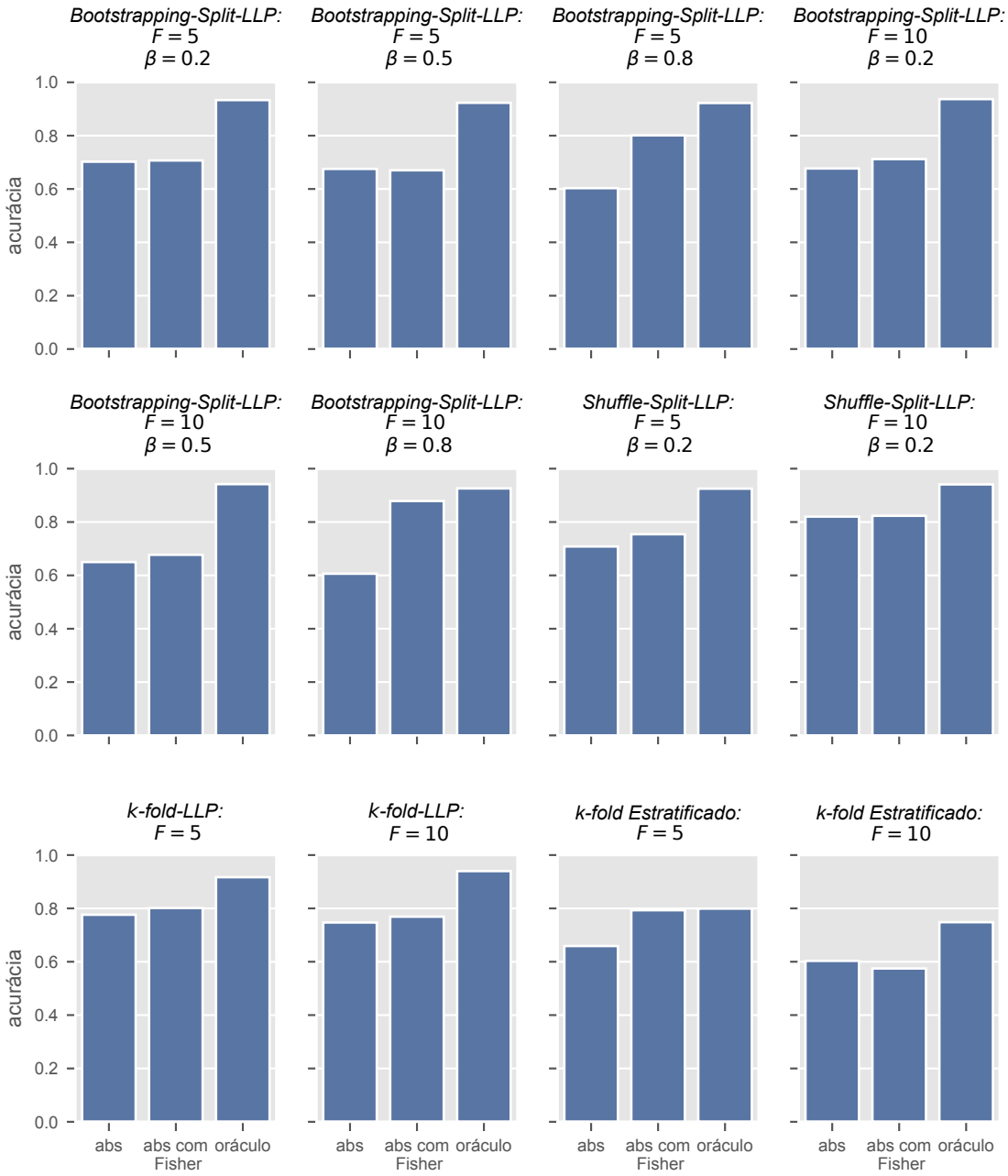


Figura 17 – Resultados no conjunto de dados *small-toy-llp-12k*.

próximo do `oráculo`. Mas as técnicas de *Bootstrapping-Split-LLP* com $F = 5$, $\beta = 0.8$ e $F = 10$, $\beta = 0.8$ utilizando a função de erro `erro-fisher` obtiveram uma vantagem nas acurácias médias. Amostras com repetição utilizando um conjunto de validação maior também mostraram vantagem neste cenário. Além disso, novamente o `oráculo` no k-fold Estratificado teve um desempenho bem abaixo em relação aos outros, mostrando que os conjuntos de treino e validação gerados por essa abordagem não são tão bons. A intuição para esse resultado neste conjunto de dados é a mesma do *small-toy-llp-4k*, já que são conjuntos de dados similares, mas com um número diferente de instâncias. Nesse caso, a melhora ainda é mais significativa, e isso pode ser pelo fato de que o número de instâncias desse conjunto de dados é maior, diminuindo a chance de erro amostral.

