CLASSIFICADORES DE ALTA INTERPRETABILIDADE E DE ALTA PRECISÃO

ITAMAR HATA

CLASSIFICADORES DE ALTA INTERPRETABILIDADE E DE ALTA PRECISÃO

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Adriano Veloso
Coorientador: Nivio Ziviani

Belo Horizonte
Dezembro de 2013

Hata, Itamar

H361c Classificadores de alta interpretabilidade e de alta precisão / Itamar Hata. — Belo Horizonte, 2013
xxii, 60 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Adriano Veloso Coorientador: Nivio Ziviani

1. Computação - Teses. 2. Aprendizado do computador. I. Título.

CDU 519.6*82 (043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Classificadores de alta precisão e alta interpretabilidade

ITAMAR SAKAE VIANA HATA

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. ADRIANO ALONSO VELOSO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. NIVIO ZIVIANI - Coorientador
Departamento de Ciência da Computação - UFMG

PROF. BERTHIER RIBEIRO DE ARAÚJO NETO
Departamento de Ciência da Computação - UFMG

PROF. RENATO ANTÔNIO CELSO FERREIRA
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 12 de dezembro de 2013.

Agradecimentos

Nem sempre os caminhos percorridos foram fáceis ou certos. Para vencer os obstáculos, suportar as dificuldades e me aconselhar a tomar as decisões corretas, contei com a ajuda de várias pessoas que admiro e considero como amigos.

Gostaria de começar agradecendo à minha mãe Adriene Hata, que nunca deixou de acreditar em mim, ao meu pai Roberto Hata e aos meus irmãos Helena Hata e William Hata, que estão sempre ao meu lado. Agradeço à minha noiva e companheira Ana Paula Barros, super fofinha, que apoiou minhas decisões e esteve sempre comigo, não se importando em perder alguns finais de semana e feriados enquanto eu estudava (de tanto ouvir assuntos de informática, tornou-se uma exímia cientista da computação). Eu não teria entrado na faculdade se não fosse por minha tia Darclé Viana, que acreditou em mim. Também tive um apoio essencial dos meus tios Elizabeth Viana e José Arnaldo e das minhas primas Daniele Viana, Érica Viana e Helen Viana. No pré-vestibular, tive uma grande ajuda do professor Alisson Marques, que me deu aulas particulares gratuitas de matemática, e do professor de física Jaques Braga, que sempre me incentivava a estudar mais. Agradeço à minha avó Helena Viana e a Solanda Teixeira pelas orações que fizeram por mim.

Gostaria de agradecer também ao meu primeiro orientador de pesquisa Renato Ferreira, que foi fundamental para minha entrada no mestrado, e a Jussara Almeida, Ítalo Cunha e Virgílio Almeida, meus orientadores durante a maior parte da graduação. Trabalhar com eles foi essencial na minha formação. Entre diversas outras coisas, eles me ensinaram a ter disciplina e dedicação e a conduzir um projeto metodologicamente, seja na pesquisa ou no trabalho.

A graduação não foi fácil, mas sempre pude contar com o apoio e as dicas de colegas como Carlos Souza, Daniel Chaltein, Dineu Assis, Fábio Miranda, Guilherme Pimenta, Luiz Felipe Viana, Paulo Roberto, Rafael Almeida e Rodrigo Keller, além dos colegas do laboratório VoD Alex Borges, Cristiano Costa, Eduardo Colaço, Fabiano Belém e Flávio Vinícius e colegas do laboratório Speed Arlei Silva, Bruno Coutinho, Charles Ferreira, Claudiane Rodrigues, Carlos Teixeira, Luam Totti, Leonardo Mata, Pedro Calais, Rodrigo Oliveira, Thatyene Oliveira e Walter Filho.

Na graduação, uma fase marcante foram minhas participações nas Maratonas de Programação. Fiz amigos de grande valia nesse evento, aprendi a trabalhar em equipe, me diverti

e aprimorei meus conhecimentos em algoritmos. O aniMOUSEs, nosso time da maratona, era composto por Leonardo Conegundes, Felipe Machado, por mim e, no último ano, pelo Thiago Goulart. Devo muito aos integrantes do aniMOUSEs. Além deles, outras pessoas faziam parte da maratona e também foram importantes na minha trajetória, entre elas estão Alexandre Davis, Diego da Hora, Diego Rennó, Dilson Guimarães, Erickson Nascimento, Fabrício Benevuto, Fernando Duarte, Fernando Fernandes, Filipe Arcanjo, Frederico Quintão, Giselle Reis, Henrique Pinto, João Palotti, Lídio Ramalho, Nilson Figueiredo e Thiago Noronha.

Agradeço aos colegas que fizeram parte do meu dia a dia no mestrado Afonso Sampaio, Alex Guimarães, Carlos Caetano, Harlley Lima, Sara Guimarães, Sílvio Soares e Vítor Sales e aos colegas do LATIN, laboratório no qual fiz pesquisa, Adolfo Guimarães, Aécio Santos, Aline Bessa, Arthur Câmara, Cristiano Carvalho, Leonardo Santos, Leonardo Vilela, Marco Túlio, Sabir Ribas, Rodrygo Santos, Thales Costa, Wladmir Brandão e, em especial, ao Anísio Lacerda, que sentava ao meu lado e me ajudava diariamente. Agradeço também aos colegas do grupo de pesquisa de aprendizado de máquina LAMA Adriano Pereira, Antônio Carlos, Bruna Neuenschwander, Carlos Alessandro, Gabriel Carvalho, Isabella Brito, Mariane Souza e, em especial, ao Roberto Oliveira, que foi um grande parceiro durante o mestrado. Agradeço aos amigos Jean Freire, Lucas Castro, Lucas Eustáquio, Paulo Bicalho, Stella Lara e Thássia Almeida, que discutiam diariamente comigo coisas sobre a vida, o universo e tudo mais. Agradeço aos professores Alberto Laender, Antônio Otávio, Berthier Ribeiro, Dorgival Neto, Giselle Pappa, Marcos Gonçalvez, Mirella Mouro, Renato Assunção, Roberto Bigonha, Rodolfo Resende, Sérgio Campos e Wagner Meira, que foram fundamentais na minha formação como cientista da computação, desde da época da graduação.

Gostaria de agradecer aos meus orientadores Adriano Veloso e Nivio Ziviani. Este trabalho só foi possível graças a eles. No entanto, as contribuições deles vão além das orientações para realização deste trabalho, pois elas foram essenciais na minha formação como mestre em ciência da computação. Os dois são profissionais inteligentes e experientes que utilizei como referência na minha trajetória. Eles me transmitiram conhecimentos valiosos (ensinamentos que perdurarão pelo resto da vida), me guiaram na tomada de decisões e me incentivaram nos momentos difíceis.

Por fim, gostaria de agradecer à banca examinadora e às pessoas que revisaram esta dissertação, pelo tempo dedicado e pelas dicas valiosas: Adriano Veloso, Adriene Hata, Ana Paula Barros, Anísio Lacerda, Berthier Ribeiro, Izabela Maffra, Nivio Ziviani, Paulo Bicalho, Renato Ferreira, Roberto Oliveira, Stella Lara e Thales Costa.

"You can, for example, never foretell what any one man will do, but you can say with precision what an average number will be up to. Individuals vary, but percentages remain constant. So says the statistician. "

(Sir Arthur Conan Doyle, Sherlock Holmes, *The Sign of Four*)

Resumo

Na construção de uma aplicação de aprendizado de máquina, um especialista define seu objetivo, determina, baseado em algumas hipóteses, os dados que possuem uma relação causal com o objetivo, seleciona o modelo que melhor se adequa a suas hipóteses e dados, realiza experimentos e analisa a qualidade da solução. Na fase de análise, uma propriedade fundamental do modelo é a interpretabilidade. Em alguns domínios de aplicação, como médica ou de negócios, a interpretabilidade é tida como diferencial. Para que um modelo seja interpretável é indicado que ele possua poucos atributos e siga o princípio da parcimônia, no qual, dentre explicações equivalentes, as mais simples são preferíveis. Esse princípio tem se mostrado adequado para classificadores baseados em regras de associação, nos quais a quantidade de regras utilizadas podem ser substancialmente reduzidas utilizando-se representações condensadas, como os conjuntos máximos ou fechados. Porém, a quantidade restante de regras ainda é grande, sendo os modelos resultantes de difícil interpretabilidade. Neste trabalho, propomos uma estratégia de redução mais agressiva, que ao mesmo tempo mantém a acurácia do classificador. Essa estratégia consiste em avaliar cada regra sob um critério estatístico e filtrar as melhores. Cada regra representa um ponto num espaço no qual cada dimensão é representada por um critério estatístico. Pontos que não são dominados por nenhum outro compõem a fronteira de Pareto e correspondem às regras que são ótimas, dado que não existe regra que possa ser melhor do que elas nos critérios selecionados. Um conjunto sistemático de experimentos envolvendo dados de referência, assim como dados de aplicações atuais, seguido de um conjunto extensivo de testes de significância revelam que a estratégia proposta foi capaz de reduzir em até duas ordens de grandeza (reduções de até 96%) a quantidade de regras utilizadas na classificação, produzindo classificadores mais interpretáveis, sem prejudicar a acurácia dos modelos.

Palavras-chave: Aprendizado de Máquina, Interpretável, Parcimônia, Pareto.

Abstract

Building a Machine Learning application typically requires an expert who defines the objectives and data that have a causal relationship with these objectives, selects the best model that fits the assumptions and data, conducts some experiments and analyses the quality of the solution. In the analysis phase, a fundamental property of the model is its interpretability. In some application domains, such as medical or business, the interpretability is taken as a differential solution. To build an interpretable model, it is recommended the use of few features within the parsimony principle, which states that everything being equal, simpler explanations are preferable. Recently, this principle has shown to be well-suited to associative classifiers, where the number of rules composing the classifier can be substantially reduced by using condensed representations such as maximal or closed rules. However, the remaining amount of rules is still large, and the resulting models are hard to interpret. In this work we propose a more aggressive filtering strategy, which decreases the number of rules within the classifier without hurting its accuracy. Our strategy consists in evaluating each rule under different statistical criteria, and filtering only those rules that show a positive balance between all the criteria considered. Specifically, each candidate rule is associated with a point in an n-dimensional scatter-gram, where each coordinate corresponds to a statistical criterion. Points that are not dominated by any other point in the scatter-gram compose the Pareto frontier, and correspond to rules that are optimal in the sense that there is no rule that is better off when all the criteria are taken into account. Finally, rules lying in the Pareto frontier are filtered and compose the classifier. A systematic set of experiments involving benchmark data as well as recent data from actual application scenarios, followed by an extensive set of significance tests, reveal that the proposed strategy decreases the number of rules by up to two orders of magnitude and produces classifiers that are more readable without hurting accuracy.

Keywords: Machine Learning, Interpretable, Parsimony, Pareto.