A função de erro `erro-fisher` tem um desempenho superior a `erro-abs` em praticamente todos os cenários nesse conjunto de dados. O fato de que as *bags* possuem uma separação menor nesse conjunto de dados provavelmente introduziu mais incerteza nas soluções encontradas pelo algoritmo. Neste cenário de mais incerto, priorizar *bags* informativas no cálculo do erro pode ajudar no desempenho final.

5.3.4 Conjuntos de dados *mrp-sim-llp-3k*

O *mrp-sim-llp-3k* um conjunto de dados é sintético para o problema de MRP, que foi transformado para LLP utilizando o atributo estado como *bag*. Ao contrário dos outros três conjuntos de dados apresentados anteriormente, este não é um conjunto de dados satisfaz as propriedades do Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2). O objetivo é mostrar o funcionamento das abordagens propostas por esse trabalho em cenários diversos. Os resultados desse conjunto de dados são mostrados pela Figura 18 e pela Tabela 6.

			`erro-abs`	`erro-fisher`	`oráculo`
Bootstrapping-Split-LLP	$F = 5$	$\beta = 0.2$	0.65 (0.02)	0.67 (0.03)	0.76 (0.00)
		$\beta = 0.5$	0.63 (0.02)	0.68 (0.02)	0.76 (0.00)
		$\beta = 0.8$	0.64 (0.04)	0.68 (0.04)	0.76 (0.00)
	$F = 10$	$\beta = 0.2$	0.66 (0.02)	0.67 (0.04)	0.76 (0.00)
		$\beta = 0.5$	0.64 (0.02)	0.68 (0.03)	0.76 (0.00)
		$\beta = 0.8$	0.63 (0.03)	0.66 (0.03)	0.76 (0.00)
Shuffle-Split-LLP	$F = 5$	$\beta = 0.2$	0.65 (0.04)	0.66 (0.03)	0.76 (0.00)
	$F = 10$	$\beta = 0.2$	0.64 (0.04)	0.65 (0.02)	0.76 (0.00)
k-fold-LLP	$F = 5$		0.64 (0.03)	0.66 (0.02)	0.76 (0.00)
	$F = 10$		0.62 (0.01)	0.67 (0.03)	0.76 (0.00)
k-fold Estratificado	$F = 5$		0.63 (0.01)	0.66 (0.04)	0.76 (0.00)
	$F = 10$		0.64 (0.02)	0.67 (0.04)	0.76 (0.00)

Tabela 6 – Resultados no conjunto de dados *mrp-sim-llp-3k*.

É possível observar que neste cenário, nenhum método de divisão dos dados entre treino e validação possui vantagem. Isso mostra que o desempenho dos métodos propostos para o cenário de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2) não são prejudicado em cenários de Aprendizado com Proporções de Rótulos Padrão (ver Seção 3.2.1).

Em relação a função de erro, a `erro-fisher` possui uma acurácia média ligeiramente superior que a `erro-abs` em todos os cenários nesse conjunto de dados. Apesar do número de instâncias nesse conjunto de dados ser pequeno, o número de *bags* é grande. Portanto, ser mais preciso em *bags* mais informativas pode ajudar no desempenho como um todo.

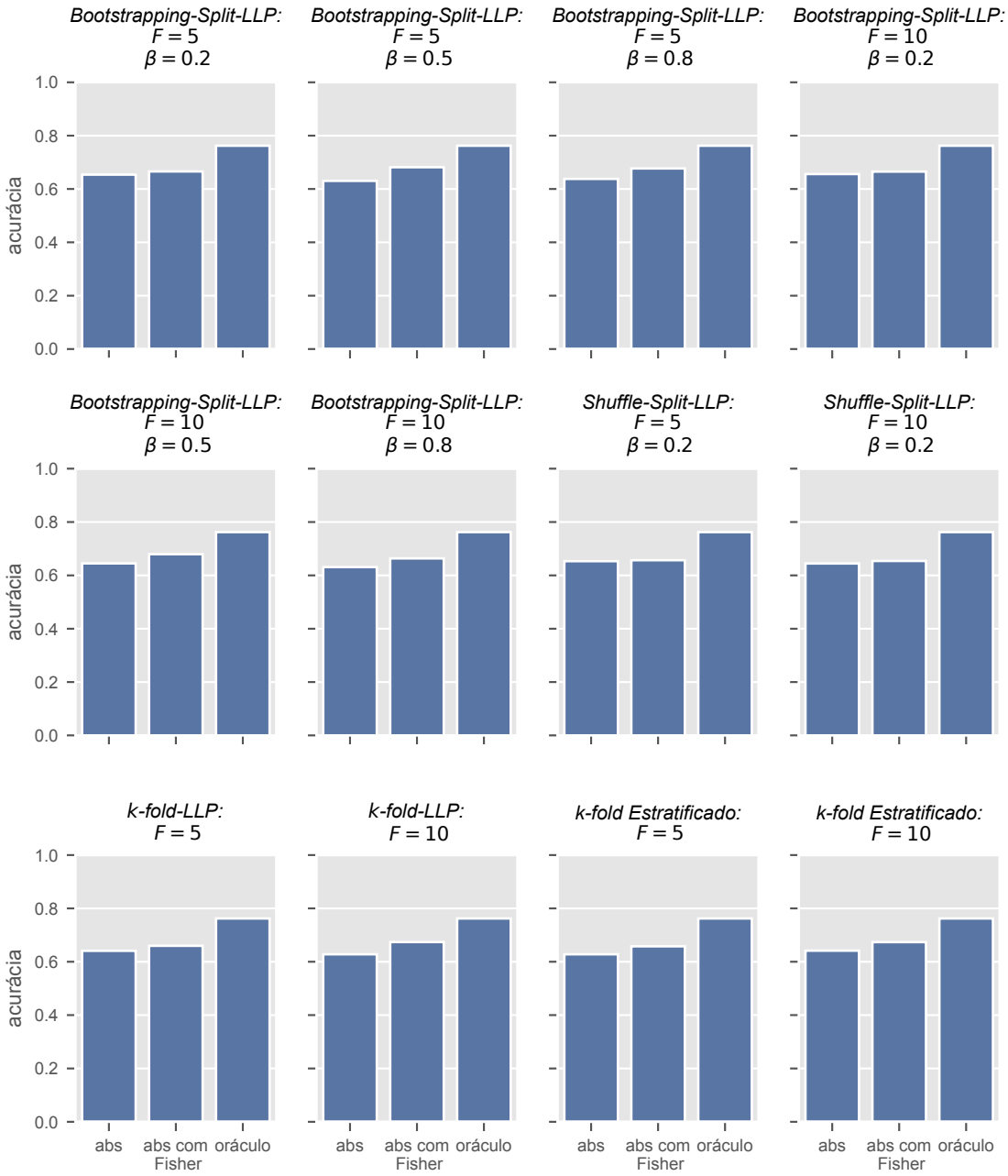


Figura 18 – Resultados no conjunto de dados *mrp-sim-llp-3k*.

5.3.5 Conjuntos de dados *census-2015-llp*

O conjunto de dados *census-2015-llp* é formado por dados do censo americano de 2015 onde cada instância é referente a uma região chamada de *Census Track*. Este conjunto de dados também foi transformado em LLP utilizando o atributo estado como *bag*. Assim como o *mrp-sim-llp-3k*, este conjunto de dados satisfaz as propriedades do Aprendizado com Proporções de Rótulos Padrão (ver Seção 3.2.1), onde métodos mais simples funcionam. Suas instâncias são estatísticas acumuladas de uma certa região, o que dificulta a ocorrência instâncias muito parecidas com rótulos diferentes em *bags* diferentes. Apesar disso, este conjunto de dados é utilizado por se tratar de dados reais nos quais as **bags** são interpretáveis e fazem sentido prático. A Figura 19 e a Tabela 7 mostram os

resultados no *census-2015-llp*.