Lista de Figuras

Lista de Tabelas

Lista de Siglas

y_i	uma saída ou rótulos arbitrários.
$\mathcal{Y}$	conjunto de saída.
X	conjunto de entrada.
$\mathcal{X}$	subconjunto de atributos.
$\mathcal{X} \rightarrow y_i$	uma regra de decisão arbitrária.
$\mathcal{P}$	uma distribuição de probabilidade.
$\mathcal{S}$	conjunto de treinamento.
$\mathcal{T}$	conjunto de teste.
$\mathcal{F}_s$	uma aproximação de $\mathcal{P}$ utilizando $\mathcal{S}$.
$\mathcal{D}$	uma base de dados, $\mathcal{D} = \mathcal{S} \cup \mathcal{T}$.
$\boldsymbol{t}(\mathcal{X})$	conjunto de transações de $\mathcal{D}$ em que $\mathcal{X}$ ocorre.
$\mathcal{I}$	conjunto de itens presentes em $\mathcal{D}$.
$\mathcal{CR}$	um critério estatístico.
$D^{\mathcal{CR}}$	espaço de utilidade multicritério.
$\mathcal{R}$	conjunto de regras utilizadas por um algoritmo associativo.
$\mathcal{R}^x$	conjunto de regras utilizadas para classificar uma instância x.
η	número de Fronteiras de Pareto a serem utilizadas no filtro.
MDL	descrição de tamanho mínima (*Minimum Description Length*).
LAC	Classificação Associativa Sob Demanda - *Lazy Associative Classification*.
PE-LAC	LAC utilizando a fronteira de Pareto como filtro de regras.
CL-LAC	LAC utilizando o conjunto fechado das regras de associação.
CL-PE-LAC	LAC utilizando a fronteira de Pareto em cadeia com o conjunto fechado.

Sumário

Capítulo 1

Introdução

Em aprendizado de máquina, existe uma dicotomia entre modelos precisos e interpretáveis. Para que um modelo seja interpretável, é desejável que ele seja uma representação fiel do problema, de modo que suas decisões possam ser explicadas pelas hipóteses e dados. Além disso, devido a fatores cognitivos dos humanos, o modelo deve conter poucas variáveis e possuir alguma representação visual (Vellido et al. [2012]). Na busca por acurácia, existe uma tendência em abrir mão da interpretabilidade. Especialistas utilizam atributos que não possuem relação clara com o objetivo, utilizam centenas de atributos e criam modelos complexos de interpretar, a fim de maximizar a acurácia. Analisando uma aplicação de aprendizado de máquina, percebe-se que sua interpretabilidade é fundamental, tanto na etapa de construção por um especialista quanto na etapa de utilização, para que um usuário possa compreender porque foi tomada determinada decisão. A construção de uma aplicação de aprendizado pode ser dividida em sete fases, como ilustrado na figura 1.1. Na primeira fase, devem ser definidos os objetivos e o escopo da aplicação e, na última fase, deve ser analisado se o sistema atual alcança esses objetivos. Caso ele não os alcance, deve-se recomeçar a partir da fase 1 refinando o problema e definindo novos objetivos. Normalmente, um especialista seleciona os atributos que possuam uma relação causal com o resultado do modelo baseando-se em algumas hipóteses. Na fase 5, o especialista seleciona o modelo que melhor se adequa a suas hipóteses e dados. Na fase 6, um conjunto de experimentos é definido e executado. Na fase 7, ele analisa os resultados. Para que o especialista seja capaz de avaliar o comportamento do modelo escolhido diante de suas hipóteses e dos dados, é fundamental que o modelo seja fácil de interpretar.

Em alguns domínios de aplicação, como médica ou de negócios, a interpretabilidade é um requisito fundamental. Por exemplo, em diagnóstico de câncer do pulmão, o médico deve saber quais os atributos foram decisivos na predição (Mramor et al.

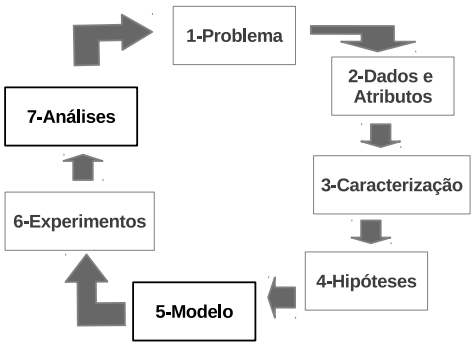


Figura 1.1: Ciclo de construção de uma aplicação de aprendizado de máquina.

[2007]). A maioria dos sistemas médicos são desenvolvidos com foco em interpretabilidade, utilizam poucos atributos e não são necessariamente otimizados para obter a melhor acurácia. Exemplos de sistemas médicos interpretáveis bastante utilizados são o TIMI Score (MD et al. [2000]) para detecção de infarto no miocárdio, o CURB-65 (Lim et al. [2003]) para previsão de morte por pneumonia e o CHADS2 (Gage et al. [2001]) para detecção de derrame cerebral a partir da medida de fibrilação atrial (frequência cardíaca irregular e rápida). Outras aplicações nas quais a interpretabilidade do modelo é fundamental incluem detecção de fraudes, análise de crédito e análise de impacto de uma campanha de marketing. Nessas aplicações, especialistas tendem a preferir modelos mais transparentes (Letham et al. [2012, 2013]).

Porém, historicamente, o objetivo dos algoritmos de classificação tem sido maximizar a acurácia geralmente produzindo classificadores difíceis de interpretar, como KNNs (Cover & Hart [1967]), SVMs (Cortes & Vapnik [1995]; Joachims [2006]) e Naive Bayes (Domingos & Pazzani [1997]; Lowd & Domingos [2005]). Atualmente, classificadores interpretáveis são representados principalmente por árvores de decisão (Breiman et al. [1984]; Gehrke et al. [1999]) e classificadores associativos (Liu et al. [1998]). Entretanto, modelar problemas com bases de dados grandes e complexas utilizando estruturas de árvores ou de regras de associação, normalmente, resulta em modelos grandes e difíceis de interpretar.

Neste trabalho, teremos como base classificadores associativos, sendo nosso objetivo reduzir o máximo possível a quantidade de regras utilizadas por estes modelos sem que seja afetada a acurácia. Nossa estratégia consiste em determinar a utilidade (ou eficiência) de cada regra utilizando diferentes critérios estatísticos de regras de associação. Os critérios utilizados, explicados com detalhes na seção 2.1.1, foram: suporte, confiança, valor adicionado e Yules'Q (Tan et al. [2002]). Intuitivamente, queremos selecionar as regras que apresentam os melhores compromissos diante dos critérios. A

solução proposta consiste na filtragem de regras de associação que irão compor o modelo, com base no conceito de eficiência de Pareto (Palda [2011]). Esse é um conceito central na economia, desenvolvido pelo italiano Vilfredo Pareto no século XIX. Dado um grupo de pessoas, uma transação realizada entre elas é considerada uma melhoria de Pareto se pelo menos um indivíduo aumentou sua utilidade total e a utilidade dos outros envolvidos não sofreu redução. O grupo de pessoas irá alcançar um estado Pareto-Ótimo, ou Pareto-Eficiente, quando não for possível realizar mais nenhuma melhoria de Pareto, ou seja, se não for possível realizar outra transação sem degradar a utilidade de algum envolvido. O mesmo conceito pode ser explorado para selecionar regras de classificação eficientes. Nesse caso, cada regra representa um ponto em um espaço, no qual cada dimensão é representada por um critério estatístico. Pontos que não são dominados por nenhum outro compõem a **fronteira de Pareto** e correspondem as regras que são ótimas, dado que não existe regra que possa ser melhor do que elas nos critérios selecionados (Corne et al. [2000]). As regras da fronteira de Pareto são selecionadas pelo nosso filtro e compõem o classificador final.

Foram realizados um conjunto de experimentos envolvendo bases de referência do repositório da Universidade da Califórnia, Irvine (UCI, Newman et al. [1998]), assim como dados de aplicações reais de análise de sentimentos (Pak & Paroubek [2010]; Silva et al. [2011a]). Nos resultados, a estratégia de filtro proposta, baseada no conjunto ótimo de regras multicritérios, foi capaz de reduzir em até duas ordens de grandeza (reduções de até 96%) a quantidade de regras utilizadas na classificação, produzindo classificadores muito mais interpretáveis. Além disso, uma bateria de testes de significância indica que a acurácia obtida pelos classificadores mais simples gerados pela nossa estratégia é estatisticamente equivalente à acurácia obtida pelos classificadores mais complexos. Como resultado, os classificadores continuaram precisos, porém, bem mais legíveis.

1.1 Objetivo

O objetivo deste trabalho consiste em melhorar a interpretabilidade dos classificadores baseados em regras de associação, sem prejudicar sua acurácia. Escolhemos utilizar classificadores baseados em regras, uma vez que as regras de associação são uma forma de representação natural de padrões pelos humanos. Além disso, esses classificadores estão entre o estado da arte de algoritmos de aprendizado de máquina. Porém, apesar de precisos, a quantidade de regras geradas por eles tende a ser grande, o que dificulta sua legibilidade. Com isso, para aumentar a legibilidade destes modelos,

propomos um método de filtragem para reduzir o máximo possível o tamanho do seu conjunto de regras sem que seja prejudicado seu poder de generalização.

1.2 Contribuições

As principais contribuições deste trabalho são:

- Elaboração de um filtro de regras de associação para reduzir a quantidade de regras e tornar os modelos mais legíveis. O filtro proposto segue o princípio da navalha de Occam (Blumer et al. [1987]; Domingos [1999]; Zahálka & Zelezný [2011]), pois ele gera classificadores mais simples e com acurácia equivalente.
- Elaboração de uma estratégia para combinar vários critérios estatísticos utilizando um espaço de utilidade. Trabalhos anteriores utilizam um critério para avaliar a importância das regras.
- Realização de um conjunto extensivo de experimentos comparando os modelos que utilizam: 1) todas as regras acima de um suporte mínimo, 2) o conjunto fechado das regras e 3) nossa estratégia de filtro de Pareto. Este foi o primeiro trabalho a estudar a qualidade do conjunto fechado de regras em um classificador sob demanda (*lazy*).
- Fortalecimento da importância da interpretabilidade na tarefa de aprendizado de máquina.
- Disponibilização de implementações de referências dos algoritmos e das bases de dados utilizados neste trabalho no Repositório de Aprendizado de Máquina da UFMG (Veloso & Dafe [2012]).

As principais contribuições deste trabalho foram apresentadas no periódico *Journal of Information and Data Management* (Hata et al. [2013]). Além disso, a estratégia de utilizar a fronteira de Pareto para encontrar as melhores soluções foi utilizada como contribuição em um algoritmo de sistemas de recomendação aceito para publicação na *Transactions on Intelligent Systems and Technology* (Ribeiro et al. [2014]).

1.3 Organização do texto

O texto está organizado da seguinte forma: no capítulo 2, serão introduzidos os conceitos fundamentais utilizados neste trabalho e será apresentada uma revisão da bibliografia relevante ao tema; no capítulo 3, será explicada nossa estratégia de filtro

de regras; no capítulo 4, será mostrado um conjunto de resultados de experimentos que foram realizados para avaliar a eficácia da nossa estratégia; por fim, no capítulo 5, são apresentadas nossas principais conclusões e os trabalhos futuros.

Capítulo 2

Referencial Teórico e Trabalhos Relacionados

Neste capítulo, são introduzidos os conceitos fundamentais de regras de associação, classificação associativa, interpretabilidade e uma revisão da bibliografia relevante ao tema.

2.1 Regras de Associação

Uma relação entre conjuntos pode ser representada por uma regra de associação do tipo $\mathcal{X} \rightarrow y_i$. No caso do aprendizado de máquina, o conjunto antecedente/independente $\mathcal{X}$ é representado por um subconjunto de itens $\mathcal{I} = \{i_1, i_2, ..., i_m\}$, discretos e mutuamente exclusivos, e o consequente/dependente y_i é um item do conjunto de rótulos ou classes $\mathcal{Y}$. Dado um indivíduo a ser classificado, os itens são as informações observáveis e os rótulos são o conjunto de categorias a que um indivíduo pode pertencer. Um conjunto de itens $\mathcal{X}$ de tamanho k é chamado de *k-itens*.

Estratégias baseadas em regras de associação têm sido amplamente utilizadas uma vez que as regras representam uma forma natural de expressar possíveis relações de causa e consequência. Alguns trabalhos que utilizaram regras de associação com sucesso incluem aplicações em recuperação da informação (Pôssas et al. [2005]; Silva et al. [2011b]), mineração de dados (Agrawal et al. [1993]; Zaki & Gouda [2003]) e aprendizado de máquina (Jovanoski & Lavrac [2001]; Veloso & Meira [2011]).

Porém, a quantidade de regras possíveis de serem formadas a partir de um conjunto de atributos é exponencial e determinar quais são as regras relevantes é um desafio. Por isso, critérios estatísticos para determinar quais regras são relevantes e algoritmos para geração de regras de forma eficiente devem ser utilizados. A seguir,

explicaremos alguns critérios estatísticos, alguns algoritmos para geração de regras e como as regras de associação podem ser utilizadas na tarefa de classificação.

2.1.1 Critérios Estatísticos

Os critérios estatísticos são medidas criadas com a finalidade de avaliar a qualidade das regras. Eles têm sido utilizados para descartar regras que não satisfazem um limiar mínimo (Lenca et al. [2008]; Letham et al. [2012]) ou para ponderar as regras, de acordo com sua utilidade, com a finalidade de ordená-las ou de estimar seu impacto em uma determinada causa (Vaillant et al. [2006]).

Os algoritmos baseados em regras de associação, em particular, necessitam de alguma métrica para avaliar a utilidade da regra gerada e, com isso, decidir qual o rótulo da instância sob avaliação. Normalmente, esses algoritmos utilizam a confiança como critério estatístico, pois ela mede a probabilidade da classe condicionada aos atributos (Jovanoski & Lavrac [2001]; Veloso et al. [2006]; Simon et al. [2011]). Porém, como apresentado por Tan et al. [2002], não existe um critério que seja melhor independentemente do domínio de aplicação devido às suas propriedades intrínsecas. Veloso et al. [2009] analisaram a eficácia de oito critérios estatísticos em um algoritmo de aprendizado de máquina baseado em regras de associação. Os autores concluíram que, dependendo do domínio de competência da aplicação, um critério é preferível em relação aos outros. A tabela 2.1 enumera quatro dos critérios que foram utilizados e suas fórmulas. Abaixo serão descritas as intuições por trás de cada um.

m_1 **Confiança**: mede a acurácia da regra $\mathcal{X} \rightarrow y_j$ ou a probabilidade da classe y_j condicionada aos atributos em $\mathcal{X}$;

m_2 **Suporte**: mede a frequência na qual a regra acontece na base;

m_3 **Valor Agregado**: mede o ganho na acurácia ao utilizar a regra $\mathcal{X} \rightarrow y_j$ em vez de prever y_j diretamenta atravéz de amostragem aleatória. Valores negativos indicam que é melhor prever y_j diretamente do que utilizar a regra para fazer predição;

m_4 **Yules'Q**: normalização de outro critério conhecido como razão de probabilidade (*odds ratio*); ele mede a correlação entre os atributos $\mathcal{X}$ e a classe y_j. Seus valores variam de -1 a 1, sendo que 1 implica em uma associação positiva perfeita entre $\mathcal{X}$ e y_j, 0 indica nenhuma associação, e -1, uma associação negativa perfeita.

Além dos quatro critérios estatísticos apresentados acima, existem vários outros, sendo um desafio decidir qual critério utilizar para avaliar a utilidade das regras de

Tabela 2.1: Critérios estatísticos de avaliação das regras de associação.

	Métrica	Fórmula
m_1	Confiança	$P(y_j\|\mathcal{X})$
m_2	Suporte	$P(\mathcal{X} \cup y_j)$
m_3	Valor Agregado	$P(y_j\|\mathcal{X}) - P(y_j)$
m_4	Yules'Q	$\frac{P(\mathcal{X}\cup y_j)P(\overline{\mathcal{X}\cup y_j})-P(\mathcal{X}\cup\overline{y_j})P(\overline{\mathcal{X}}\cup y_j)}{P(\mathcal{X}\cup y_j)P(\overline{\mathcal{X}\cup y_j})+P(\mathcal{X}\cup\overline{y_j})P(\overline{\mathcal{X}}\cup y_j)}$

associação. Com a finalidade de guiar essa escolha, alguns pesquisadores estudaram quais as propriedades importantes que um critério $\mathcal{CR}$ deve possuir. A seguir, serão apresentadas oito dessas propriedades, sendo que as três primeiras foram introduzidas por Piatetsky-Shapiro [1991].

P1 : $\mathcal{CR} = 0$ se $\mathcal{X}$ e y_j são estatisticamente independentes.

P2 : $\mathcal{CR}$ cresce monotonicamente com $P(\mathcal{X}, y_j)$ quando $P(\mathcal{X})$ e $P(y_j)$ permanecem os mesmos.

P3 : $\mathcal{CR}$ cresce monotonicamente com $P(\mathcal{X})$ ou $P(y_j)$ quando os parâmetros $P(\mathcal{X}, y_j)$, $P(\mathcal{X})$ ou $P(y_j)$ permanecem os mesmos.

As próximas cinco propriedades foram introduzidas por Tan et al. [2002]. Elas foram descritas utilizando-se uma matriz de contingência $\boldsymbol{M}$, mostrada na tabela 2.2, sendo cada métrica representada por uma função O sob $\boldsymbol{M}$. Por exemplo, a confiança é dada por $\frac{M_{1,1}}{M_{1,1}+M_{1,2}}$.

Tabela 2.2: Matriz de contingência $\boldsymbol{M}$.

$P(\mathcal{X}, y_j)$	$P(\mathcal{X}, \overline{y_j})$
$P(\overline{\mathcal{X}}, y_j)$	$P(\overline{\mathcal{X}}, \overline{y_j})$

P4 [Simetria sob permutação variável]: um critério $\mathcal{CR}$ é simétrico sob permutação variável se $\mathcal{X} \to y_j = y_j \to \mathcal{X}$, ou seja, $O(\boldsymbol{M}^T) = O(\boldsymbol{M})$ para todas as possíveis matrizes $\boldsymbol{M}$.

P5 [Invariância na multiplicação da fileira/coluna por escalar]: seja $\boldsymbol{R} = \boldsymbol{C} = \begin{bmatrix} k_1 & 0 \\ 0 & k_2 \end{bmatrix}$, sendo k_1 e k_2 constantes positivas. O produto $\boldsymbol{R} \times \boldsymbol{M}$ corresponde a multiplicar a primeira fileira de $\boldsymbol{M}$ por k_1 e a segunda por k_2, enquanto $\boldsymbol{M} \times \boldsymbol{C}$ corresponde a multiplicar a primeira coluna de $\boldsymbol{M}$ por k_1 e a segunda por k_2. Um critério é invariante sob multiplicação de coluna ou fileira por escalar se $O(\boldsymbol{R} \times \boldsymbol{M}) = O(\boldsymbol{M})$ e $O(\boldsymbol{M} \times \boldsymbol{C}) = O(\boldsymbol{M})$ para todas matrizes $\boldsymbol{M}$.

P6 [Assimetria entre permutação de fileira/coluna]: seja $\boldsymbol{V} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, um critério é assimétrico sob permutação de fileiras se $O(\boldsymbol{V} \times \boldsymbol{M}) = -O(\boldsymbol{M})$ e assimétrico sob permutação de colunas se $O(\boldsymbol{M} \times \boldsymbol{V}) = -O(\boldsymbol{M})$ para todas matrizes $\boldsymbol{M}$. Critérios que são simétricos sob permutação de fileira e coluna não distinguem entre correlações positivas e negativas da tabela de contingência e devem ser utilizados com cautela.

P7 [Invariância sob inversão]: a inversão consiste em trocar as fileiras e colunas de lugar. Seja $\boldsymbol{V} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, um critério é invariante sob a inversão se $O(\boldsymbol{V} \times \boldsymbol{M} \times \boldsymbol{V}) = O(\boldsymbol{M})$ para todas matrizes $\boldsymbol{M}$. Para o caso no qual os valores são binários, a inversão pode ser pensada como o operador de negação, ou seja, trocar zeros por uns e vice-versa. Logo, critérios invariantes são ruins para serem utilizados em aplicações que requerem tratamento diferenciado entre os valores binários da variável, como cestas de compras.

P8 [Invariância nula]: um critério binário possui invariância nula se $O(\boldsymbol{M} + \boldsymbol{C}) = O(\boldsymbol{M})$, para $\boldsymbol{C} = \begin{bmatrix} 0 & 0 \\ 0 & k \end{bmatrix}$, sendo k uma constante positiva. Para variáveis binárias, esta operação corresponde a adicionar mais transações na base de dados que não contenham $\mathcal{X}$ e y_j. Esta propriedade é útil em aplicações com base de dados esparsa nas quais a co-presença é mais importante do que a co-absência.

A tabela 2.3 mostra qual o intervalo dos valores possíveis para cada critério e suas propriedades. Note que nenhum critério é capaz de atender a todas as propriedades. Em nossa estratégia, vamos tirar proveito desse fato para combinar os critérios estatísticos com a finalidade de complementar suas propriedades.

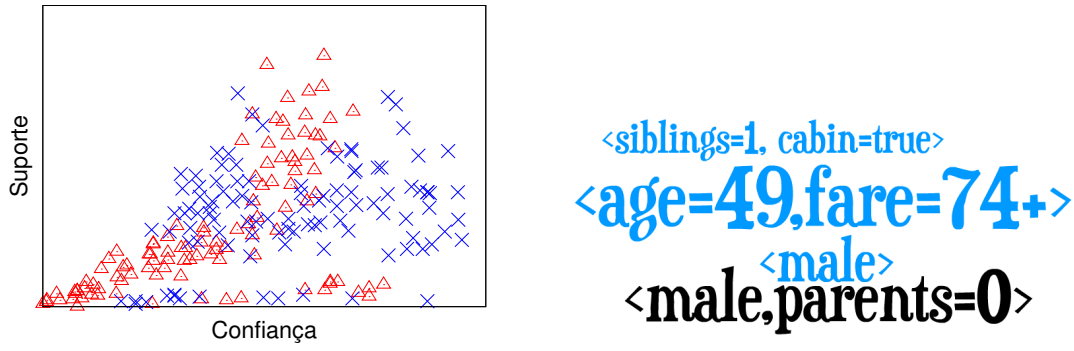
Tabela 2.3: Propriedades dos critérios estatísticos.