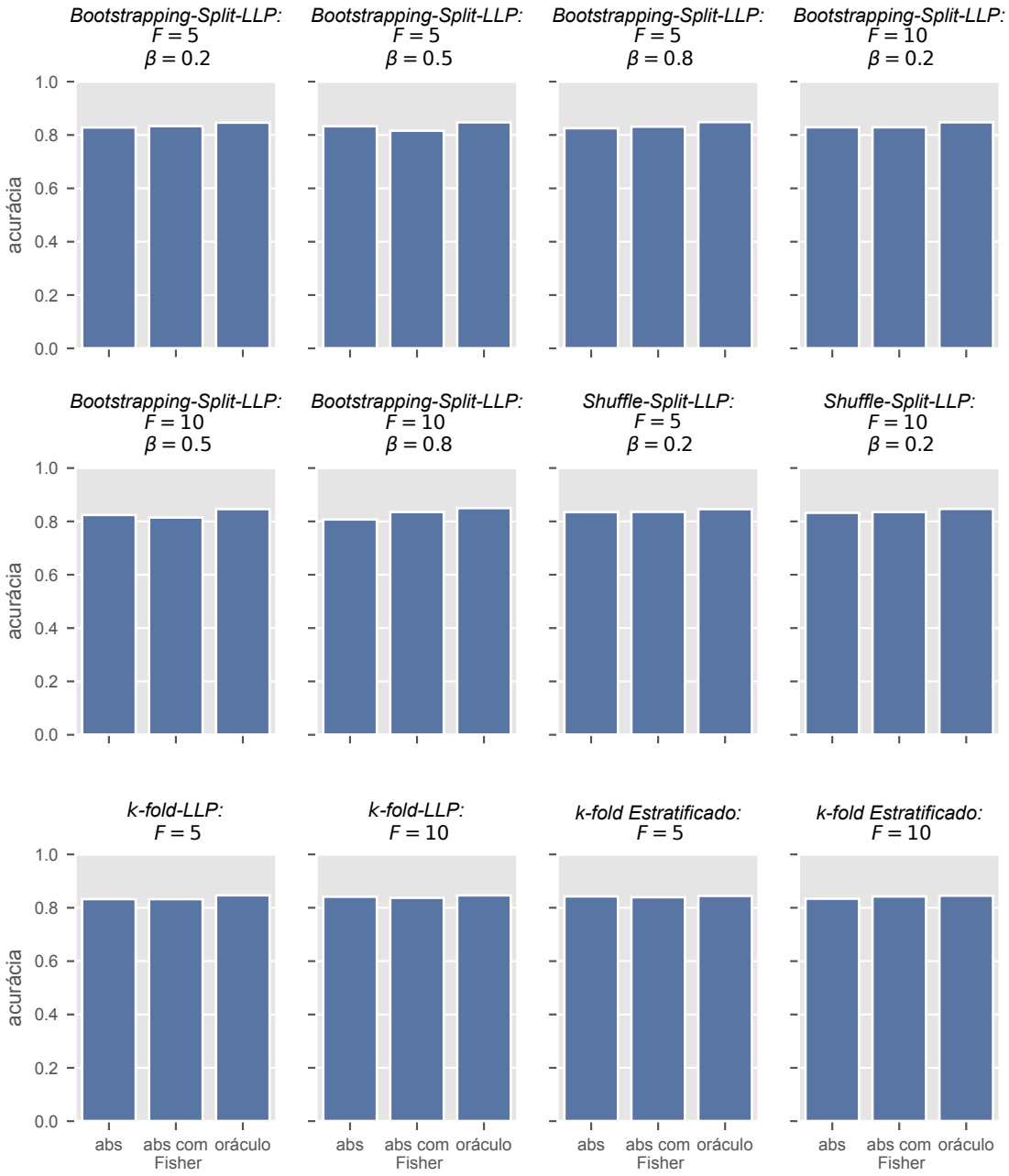


Figura 19 – Resultados no conjunto de dados *census-2015-llp*.

Este é um cenário onde nenhum método de divisão dos dados entre treino e validação e nenhuma função de erro possui grande vantagem, o que reforça que o desempenho dos métodos propostos para cenários de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2) não são prejudicados em cenários de Aprendizado com Proporções de Rótulos Padrão (ver Seção 3.2.1). Também é possível observar que o `oráculo` não possui praticamente nenhuma ganho em relação as outras funções de erro, o que mostra que esse cenário é extremamente fácil para o algoritmo utilizado, que consegue encontrar boas soluções facilmente apesar das dificuldades inerentes do problema de LLP.