	Métrica	Extensão	Propriedades							
			P1	P2	P3	P4	P5	P6	P7	P8
m_1	Confiança	0...1	0	1	0	0	0	0	0	1
m_2	Suporte	0...1	0	1	0	1	0	0	0	0
m_3	Valor Agregado	$-0,5$...1	1	1	1	0	0	0	0	0
m_4	Yules'Q	-1...1	1	1	1	1	1	1	1	0

2.1.2 Visualização das Regras de Associação

Uma metáfora visual tem como objetivo representar informações complexas a fim de que elas possam ser absorvidas rapidamente pelo especialista e ele seja capaz de realizar um raciocínio dedutivo a partir do modelo gerado pelo classificador e de sua representação visual. Ou seja, a representação gráfica permite a assimiliação rápida de informações. Isso ocorre devido ao canal visual possuir uma banda grande de dados em direção ao cérebro, permitindo às pessoas processarem quantidades grandes de informação simultaneamente (Thomas & Cook [2005]).

Um dos desafios da visualização dos modelos está relacionado com a escalabilidade, pois, devido a fatores cognitivos, os seres humanos não são capazes de processar figuras com mais de duas ou três dimensões, que contenham mais de cinco ou seis atributos (Vellido et al. [2011]). Por isso, a etapa de filtragem, como a proposta neste trabalho, é fundamental para melhorar a compreensão de um modelo baseado em regras.



(a) Gráfico de dispersão. (b) Nuvem de palavras.

Figura 2.1: Visualização de regras de associação.

A figura 2.1 mostra dois exemplos de modelos de visualização de regras de associação. A figura 2.1a é um gráfico de dispersão composto pelos critérios estatísticos

confiança e suporte, sendo os pontos representados por regras geradas a partir dos atributos de uma instância de teste. Os pontos de cor azul pertencem a uma classe e os de cor vermelha, a outra. Essa estratégia de visualização foi proposta inicialmente no trabalho de Bayardo & Agrawal [1999]. Ela funciona bem quando o número de regras e atributos é grande, porém, o gráfico de dispersão apenas permite compreender qual é a classe dominante. Já na figura 2.1b, além de ser possível compreender qual é a classe dominante, pode-se analisar os fatores que influenciaram na decisão tomada pelo classificador. No entanto, seu uso não é indicado quando o modelo gera muitas regras. Esse gráfico é conhecido como nuvem de palavras, no qual cada regra é representada por uma "palavra", o tamanho da regra é proporcional à sua confiança e a cor da regra representa a sua classe. Outras estratégias de visualização de regras de associação são apresentadas por Hahsler & Chelluboina [2013].

2.1.3 Mineração de Conjunto Frequente

A mineração de conjunto frequente tem como objetivo descobrir quais regras de associação, que descrevem uma base de dados $\mathcal{D}$, possuem suporte acima de um determinado limite. Sua aplicação inicial mais popular foi na análise de cestas de compras (Agrawal et al. [1993]; Meira & Zaki [2011]). Por exemplo, em um supermercado, poderia ser detectada a regra "clientes que compram leite e cereal tendem a comprar banana".

Dado um conjunto de itens $\mathcal{I}$, o total de conjuntos que podem ser formados a partir dele é igual a $2^{|\mathcal{I}|}$, incluindo o conjunto vazio. A tabela 2.4 mostra o tempo de execução teórico para gerar todos os subconjuntos, variando a quantidade de itens, ajustado para um algoritmo linear que executa um milhão de operações por segundo, (Ziviani [1993]). Logo, para conjuntos com mais de 30 itens, o problema fica inviável. Na prática, é definido um limite de suporte e apenas são gerados os conjuntos frequentes com base na propriedade de que um conjunto somente é frequente se seus subconjuntos forem frequentes. Essa estratégia é um **filtro a priori** das regras de associação, pois ela é capaz de filtrar as regras antes de gerá-las. Um **filtro a posteriori** depende que a regra seja gerada para só depois filtrá-la.

Tabela 2.4: Tempo de execução teórico.

Função de custo	Tamanho de $\mathcal{I}$					
	10	20	30	40	50	60
$2^{\lvert\mathcal{I}\rvert}$	0,001 seg.	1 seg.	18 min.	13 dias	35 anos	366 sec.

Dada essa propriedade a priori do suporte, foi proposto por Agrawal & Srikant [1994] o algoritmo conhecido como Apriori para geração de conjuntos frequentes, que reduz o tempo de execução consideravelmente. A figura 2.2 apresenta uma treliça contendo todos os conjuntos de itens gerados a partir do conjunto $\{A, B, C, D, E\}$. Em cada nível k da treliça, os conjuntos possuem tamanho k, sendo que o nível 1 possui cinco conjuntos de tamanho 1 e o nível k possui $\binom{|\mathcal{I}|}{k}$ conjuntos. Dado que os itens estão ordenados lexicograficamente, o prefixo de um conjunto é dado pelos $k-1$ itens à esquerda, mostrados de azul na figura. Os conjuntos de itens que possuem o mesmo prefixo estão colados. Para gerar os conjuntos do nível $k+1$, basta fazer a união dos conjuntos de k-itens que possuem o mesmo prefixo e o k-ésimo item do conjunto à esquerda deve ser lexicograficamente maior que o k-ésimo item do conjunto à direita. Na treliça, as arestas ligam os dois subconjuntos do nível k que foram unidos para formar o conjunto do nível $k+1$. No exemplo, o conjunto $\{E\}$, colorido de roxo, não é frequente. Portanto, todos os conjuntos coloridos de roxo, que possuem $\{E\}$ como subconjunto, não serão gerados.

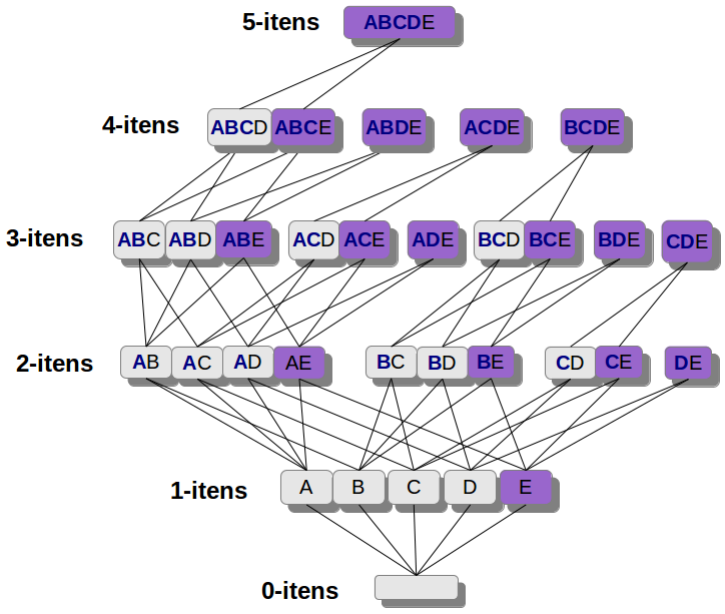


Figura 2.2: Treliça.

O Apriori realiza uma busca em largura na treliça, sendo que, a cada conjunto de itens gerado, ele precisa consultar a base de dados para calcular seu suporte. O formato da base de dados do Apriori ficou conhecido como horizontal e, nele, cada transação t_i possui um conjunto de atributos $\mathbf{i}(t_i)$. Logo, são realizadas várias leituras

da base de dados. Para melhorar esse aspecto, foi proposto por Zaki et al. [1997] o algoritmo Eclat. O Eclat mantém a base de dados no formato vertical, ou seja, cada atributo possui uma lista de transações nas quais ele ocorre. Nesse formato, para gerar um conjunto de tamanho $k+1$, basta fazer a interseção das listas de identificadores dos dois conjuntos de tamanho k. Além disso, foi demonstrado que os conjuntos de itens de tamanho $k+1$ só dependem dos conjuntos de itens de tamanho k que possuem $k-1$ atributos iguais. Desse modo, é possível caminhar em profundidade na treliça e liberar memória (as listas de transações de cada conjunto de itens) durante o caminhamento. Com essas melhorias, o Eclat conseguiu uma melhora de desempenho de mais de uma ordem de magnitude em relação ao Apriori.

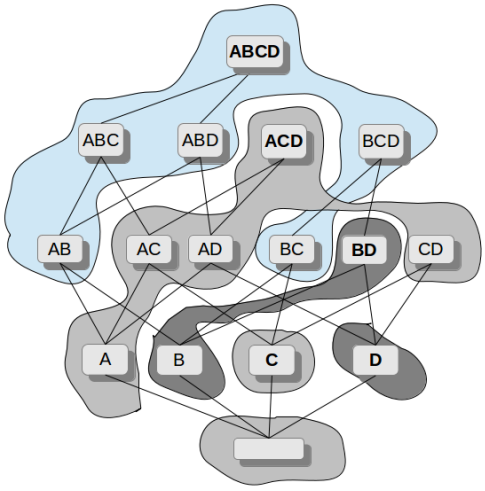
O problema do Eclat é que as listas de transações dos conjuntos de itens ficam muito grandes para caberem em memória. Para resolver isso, foi proposto o Diff Eclat por Zaki & Gouda [2003]. O Diff Eclat é uma extensão do Eclat, que, em vez de manter uma lista de transações em que cada conjunto de itens ocorre, armazena uma lista que representa a diferença entre as transações dos subconjuntos que geraram o conjunto. Por exemplo, sendo $\boldsymbol{t}(A)$ o conjunto de transações que o item A ocorre e $\boldsymbol{d}(AB)$ as transações que ocorrem em B e não ocorrem em A, temos $\boldsymbol{d}(AB) = \boldsymbol{t}(B) - \boldsymbol{t}(A)$. O cálculo do suporte de AB é dado por $\sigma(AB) = \sigma(A) - |\boldsymbol{d}(AB)|$. Portanto, além da lista de diferenças deve-se armazenar o suporte do conjunto. O cálculo de $\boldsymbol{d}(ABC)$ é dado por $\boldsymbol{d}(AC) - \boldsymbol{d}(AB)$ e assim por diante. O Diff Eclat reduz drasticamente o consumo de memória em relação ao Eclat e, no artigo do Diff Eclat, foi mostrado que seu desempenho é ordens de magnitude melhor do que as outras estratégias do estado da arte de mineração de itens frequentes. Isso faz com que o Diff Eclat seja mais escalável, permitindo que ele enumere padrões frequentes mesmo em bases de dados densas com um suporte mínimo pequeno.

2.1.4 Representações Condensadas

Em algumas situações, mesmo com uma restrição justa de suporte mínimo, é gerado um número grande de regras de associação. Isto implica em problemas de desempenho e legibilidade dos dados. Para solucionar esses problemas, foram feitas várias propostas de representações condensadas das regras de associação. Uma representação condensada somente armazena a cobertura composta pelas regras não redundantes. Com essa cobertura, é possível inferir as outras regras e, em muitas situações práticas, elas são ordens de grandeza menores do que a coleção completa de regras de associação (Calders & Goethals [2007]). Dentre as representações condensadas existentes, as mais populares são as dos conjuntos geradores (ou conjuntos livres) e as dos conjuntos

TID	Itens			
1	B	D		
2	A	B	C	D
3	A	C	D	
4	C			

(a) Base de dados.



(b) Treliça.

Figura 2.3: Representações condensadas das regras de associação.

fechados.[1] Suas definições são dadas a seguir. Seja $\mathcal{R}^x$ um conjunto de regras:

Geradores: uma regra $\mathcal{X} \rightarrow y_j \in \mathcal{R}^x$ pertence ao conjunto dos geradores se não existe outra regra $\mathcal{C} \rightarrow y_j \in \mathcal{R}_x$ tal que $\mathcal{X} \supseteq \mathcal{C}$ e ambas as regras $\mathcal{X} \rightarrow y_j$ e $\mathcal{C} \rightarrow y_j$ possuam o mesmo suporte.

Fechados: uma regra $\mathcal{X} \rightarrow y_j \in \mathcal{R}^x$ pertence ao conjunto fechado se não existe outra regra $\mathcal{C} \rightarrow y_j \in \mathcal{R}_x$ tal que $\mathcal{X} \subseteq \mathcal{C}$ e ambas as regras $\mathcal{X} \rightarrow y_j$ e $\mathcal{C} \rightarrow y_j$ possuam o mesmo suporte.

A figura 2.3 mostra um exemplo de mineração de conjuntos geradores e fechados, retirado do trabalho de Yahia et al. [2006]. A tabela 2.3a representa a base de dados que foi utilizada para gerar a treliça da figura 2.3b. Os conjuntos agrupados sob o mesmo contorno formam uma classe de equivalência, ou seja, possuem o mesmo conjunto de transações. Os menores conjuntos de uma classe de equivalência são os geradores e o maior conjunto de uma classe de equivalência, o fechado. No exemplo, o conjunto de geradores é dado por $\{\{\emptyset\}, \{A\}, \{B\}, \{C\}, \{D\}, \{AB\}, \{BC\}\}$ e o conjunto dos fechados é dado por $\{\{\emptyset\}, \{C\}, \{D\}, \{BD\}, \{ACD\}, \{ABCD\}\}$. Neste trabalho, utilizaremos o conjunto fechado como base de comparação com nossa estratégia de filtro de regras. Ao utilizar o conjunto fechado neste exemplo, o número de conjuntos passou de 16 para 6, uma redução de 62%.

[1] Uma revisão detalhada sobre representações condensadas pode ser encontrada no trabalho de Calders & Goethals [2007].

2.2 Aprendizado de Máquina

Aprendizado de máquina consiste em utilizar o computador para encontrar padrões em dados passados para realização de inferência nos dados futuros, mesmo sem saber a priori qual processo gerou os dados (Alpaydin [2004]). Pode ser que não seja possível identificar completamente o processo que gerou os dados, mas é possível construir uma aproximação boa e útil. Suas áreas de aplicação são abundantes: classificação de clientes para aplicações de crédito utilizada pelos bancos, análise de sentimento, desambiguação de entidades, recomendação de rótulos (tags), ordenação de resultados de máquina de busca, diagnóstico de câncer de pulmão, entre outras.

Neste trabalho, lidaremos com aprendizado de máquina supervisionado aplicado à classificação de dados. De acordo com Grünwald & Langford [2007], um problema de aprendizado de máquina é definido em um domínio de entrada (ou característica) X, um domínio de saída (ou rótulo da classe) $\mathcal{Y}$ e uma distribuição de probabilidade $\mathcal{P}$ sob $X \times \mathcal{Y}$, tal que um classificador é uma função $\mathcal{F}_S : X \to \mathcal{Y}$.

No caso supervisionado, o problema de classificação consiste em encontrar um classificador que gere uma distribuição de probabilidade o mais próxima possível de $\mathcal{P}$, procurando em um espaço de hipóteses $\mathcal{H}$, utilizando uma amostra $\mathcal{S} = \{s_1 = (x_1, y_1), s_2 = (x_2, y_2), ..., s_n = (x_n, y_n)\} \sim P^n$, gerada por n amostras independentes da distribuição $\mathcal{P}$. O conjunto $\mathcal{S}$ é conhecido como instâncias de treinamento e as instâncias que serão classificadas utilizando o classificador $\mathcal{F}_S$ são conhecidas como instâncias de teste, dadas pelo conjunto $\mathcal{T} = \{t_1 = (x_1, ?), t_2 = (x_2, ?), ..., t_m = (x_m, ?)\} \sim \mathcal{P}^m$.

2.2.1 Classificação Associativa

Na classificação associativa, o modelo gerado $\mathcal{F}_S$ consiste em um conjunto de regras de associação nas quais cada atributo mede a importância de uma determinada característica na instância de teste. Em nosso caso, vamos utilizar apenas atributos discretos mutuamente exclusivos.[2] Os métodos de aprendizado de classificadores associativos podem ser analisados considerando-se duas características, sendo elas o momento da geração das regras e a forma de calcular o rótulo da instância. Dependendo do momento em que o classificador gera seu conjunto de regras, ele pode ser

[2]Caso o atributo possua valores reais, estes valores podem ser discretizados utilizando um método de discretização, como apresentado por Fayyad & Irani [1993]. Esse método utiliza ganho de informação (minimização da entropia), para decidir como discretizar os dados, e Descrição de Tamanho Mínimo (*Minimum Description Length* - MDL), para decidir a quantidade ideal de intervalos a serem utilizados na discretização. Uma das contribuições deste trabalho é a implementação eficiente do método de discretização de dados, presente no Repositório de Aprendizado de Máquina da UFMG (Veloso & Dafe [2012]).

categorizado como global ou local. As explicações de ambos os momentos são dadas a seguir:

Geração global (*eager*): o conjunto de regras é gerado uma única vez a partir das instâncias de treino, porém o espaço de regras possíveis é exponencial, sendo inviável gerar todas as regras. Portanto, deve-se escolher prematuramente quais são as regras que melhor modelam o treino. Além disso, podem vir a ser geradas regras que não serão utilizadas para classificar novas instâncias. A vantagem desse método é que o tempo de classificação durante o teste torna-se desprezível.

Geração local (*lazy*): os modelos são gerados durante o teste para cada instância e, na etapa de treino, ocorre apenas a armazenagem dos dados. Ou seja, não são geradas regras desnecessárias.

Além dos momentos de geração das regras, os classificadores são categorizados dependendo de como eles definem o rótulo de instância de teste. No caso, pode ser utilizada uma única regra de uma lista ordenada ou uma mistura de regras, como explicado a seguir:

Lista ordenada: o classificador ordena as regras de acordo com algum critério. Na etapa de classificação, será utilizada para rotular a instância a primeira regra cujos atributos sejam um subconjunto dos atributos da instância de teste. A vantagem desse método é a utilização de apenas uma regra de associção para classificar uma instância do teste. Portanto, é fácil interpretar o que levou o modelo a tomar deteminada decisão. No entanto, se a lista não estiver ordenada corretamente, o modelo gerado não será preciso e existe um número fatorial de listas possíveis (Letham et al. [2012]).

Mistura de regras (*ensemble*): na etapa de classificação, são utilizadas todas as regras cujos atributos sejam um subconjunto dos atributos da instância de teste. No caso, cada regra representa um voto ponderado para uma determinada classe. Na prática, modelos de mistura de regras têm apresentado resultados superiores aos de listas ordenadas. Isso ocorre porque é removida a restrição da ordem entre as regras. Outro fator importante é que, na mistura de regras, cada regra pode ser vista como um modelo fraco que é agrupado aos demais com o objetivo de formar um modelo forte, sendo que este modelo forte tende a ser mais preciso.

Dentre as características apresentadas para classificadores associativos, os melhores resultados vêm sendo obtidos a partir da geração local (sob demanda) do conjunto

de regras juntamente com a utilização de um modelo de mistura para determinar o rótulo de classe (Jerome & Bogdan [2008]; Veloso et al. [2006]). A geração global é recomendada para a tarefa de mineração de dados quando se procura encontrar padrões que expliquem a base. Porém, para a tarefa de classificação, a geração global cria várias regras desnecessárias para uma determinada instância e, além disso, pode deixar de gerar algumas regras importantes. Logo, neste trabalho, utilizaremos como base um classificador associativo sob demanda. Serão dados mais detalhes sobre ele na seção a seguir.

2.2.2 Classificação Associativa Sob Demanda (LAC)

A geração de regras sob demanda é uma estratégia recente utilizada para evitar o imenso espaço de busca formado por todos os possíveis subconjuntos dos atributos presentes na base de dados (Veloso et al. [2006]; Veloso & Meira [2011]). Em vez de gerar um conjunto global de regras durante o treino, o algoritmo gera um conjunto local específico para cada instância de teste. Isto é feito realizando uma projeção dos dados de treino $\mathcal{T}$, com base nos atributos da instância de teste x_i. Então, é gerado um conjunto de regras $\mathcal{R}_i = \{\mathcal{X} \rightarrow y_j \mid \mathcal{X} \subseteq x_i\} \; \forall \; y_j \in \mathcal{Y}$, no qual os atributos das regras geradas são subconjuntos dos atributos da instância de teste. Cada regra do tipo $\mathcal{X} \rightarrow y_j$ representa um voto para uma classe y_j, sendo os votos ponderados por um critério estatístico $\mathcal{CR}$ (normalmente é utilizada a confiança como critério estatístico porque ela representa a probabilidade condicional da classe, dados os atributos da regra). No final, a classe que recebeu a maior votação é escolhida como rótulo da instância x_i.