			`erro-abs`	`erro-fisher`	`oráculo`
Bootstrapping-Split-LLP	$F = 5$	$\beta = 0.2$	0.83 (0.01)	0.83 (0.00)	0.85 (0.00)
		$\beta = 0.5$	0.83 (0.00)	0.82 (0.01)	0.85 (0.00)
		$\beta = 0.8$	0.83 (0.01)	0.83 (0.01)	0.85 (0.01)
	$F = 10$	$\beta = 0.2$	0.83 (0.01)	0.83 (0.01)	0.85 (0.00)
		$\beta = 0.5$	0.82 (0.02)	0.81 (0.02)	0.85 (0.00)
		$\beta = 0.8$	0.81 (0.02)	0.84 (0.01)	0.85 (0.00)
Shuffle-Split-LLP	$F = 5$	$\beta = 0.2$	0.83 (0.00)	0.84 (0.01)	0.85 (0.00)
	$F = 10$	$\beta = 0.2$	0.83 (0.00)	0.84 (0.01)	0.85 (0.00)
k-fold-LLP	$F = 5$		0.83 (0.01)	0.83 (0.01)	0.85 (0.00)
	$F = 10$		0.84 (0.01)	0.84 (0.01)	0.85 (0.00)
k-fold Estratificado	$F = 5$		0.84 (0.00)	0.84 (0.01)	0.84 (0.00)
	$F = 10$		0.83 (0.01)	0.84 (0.00)	0.85 (0.00)

Tabela 7 – Resultados no conjunto de dados *census-2015-llp*.

5.4 Discussões

Neste capítulo, os conjuntos de dados utilizados para os experimentos foram introduzidos. Então, todo o ambiente experimental foi mostrado em detalhes, para então os resultados serem apresentados. Pelo que foi mostrado, em cenários de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2), existe ganho em utilizar a técnica de *Bootstrapping-Split-LLP* para divisão dos dados entre treino e validação com um valor de β alto, entre 0.5 e 0.8. Uma hipótese para o bom desempenho dessa estratégia é que, como o erro é uma informação relacionada a proporção das *bags*, ter uma boa amostra dos dados por *bag* também no conjunto de validação diminui o erro de amostragem, o que torna erro calculado mais confiável. Além disso, os métodos propostos fornecem uma amostra mais representativa do conjunto de dados completo em relação a linha de base, o que pode ajudar o modelo a ser mais generalizável, especialmente em cenários de Aprendizado com Proporções de Rótulos Geral.

Além disso, utilizar a função de erro `erro-fisher` mostrou certa vantagem em cenários de Aprendizado com Proporções de Rótulos Geral (ver Seção 3.2.2) com muitas instâncias e/ou *bags* e também em um cenário de Aprendizado com Proporções de Rótulos Padrão (ver Seção 3.2.1) mais difícil. Em muitos casos, a melhora de desempenho causada pela `erro-fisher` em relação a `erro-abs` é notável, mostrando que um esquema de pesos mais inteligente pode melhorar ainda mais o desempenho de algoritmos de LLP em cenários gerais.

Em linhas gerais, utilizar o *Bootstrapping-Split-LLP* para divisão dos dados entre treino e validação com um valor de β alto com uma função de erro `erro-fisher` para a seleção de hiperparâmetros em LLP se mostrou a melhor estratégia geral nos conjuntos de dados mostrados.

6 Conclusões

Este trabalho abordou o problema de seleção de hiperparâmetros no problema de Aprendizado com Proporções de Rótulos. A primeira contribuição deste trabalho foi definir formalmente os problemas de Aprendizado com Proporções de Rótulos Padrão e Aprendizado com Proporções de Rótulos Geral. Descrever o problema de LLP Geral formalmente é importante, já que este representa melhor todas as aplicações práticas de LLP.

Com o problema de Aprendizado com Proporções de Rótulos Geral definido, foi proposta uma estratégia genérica para seleção de hiperparâmetros em LLP focada no problema de LLP geral. Para utilizar essa estratégia é necessário definir como será feita a divisão dados entre treino e validação e como será computado o erro de um modelo de LLP no conjunto de validação dada uma certa combinação de hiperparâmetros.