Algoritmo 1 LAC Classificação Associativa Sob Demanda

Entrada: Conjunto de treino $\mathcal{S}$, conjunto de teste $\mathcal{T}$, suporte mínimo σ_{min}, confiança mínima θ_{min}, tamanho máximo da regra l

1: **for all** instância $x_i \in \mathcal{T}$ **do**
2: $\quad R_i \leftarrow \boldsymbol{induceRules}(x_i, \mathcal{S}, \sigma_{min}, \theta_{min}, l)$ ▷ Apriori, Eclat ou DEclat
3: $\quad$ **for all** classe $y_j \in \mathcal{Y}$ **do**
4: $\quad\quad s(x_i, y_j) \leftarrow \sum_{r \in R_i^{y_j}} \mathcal{CR}(r)$
5: $\quad \mathcal{F}_s(x_i) = y_j \leftarrow \{y_j \in \mathcal{Y} : argmax(\; s(x_i, y_j)\;)\}$

O funcionamento do classificador associativo sob demanda descrito acima é mostrado no algoritmo 1 e ficou conhecido na literatura como LAC (*Lazy Associative Classification*) (Veloso et al. [2006]). A função ***induceRules***, encarregada de gerar as regras de associação relacionadas à instância x_i, pode utilizar qualquer algoritmo de

mineração de regras, como os apresentados na seção 2.1.3. Como requisito do LAC, o algoritmo que irá gerar as regras de associação deve ser capaz de aplicar três tipos de filtros dois a priori, que são as restrições do suporte (σ_{min}) e do tamanho da regra (l), e um a posteriori, que é a restrição da confiança (θ_{min}). Portanto, apenas serão utilizadas pelo LAC as regras que satisfizerem às condições $|\mathcal{X}| < l$, $\sigma(\mathcal{X} \rightarrow y_j) \geq \sigma_{min}$ e $\theta(\mathcal{X} \rightarrow y_j) \geq \theta_{min}$. Apesar desses filtros, a quantidade de regras utilizadas pelo LAC ainda pode ser grande, sendo os modelos aprendidos de difícil legibilidade. Logo, o problema que vamos tratar neste trabalho é como melhorar a legibilidade de um classificador associativo sem prejudicar sua acurácia.[3] No pior caso, a complexidade de tempo do LAC é $\mathcal{O}(|\mathcal{T}| \times |\mathcal{S}| \times |\mathcal{Y}| \times \sum_{m=1}^{l} \binom{|x_i|}{m})$, sendo $\mathcal{T}$ a base de treino, $\mathcal{S}$ a base de teste, $\mathcal{Y}$ o conjunto de classes, l a restrição do maior tamanho de regra e x_i uma instância de teste. Os motivos de termos escolhido o algoritmo LAC como base neste trabalho foram:

1. O LAC é um algoritmo de classificação baseado em regras de associação que gera um modelo específico para cada instância de teste.
2. O LAC está entre os algoritmos que representam o estado da arte em aprendizado de máquina.
3. O LAC já foi utilizado com sucesso em diversas aplicações, tais como análise de sentimento, desambiguação de entidades, recomendação de rótulos (tags), ordenação de resultados de máquina de busca, diagnóstico de câncer de pulmão (Veloso & Meira [2011]).
4. O LAC possui código de referência que pode ser baixado na página do autor.[4]

2.3 Interpretabilidade

Para que um modelo seja interpretável, é desejável que ele seja uma representação fiel do problema, de modo que suas decisões possam ser explicadas pelas hipóteses e dados. Além disso, devido a fatores cognitivos dos humanos, o modelo deve conter poucas variáveis e possuir alguma representação visual (Vellido et al. [2012]). Devem

[3]Uma das contribuições deste trabalho é a implementação eficiente do algoritmo LAC. Ela permite que o usuário escolha entre utilizar o algoritmo Eclat ou o algoritmo DEclat para gerar as regras de associação. O algoritmo Eclat é mais rápido que o algoritmo DEclat para regras de tamanho até 3, devido à sua estratégia de cache, mas, a partir deste ponto, o desempenho do algoritmo DEclat é melhor. A implementação estará presente no Repositório de Aprendizado de Máquina da UFMG (Veloso & Dafe [2012]).

[4]http://homepages.dcc.ufmg.br/~adrianov/software/ddac.htm

ser selecionadas as variáveis que melhor provêem um entendimento do processo que gerou os dados e que possuem relação causal com o objetivo da classificação. Além da escolha cuidadosa dos atributos de entrada, também é importante que o modelo gerado seja de fácil compreensão. As vantagens de uma aplicação de aprendizado de máquina na qual o modelo é fácil de compreender são:

1. Facilidade em analisar o impacto dos atributos utilizados pela aplicação.
2. Facilidade em entender o comportamento do modelo diante das hipóteses assumidas, dos dados passados e das previsões futuras.
3. Facilidade em explicar as decisões tomadas pelo modelo para pessoas que não são da área de aprendizado de máquina.
4. Segurança e conforto em utilizar uma aplicação na qual o especialista compreende seu comportamento.

Além dessas vantagens, em alguns domínios de aplicação, como na área médica, na detecção de fraudes, na análise de crédito ou na análise de impacto de uma campanha de *marketing*, a interpretabilidade é um requisito fundamental. Nessas aplicações, especialistas tendem a preferir modelos mais transparentes (Letham et al. [2012, 2013]).

Neste trabalho, buscamos melhorar a interpretabilidade de um classificador baseado em regras de associação, reduzindo a quantidade de regras utilizadas por ele. Na seção seguinte, serão discutidos os compromissos entre a interpretabilidade de um modelo e sua acurácia.

2.3.1 Interpretabilidade Versus Acurácia

O conceito de eficiência de Pareto também pode ser utilizado para entender a dicotomia entre interpretabilidade e acurácia. Existe um limiar no qual não é possível aumentar a interpretabilidade do modelo sem perder em acurácia. Em aprendizado de máquina, é comum a busca por modelos preditivos de alta acurácia, mesmo que isso implique na perda de seu poder de explanação. Um modelo explanatório é aquele concebido com o objetivo de testar hipóteses causais que podem ser explicadas por uma construção teórica (assumem que os dados foram gerados por um processo estocástico). Já um modelo preditivo tem como objetivo classificar dados futuros (Shmueli [2010]). Normalmente, estatísticos preocupam-se mais com a criação de modelos explanatórios, que são construídos modelando o problema por meio de alguma distribuição estatística, enquanto que cientistas da computação preocupam-se em desenvolver o modelo o mais preciso possível. Por exemplo, um cientista da computação pode selecionar alguns

atributos que não possuam significado semântico com o problema, mas que aumentem a acurácia do modelo. O aumento da acurácia ocorre porque a amostra de dados utilizada pelo modelo não apresenta informação suficiente para determinar o processo real que gerou os dados. O erro de um algoritmo de classificação pode ser decomposto pela fórmula a seguir:

$$ERRO = VAR(y_j) + vies^2 + VAR(\mathcal{F}(\mathcal{X}))$$

sendo y_j a variável a ser prevista, $\mathcal{X}$ os atributos de entrada, $\mathcal{F}$ o modelo de classificação e $vies$ o conjunto de premissas utilizadas na especificação do modelo $\mathcal{F}$ com relação ao modelo real (Shmueli [2010]). A primeira variância representa o erro que acontece mesmo se o modelo for corretamente especificado, e a segunda variância (variância de estimação) é o resultado da utilização de uma amostra para estimar $\mathcal{F}$. A partir da decomposição do erro, pode ser observada uma fonte de diferença entre modelos explanatórios e preditivos. Na modelagem explanatória, o foco é minimizar o viés para obter a representação mais próxima possível do problema real. Em contraste, a modelagem preditiva procura minimizar a combinação do viés com a variância de estimação, ocasionalmente sacrificando a proximidade do modelo com o problema real, para maximizar a acurácia.

Na prática, é desejável que o modelo possua ambas as características, ou seja, tenha poder preditivo e explanatório. Um exemplo disso aconteceu no Prêmio Netflix[5], cujo objetivo da competição era prever a avaliação dos usuários para os filmes, sendo o prêmio para a melhor solução de 1 milhão de dólares. A maioria dos competidores possuía expertise em ciência da computação e alguns, em estatística. A competição começou em 2006 com mais de 20 mil participantes e terminou em 2009. O vencedor foi uma composição de três times de pesquisadores dos laboratórios AT&T, da Yahoo! e da Pragmatic Theory. O interessante desse time é que ele era um misto de estatísticos com cientistas da computação. Ou seja, foram utilizadas misturas de estratégias preditivas e explanatórias. Uma explicação mais detalhada da estratégia utilizada pelo time vencedor pode ser encontrada no trabalho de Bell et al. [2010].

2.4 Parcimônia (Navalha de Occam)

O princípio da parcimônia, também conhecido como "navalha de Occam", informalmente declara que "tudo mais sendo igual, é preferível a hipótese mais simples". No contexto de aprendizado de máquina e teoria da predição, esta ideia pode ser refinada

[5] http://netflixprize.com

escolhendo-se definições precisas para as palavras "igual" e "simples". Porém, são possíveis várias definições para estas palavras. Por exemplo, simplicidade é tipicamente definida como uma função do tamanho do classificador. Para essa tarefa, poderia ser utilizado o tamanho da descrição de um programa dada pela complexidade Kolmogorov (Nannen [2003]). A complexidade Kolmogorov de um programa representa o tamanho da menor máquina de Turing capaz de codificá-lo. Porém, não é possível calcular essa complexidade. Então, na prática, são utilizadas heurísticas para modelar a complexidade do modelo, como o número de nodos em uma árvore de decisão, o número de condições em um conjunto de regras de associação ou o número de parâmetros do classificador (Domingos [1999]; Li et al. [2006]). Neste trabalho, vamos estimar a simplicidade de um classificador por meio da quantidade de regras de associação que ele possui. Definições de "igual" também são problemáticas (Lattimore & Hutter [2011]). Poderia ser considerado que um classificador é melhor do que o outro se ele provesse um erro menor no treino. Porém, essa definição pode levar a problemas como sobreajustamento (*overfitting*). Então, neste trabalho, nós vamos comparar os classificadores por meio do erro de generalização (i.e., por meio de validação cruzada). Por fim, apenas será considerado que um classificador é melhor do que o outro se seu erro de generalização for significativamente menor. Portanto, serão utilizados testes de significância para comparar os classificadores, a fim de assegurar que eles são estatisticamente diferentes.

Além da legibilidade do modelo, outro ponto importante que justifica a preferência por modelos simples é o fato de que quanto mais complexo ele for, maior é a chance de ele ser pior para prever dados futuros, pois modelos complexos tendem a sobreajustar os dados de treino (Sober et al. [2002]). Algumas teorias de seleção de modelos que se baseiam nesse pensamento e têm sido utilizadas com sucesso são Critério de Informação de Akaike (AIC) e Descrição de Tamanho Mínimo (MDL) (Nannen [2003]). Em ambas, o objetivo é minimizar a soma entre o tamanho da descrição do modelo e o erro do modelo no treino, a fim de minimizar o erro de generalização.

2.5 Eficiência de Pareto

O conceito de eficiência de Pareto teve início na economia e foi desenvolvido pelo italiano Vilfredo Pareto no meio do século XIX. Dado um grupo de pessoas, uma transação realizada entre elas é considerada uma melhoria de Pareto se pelo menos um indivíduo aumentou sua utilidade total e a utilidade dos outros envolvidos não sofreu redução. O grupo de pessoas irá alcançar um estado Pareto-Ótimo, ou Pareto-Eficiente, quando não for possível realizar mais nenhuma melhoria de Pareto, ou seja,

não for possível realizar outra transação sem degradar a utilidade de algum envolvido. O subconjunto de soluções que satisfazem à condição de eficiência de Pareto formam a fronteira de Pareto (Börzsönyi et al. [2001]). A fronteira de Pareto também é conhecida por *skyline* ou *maximal vector* (Godfrey et al. [2007]). A fronteira de Pareto consiste num subconjunto de pontos tal que nenhum desses pontos seja dominado por qualquer outro. Um ponto em $\mathcal{R}^d$ consiste em um vetor de números reais de tamanho d. Um ponto domina o outro se ele é melhor ou igual em todas as dimensões e é melhor em pelo menos uma dimensão. A eficiência de Pareto tem sido utilizada em outras áreas além da economia, como, por exemplo, otimização (Ribeiro et al. [2014]). Na área de otimização, a eficiência de Pareto é utilizada quando se deseja maximizar uma função multiobjetivo, sendo a fronteira de Pareto o conjunto de soluções ótimas.

A seguir, será apresentado um exemplo baseado em hotéis extraído do trabalho de Godfrey et al. [2007]. Nesse exemplo, um cliente pede a uma agência de viagens opções de hotéis que levem em consideração a distância da praia, o número de estrelas e o preço. A agência possui uma lista de hotéis, apresentados na tabela 2.5. Portanto, cada hotel pode ser visualizado como um ponto no espaço tridimensional. Aplicando a fronteira de Pareto nessa tabela, ela irá retornar os hotéis que não são dominados por nenhum outro, neste caso, os hotéis em negrito. O hotel Aga foi eliminado por comparação com o hotel Uma; Fol foi eliminado por comparação com o hotel Aga, Kaz ou Uma; Tor foi eliminado por comparação com Neo. Repare que um ponto na fronteira de Pareto não precisa ser o melhor em nenhum critério. Por exemplo, Uma não é o hotel com mais estrelas, não é o mais perto da praia e não é o mais barato. Entretanto, ele representa um bom balanceamento entre todos os critérios.

Tabela 2.5: Dados de hotéis, exemplo de eficiência de Pareto

Nome	Estrelas	Distância (m)	Preço (R$)
Aga	⋆⋆	700	1175
Fol	⋆	1200	1237
Kaz	⋆	200	750
Neo	⋆⋆⋆	200	2250
Tor	⋆⋆⋆	500	2250
Uma	⋆⋆	500	980

Em nosso trabalho, utilizaremos a fronteira de Pareto como um filtro de regras de associação com o intuito de encontrar as melhores regras em um espaço de critérios estatísticos. Em particular, utilizaremos a estratégia conhecida como *Block-nested-loops* (BNL) desenvolvida por Börzsönyi et al. [2001] para encontrar a fronteira de

Pareto.

No algoritmo BNL, é mantida uma janela com os pontos que não foram dominados por nenhum ponto. Então, para cada ponto de um conjunto de pontos, o ponto atual é comparado com os pontos presentes na janela. Caso o ponto atual domine algum ponto na janela, o ponto dominado será removido. Caso o ponto atual não seja dominado por nenhum dos pontos, ele entra na janela. Para melhorar o desempenho do algoritmo, os pontos da janela que dominam algum dos pontos em comparação são passados para a frente da lista, pois estes pontos têm maior chance de dominar os próximos pontos. Com isso, no caso em que o ponto atual for dominado por algum ponto da janela, ele tem maior chance de ser dominado no início. Dessa forma, não é necessário percorrer toda a janela. O algoritmo 2 apresenta os detalhes principais do algoritmo BNL. Sendo n o número de pontos a serem avaliados e d o número de dimensões desses pontos, no melhor caso e no caso médio, a complexidade de tempo do BNL é $\mathcal{O}(d \times n)$ e, no pior caso, $\mathcal{O}(d \times n^2)$. Esse é o melhor algoritmo para calcular a fronteira de Pareto quando os dados podem ser armazenados em memória principal [6].

Algoritmo 2 BNL Block-nested-loops

Entrada: Conjunto de pontos R

```
1:  window ← ∅
2:  for all ponto r ∈ R do
3:      notDominated ← true
4:      for all ponto w ∈ window do
5:          if r dominantes w then
6:              window.remove(w)
7:          else if r is dominated by w then
8:              window.remove(w)
9:              window.push_front(w)
10:             notDominated ← false
11:             break
12:     if notDominated then
13:         window.push_back(r)
        return window                          ▷ fronteira de Pareto
```

[6] Uma das contribuições deste trabalho é uma implementação de referência do algoritmo BNL que estará presente no Repositório de Aprendizado de Máquina da UFMG (Veloso & Dafe [2012]).

2.6 Trabalhos Relacionados

Os esforços de pesquisa em busca de classificadores interpretáveis têm sido direcionados a algoritmos baseados em árvores de decisão, sistemas especialistas baseados em listas de decisão, regras de associação e redes Bayesianas (Madigan et al. [1997]). Árvores de decisão são os algoritmos de aprendizado de máquina mais utilizados e práticos (Mitchell [1997]), sendo uma de suas vantagens a interpretabilidade do modelo gerado. O algoritmo ID3, publicado por Quinlan [1986], é um dos algoritmos fundamentais nesse tema e serviu de base para muitos outros. Cada nó da árvore gerada pelo algoritmo ID3 especifica um teste em algum atributo da instância de teste e cada aresta descendente daquele nó corresponde a um valor possível deste atributo. Uma instância é classificada começando pelo nó raiz, testando o atributo especificado por aquele nó e, então, seguindo a aresta que corresponde ao valor deste atributo. Esse processo se repete até que seja alcançado um nó folha. O nó folha irá determinar a classe da instância. No algoritmo ID3, é utilizado o conceito de ganho de informação para determinar qual atributo será escolhido para representar o nó. A fim de decidir o critério de parada para a geração de novos nós, é utilizada a estratégia MDL. O problema ao se utilizar árvores de decisão é que o espaço de busca de possíveis árvores é muito grande; por isso, normalmente, as árvores são construídas de forma gulosa e podadas heuristicamente. Exemplos de algoritmos construídos dessa maneira são o CART (Breiman et al. [1984]) e o C4.5 (Quinlan [1993]). Devido às árvores não serem aprendidas por um processo de otimização global, as escolhas locais podem levar a árvores de decisões grandes e pouco precisas (Letham et al. [2012]). Além disso, foi mostrado por Veloso et al. [2006] que, se forem utilizados os mesmos critérios de ganho de informação para avaliar as regras de associação, um algoritmo baseado em regras sempre será melhor que um de árvore de decisão. Uma alternativa para o problema de máximo local tem sido utilizar algoritmos baseados em misturas de árvores (*Random Florests*), porém, a interpretabilidade destes algoritmos é bastante limitada (Breiman [2001]).

Modelos mais simples do que árvore de decisão e que foram utilizados com sucesso por sistemas especialistas são as listas de decisão (Leondes [2002]). A base de conhecimento de um sistema especialista é composta de simples condições do tipo "se−então". Listas de decisão são um caso particular de classificação associativa, dado que as listas são compostas de regras. No passado, classificadores associativos eram construídos a partir de heurísticas para gerar as regras (Rivest [1987]; Liu et al. [1998]; Li et al. [2001]; Yin & Han [2003]; Marchand & Sokolova [2005]). Provavelmente, algumas dessas heurísticas funcionaram bem em alguns casos particulares, como quando o problema de

decisão é linearmente separável (Veloso & Meira [2011]).

Os classificadores associativos que utilizam uma lista ordenada de regras de associação podem ser vistos como um tipo especial de lista de decisão. O mais famoso deles é o ApriorіC, publicado por Jovanoski & Lavrac [2001]. Esse algoritmo utiliza o algoritmo Apriori para gerar as regras de associação com base nos dados de treino. Então, ele as ordena por meio do suporte. O ApriorіC escolhe a regra com o maior suporte e retira do conjunto todas as regras cobertas pela que foi escolhida. O processo continua até que cada classe possua n regras, sendo n um parâmetro do algoritmo. Esta técnica ficou conhecida como cobertura sequencial. Então, o algoritmo ordena as regras escolhidas de acordo com algum critério, por exemplo, a confiança, e, para cada instância de teste, será utilizada a primeira regra que for um subconjunto dos atributos da instância. No entanto, o ApriorіC não escala para bases de dados grandes. Além disso, ao utilizar apenas uma regra de associação para prever a classe da instância, o algoritmo pode sofrer problemas de sobreajustamento, ou seja, o classificador não será bom para prever as instâncias do teste (Li et al. [2001]).

A cobertura sequencial é uma heurística gulosa para tentar remover as regras que são redundantes, cobrindo as mesmas instâncias de treino. Porém, existem algoritmos exatos para essa tarefa, como, por exemplo, os algoritmos que geram o conjunto fechado das regras (Yahia et al. [2006]). Na prática, esses algoritmos são mais eficientes do que a estratégia de cobertura do ApriorіC e apresentaram reduções na quantidade de regras geradas de até duas ordens de grandeza, quando comparadas com o conjunto de regras geradas pelo algoritmo Apriori (Zaki & Hsiao [2005]).

Apesar de o conjunto fechado retirar as regras redundantes, nem todas as regras geradas por ele são boas para classificação e, como mostrado por Wilkins & Ma [1994], encontrar o subconjunto ótimo de regras de associação é NP-Difícil. Devido a isso, são utilizadas heurísticas para aproximar o conjunto ótimo. As heurísticas para escolha do conjunto de regras a ser utilizado na classificação podem ser divididas em duas categorias as heurísticas a posteriori, que utilizam algum algoritmo de mineração de dados para gerar as regras e depois aplicam alguma restrição sobre elas para filtrá-las, e as heurísticas a priori, que focam no processo de geração de regras. As heurísticas a priori evitam gerar regras que não serão utilizadas na classificação; normalmente elas utilizam como hipótese a premissa de que a combinação de duas regras "boas" irá gerar uma nova regra "boa". A seguir, alguns exemplos de algoritmos de classificação que utilizam filtros de regras a posteriori.

- O algoritmo MMRFS, proposto no trabalho de Cheng et al. [2007], seleciona as regras de associação de maneira gulosa utilizando o critério de ganho de informa-

ção. Essa estratégia produz classificadores precisos, mas a quantidade de regras de associação do modelo resultante ainda é grande.

- No trabalho de Kirsch et al. [2009], foi proposta uma estratégia de filtro que utiliza testes estatísticos de hipótese para determinar se o suporte de uma regra possuiu significância estatística quando comparado com uma base de dados aleatória. Nessa estratégia, apenas são selecionadas as regras que possuem um desvio significante do suporte.
- O trabalho apresentado por Simon et al. [2011] propôs filtrar as regras de associação utilizando um modelo de classificação logística. As confianças das regras são fornecidas como entrada para o classificador e as regras que receberem peso zero pelo modelo de classificação logística são descartadas. Nos experimentos, os autores perceberam que as regras reduntantes ou independentes estatisticamente da classe tendem a ser descartadas.
- Vreeken et al. [2011] propuseram o algoritmo Krimp para selecionar as regras de associação que melhor comprimem a base de treino com base no princípio MDL. Nos experimentos, foi mostrado que essa estratégia é capaz de produzir classificadores precisos, mas não foi citada a quantidade de regras geradas por eles.

A seguir, alguns exemplos de algoritmos de classificação que utilizam filtros de regras a priori.