Foram propostas três técnicas para dividir os dados em conjuntos de treino e validação, todas baseadas na divisão dos dados por *bag*. A técnica de *Bootstrapping-Split-LLP*, que faz amostra dos dados com repetição utilizando uma grande proporção dos dados no conjunto de validação, se mostrou a mais eficiente em cenários de Aprendizado com Proporções de Rótulos Geral se comparada com a linha de base. Além disso, uma nova função de erro para modelo de LLP treinado na estratégia de seleção de hiperparâmetros foi proposta. Utilizando a informação de Fisher como inspiração, foi proposto um esquema de pesos para as *bags*, que leva em conta o tamanho da *bag* e sua proporção para quantificar sua informação. Utilizando esse esquema de pesos, foi proposta a função `erro-fisher`, que dá peso maior no cálculo do erro para *bags* mais informativas. Os resultados mostraram que utilizar essa função melhora o desempenho da maioria dos métodos em quase todos os cenários testados.

Foram criados três conjuntos de dados sintéticos que possuem as características do problema de Aprendizado com Proporções de Rótulos Geral para avaliar o desempenho dos métodos propostos neste trabalho. Esses conjuntos de dados podem ser utilizados como referência nas avaliações de métodos de LLP, sendo mais uma contribuição deste trabalho.

As contribuições desse trabalho abrem possibilidade para alguns trabalhos futuros, sendo alguns deles:

- ***Avaliar o desempenho das abordagens propostas em cenários reais de Aprendizado com Proporções de Rótulos:*** neste trabalho foram utilizados conjuntos de dados nos quais os rótulos são conhecidos, com o objetivo de se obter a acurácia final dos modelos, mas isso não é possível em cenários reais de LLP.

Portanto, seria interessante treinar modelos utilizando as técnicas desse trabalho em cenários reais de LLP, como o de eleições mostrado por COMARELA et al. (2018).

- ***Avaliar o desempenho das abordagens propostas utilizando outros algoritmos da literatura:*** existem outros algoritmos para LLP que não foram utilizados nos experimentos realizados neste trabalho. Uma extensão natural seria utilizalos para avaliar como seria o desempenho destes em cenários de Aprendizado com Proporções de Rótulos Geral utilizando os métodos propostos.

- ***Desenvolver e testar métodos para a avaliação de algoritmos em cenários de Aprendizado com Proporções de Rótulos Geral:*** em cenários reais de LLP, os rótulos não são conhecidos, dificultando a avaliação dos modelos treinados. Portanto, criar uma forma de avaliar o desempenho de algoritmos de LLP é essencial em cenários práticos. HERNÁNDEZ-GONZÁLEZ (2019) propôs métricas de avaliação para algoritmos de LLP, mas elas não foram testadas em cenários de Aprendizado com Proporções de Rótulos Geral. Portanto, é possível evoluir mais nessa parte, definindo melhor as diretivas para a avaliação de desempenho dos modelos treinados em cenários de LLP geral.

Referências

ARDEHALY, E. M.; CULOTTA, A. Domain adaptation for learning from label proportions using self-training. In: **IJCAI**. [S.l.: s.n.], 2016. p. 3670–3676.

ARDEHALY, E. M.; CULOTTA, A. Co-training for demographic classification using deep learning from label proportions. In: IEEE. **2017 IEEE International Conference on Data Mining Workshops (ICDMW)**. [S.l.], 2017. p. 1017–1024.

CHEN, Z.; CHEN, W.; SHI, Y. Ensemble learning with label proportions for bankruptcy prediction. **Expert Systems with Applications**, Elsevier, v. 146, p. 113155, 2020.

CHEN, Z. et al. Learning with label proportions based on nonparallel support vector machines. **Knowledge-Based Systems**, Elsevier, v. 119, p. 126–141, 2017.

COMARELA, G. et al. Assessing candidate preference through web browsing history. In: **Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. New York, NY, USA: Association for Computing Machinery, 2018. (KDD '18), p. 158–167. ISBN 9781450355520. Disponível em: <https://doi.org/10.1145/3219819.3219884>.

EFRON, B. Bootstrap methods: another look at the jackknife. In: **Breakthroughs in statistics**. [S.l.]: Springer, 1992. p. 569–593.

FISH, B.; REYZIN, L. On the complexity of learning from label proportions. In: **IJCAI**. [S.l.: s.n.], 2017. p. 1675–1681.

FRIEDEN, B. R. **Science from Fisher information**. [S.l.]: Citeseer, 2004. v. 974.

GOODRICH, B. et al. **rstanarm: Bayesian applied regression modeling via Stan.** 2020. R package version 2.21.1. Disponível em: <https://mc-stan.org/rstanarm>.

HANRETTY, C. An introduction to multilevel regression and post-stratification for estimating constituency opinion. **Political Studies Review**, SAGE Publications Sage UK: London, England, v. 18, n. 4, p. 630–645, 2020.

HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. **The elements of statistical learning: data mining, inference, and prediction**. [S.l.]: Springer Science & Business Media, 2009.

HERNÁNDEZ-GONZÁLEZ, J. A framework for evaluation in learning from label proportions. **Progress in Artificial Intelligence**, Springer, p. 1–15, 2019.

KÜCK, H.; FREITAS, N. de. Learning about individuals from group statistics. In: **Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence**. [S.l.: s.n.], 2005. p. 332–339.

LIU, J. et al. Learning from label proportions with generative adversarial networks. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2019. p. 7167–7177.

METSIS, V.; ANDROUTSOPOULOS, I.; PALIOURAS, G. Spam filtering with naive bayes-which naive bayes? In: MOUNTAIN VIEW, CA. **CEAS**. [S.l.], 2006. v. 17, p. 28–69.

MITCHELL, T. M. **Machine Learning**. 1. ed. USA: McGraw-Hill, Inc., 1997. ISBN 0070428077.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of machine learning**. [S.l.]: MIT press, 2018.

PARK, D. K.; GELMAN, A.; BAFUMI, J. Bayesian multilevel estimation with poststratification: State-level estimates from national polls. **Political Analysis**, JSTOR, p. 375–385, 2004.

PATRINI, G. et al. (almost) no label no cry. In: **Advances in Neural Information Processing Systems**. [S.l.: s.n.], 2014. p. 190–198.

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

POYIADZI, R.; SANTOS-RODRIGUEZ, R.; TWOMEY, N. Label propagation for learning with label proportions. In: IEEE. **2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)**. [S.l.], 2018. p. 1–6.

QI, Z. et al. Adaboost-llp: a boosting method for learning with label proportions. **IEEE transactions on neural networks and learning systems**, IEEE, v. 29, n. 8, p. 3548–3559, 2017.

QI, Z. et al. Learning with label proportions via npsvm. **IEEE transactions on cybernetics**, IEEE, v. 47, n. 10, p. 3293–3305, 2016.

QUADRIANTO, N. et al. Estimating labels from label proportions. **Journal of Machine Learning Research**, v. 10, n. Oct, p. 2349–2374, 2009.

REN, Q.; LI, M.; HAN, S. Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives. **Big Earth Data**, Taylor & Francis, v. 3, n. 1, p. 8–25, 2019.

RUEPING, S. Svm classifier estimation from group probabilities. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. Madison, WI, USA: Omnipress, 2010. (ICML'10), p. 911–918. ISBN 9781605589077.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 3rd. ed. USA: Prentice Hall Press, 2009. ISBN 0136042597.

SHI, Y. et al. Learning from label proportions with pinball loss. **International Journal of Machine Learning and Cybernetics**, Springer, v. 10, n. 1, p. 187–205, 2019.

SHI, Y.; LIU, J.; QI, Z. Inverse convolutional neural networks for learning from label proportions. In: IEEE. **2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)**. [S.l.], 2018. p. 643–646.

SHI, Y. et al. Learning from label proportions on high-dimensional data. **Neural Networks**, Elsevier, v. 103, p. 9–18, 2018.

SZEGEDY, C.; TOSHEV, A.; ERHAN, D. Deep neural networks for object detection. In: **NIPS**. [s.n.], 2013. p. 2553–2561. Disponível em: <http://papers.nips.cc/paper/5207-deep-neural-networks-for-object-detection>.

YU, F. X. et al. On learning from label proportions. **arXiv preprint arXiv:1402.5902**, 2014.

YU, F. X. et al. ∝svm for learning with label proportions. In: JMLR. ORG. **Proceedings of the 30th International Conference on International Conference on Machine Learning-Volume 28**. [S.l.], 2013. p. III–504.

ZAKI, M. J.; MEIRA JR, W. **Data Mining and Analysis: Fundamental Concepts and Algorithms**. [S.l.]: Cambridge University Press, 2014.