- O algoritmo PANDA, proposto por Lucchese et al. [2010], estabeleceu como selecionar as k melhores regras que descrevem uma base de treino, mesmo na presença de ruído. Essa estratégia é baseada em métodos de decomposição de matrizes binárias. O número de regras a serem geradas pelo algoritmo é um parâmetro, sendo que os autores configuraram o número de regras $k = 15$ em seus experimentos. O algoritmo se mostrou eficaz para encontrar as regras que melhor descrevem o treino, porém, na tarefa de classificação, seus resultados deixaram a desejar.
- Fidelis et al. [2000] propuseram uma estratégia baseada em algoritmos genéticos para gerar o conjunto de regras de associação. No algoritmo genético, cada indivíduo é representado por um vetor de cromossomos, sendo cada cromossomo composto por três campos: peso, operador e valor. O campo operador pode conter funções relacionais do tipo "$=$", "$\neq$", "$\geq$" e "$<$". A função de adequação (*fitness*) utilizada para avaliar a qualidade das regras era dada pelo produto da sensitividade e especificidade da regra na base de treino. No final da execução

do algoritmo, a melhor regra de cada classe é nomeada como sendo o padrão que irá classificar as instâncias daquela classe. O problema desse algoritmo é que apenas uma regra por classe pode não ser suficiente para realizar predições em bases complexas. Chamaremos esse algoritmo de GA-RULES para compará-lo com outros na seção de resultados.

- O algoritmo ORC (*Ordered Rules for Classification*), proposto por Chang et al. [2012], utiliza um método de otimização inteira mista multiobjetivo para gerar as regras de associação e posteriormente ordená-las. Os objetivos a serem maximizados são o suporte, a cobertura (*coverage*) e a alavancagem (*leverage*). Esse trabalho apresentou bons resultados, gerando modelos com poucas regras e precisos. O problema dessa solução é que, na prática, ela é inviável. Para problemas nos quais nossa estratégia gasta segundos para resolver, os trabalhos de otimização inteira levaram horas.
- Os trabalhos de Letham et al. [2012, 2013] apresentaram o algoritmo conhecido como BLM (*Bayesian List Machine*). O BML é um algoritmo a posteriori que primeiro gera um conjunto global de regras de associação, por exemplo, utilizando o algoritmo Apriori. Então, ele as ordena utilizando um método conhecido como amostragem de Monte Carlo de Cadeia de Markov, dando maior probabilidade para as regras com menos atributos. O tamanho da lista de decisão final é proporcional a um parâmetro p dado como entrada.

Nosso trabalho difere dos mencionados acima, uma vez que propomos uma estratégia nova para reduzir o número de regras que compõem o classificador final. Nós introduzimos um filtro a posteriori de regras, baseado no conceito de eficiência de Pareto, capaz de reduzir a complexidade do classificador, melhorando sua interpretabilidade sem prejudicar sua acurácia. Outra vantagem de nosso algoritmo é que ele utiliza um conjunto de regras específico para cada instância de teste. Das estratégias mencionadas acima, apenas Vreeken et al. [2011] utilizou um conjunto local de regras; os outros trabalhos utilizaram um conjunto de regras global baseado no treino. Normalmente, as estratégias que geram um conjunto global de regras não escalam bem e descartam regras úteis para a classificação. Além disso, nossa estratégia difere das estratégias propostas nos trabalhos de Lucchese et al. [2010] e Vreeken et al. [2011] ao focar na tarefa de classificação e não em encontrar padrões relevantes que expliquem a base do treino. Por fim, nosso filtro difere da estratégia proposta por Fidelis et al. [2000] no sentido que nós estamos interessados em explorar os compromissos entre a complexidade e a acurácia do classificador, estando de acordo com o princípio da Navalha de Occam. A estratégia proposta por Fidelis et al. [2000], por outro lado, utiliza

apenas uma regra por classe (o menor classificador possível) e, claramente, não possui garantia de que este número de regras compõe um conjunto ótimo, pois um classificador formado por mais regras poderia ser mais preciso.

Capítulo 3

Aprendendo Classificadores Interpretáveis por meio de Regras Pareto-Eficientes

Neste trabalho, propusemos um filtro a posteriori de regras de associação baseado na teoria de eficiência de Pareto. O papel principal desse filtro é reduzir a quantidade de regras utilizadas por um classificador associativo, a fim de aumentar sua legibilidade. Escolhemos um classificador associativo que possuísse as duas propriedades desejáveis mencionadas na seção 2.2.1, ou seja, aprende um conjunto local de regras específico para cada instância e decide o rótulo da instância por meio de mistura das regras (votação ponderada). No caso, o classificador escolhido foi o LAC (Veloso et al. [2006]). Diferente dos classificadores locais, que geram um conjunto de regras com base nas instâncias de treino, ele estende seu conjunto de regras $\mathcal{R}$ dinamicamente sob demanda para cada instância de teste x, ou seja, $\mathcal{R} = \{\mathcal{R}_{x_1} \cup \mathcal{R}_{x_2} \cup ... \cup \mathcal{R}_{x_n}\}$.

A figura 3.1 mostra o diagrama de execução do LAC. Iremos utilizá-la para mostrar em qual parte do processo de classificação pode ser utilizado um filtro a posteriori de regras, ou seja, o escopo desta pesquisa. O diagrama do LAC possui quatro componentes principais, que são as instâncias de teste (Base de Teste), as instâncias de treino (Base de Treino), o algoritmo responsável pela geração de regras de associação (Gerador de Regras) e o componente responsável por agregar as regras geradas, para uma instância e prever qual seu rótulo (Preditor). O Gerador de Regras utiliza a Base de Treino para estimar a qualidade das regras de associação geradas com base em um critério estatístico e, opcionalmente, utiliza um ou mais filtros a priori, como o suporte mínimo e o tamanho máximo da regra. Após as regras serem geradas, elas podem ser enviadas para o Preditor ou serem selecionadas por um ou mais filtros a posteriori,

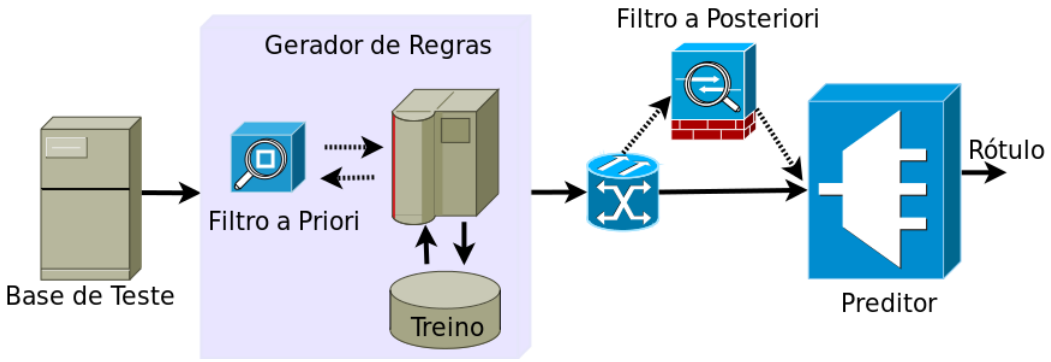


Figura 3.1: Diagrama de execução do LAC.

como, por exemplo, o filtro que utiliza a confiança mínima. De posse das regras, o Preditor decide qual é o rótulo da instância x_i por meio de votação ponderada, utilizando algum critério estatístico para informar os pesos das regras. Pode-se utilizar qualquer critério estatístico para ponderar as regras e, dependendo do domínio de aplicação, um critério será melhor que o outro. A escolha comum de critério estatístico para avaliar a utilidade das regras é a confiança. Neste trabalho, além da confiança, iremos utilizar o suporte, o valor adicionado ou Yules'Q (apresentados na seção 2.1.1) como critério estatístico na fase de computar a votação que cada rótulo recebeu.

Apesar de opcionais, o LAC não seria um algoritmo viável se não utilizasse um filtro a priori na etapa de geração das regras, pois existem aplicações que possuem centenas de atributos e o tempo gasto para gerar as regras para uma única instância seria proibitivo. Por exemplo, com base na tabela 2.4, para gerar as regras para uma instância de teste com mais de 40 atributos, seriam precisos pelo menos 13 dias. Além da eficiência na geração das regras, outro objetivo desejado de um filtro é que ele seja capaz de selecionar o melhor subconjunto de regras para serem utilizadas na classificação. Isso é possível porque nem todas as regras em $\mathcal{R}$ são boas para classificação, pois algumas delas predizem a classe errada. Ou seja, o conjunto de regras $\mathcal{R}$ é subótimo, já que podem existir outros subconjuntos que levam a predições mais precisas. Nosso objetivo neste trabalho é de aproximar ao máximo do menor subconjunto de regras possível.

3.1 Complexidade Versus Interpretabilidade

Nossa premissa básica neste trabalho é que a escolha de se utilizar um classificador menor leva a modelos mais precisos e interpretáveis em vários casos. Nossa busca por classificadores menores com a finalidade de melhorar sua acurácia foi inspirada por

estratégias de seleção de modelos como MDL e AIC (seção 2.4). Então, em nossa solução heurística, propomos aproximar os classificadores ótimos trocando complexidade por interpretabilidade. Mais precisamente, nós queremos encontrar, aproximadamente, o menor conjunto de regras $\mathcal{R}^*$ tão preciso quanto o conjunto original $\mathcal{R}$. Como iremos utilizar um classificador que gera o conjunto de regras de associação sob demanda, o problema de aprender o conjunto ótimo de regras será decomposto em vários subproblemas mais simples. Em vez de aprender um conjunto ótimo global, que é um problema mais difícil, procuramos aprender o menor conjunto local $\mathcal{R}^*_x$ que seja tão preciso quanto o conjunto $\mathcal{R}_x$ para cada instância de teste x.

3.2 Regras Pareto-Ótimas Multicritérios

Existem várias heurísticas que podem ser utilizadas para aproximar o melhor subconjunto de regras $\mathcal{R}^* = \{\mathcal{R}^*_{x_1} \cup \mathcal{R}^*_{x_2} \cup \ldots \cup \mathcal{R}^*_{x_k}\}$. Porém, além de a heurística ser efetiva, também queremos que ela seja rápida, caso contrário ela não será viável em situações reais. Para isso propomos avaliar a utilidade de cada regra considerando vários critérios estatísticos. Como mencionado na seção 2.1.1, cada critério possui um conjunto de propriedades e não existe um que seja melhor em todas as situações. Dependendo do domínio de aplicação, um critério será melhor do que o outro. Em nossa solução, queremos combinar as propriedades dos critérios de modo que elas se complementem. Para isso, modelamos cada regra como um ponto em um espaço n-dimensional $D^{\mathcal{CR}}$, sendo cada dimensão representada por um critério estatístico. Em nossa solução, estudamos espaços formados por subconjuntos dos critérios confiança, suporte, valor adicionado ou Yules'Q. Mas poderiam ser utilizados quaisquer outros critérios existentes para avaliar a utilidade das regras de associação. Dado que o classificador associativo original (LAC) já computava pelo menos uma dessas estatísticas, o custo de computar qualquer outra é desprezível, uma vez que não são necessários outros acessos aos dados da base de treino.

Para desenvolver nosso filtro, baseamo-nos na premissa de que as regras que se sobressaem em pelo menos um critério são as mais valiosas, no sentido de que carregam mais utilidade, do que as regras que não se sobressaem em nenhum critério. O conceito de eficiência de Pareto é uma maneira natural de explorar tal intuição, ou seja, regras que pertencem à fronteira de Pareto são as mais valiosas, dado que por definição, nenhuma regra pode se sobressair tanto quanto as regras na fronteira. Nós denotamos as regras na fronteira de Pareto como regras ótimas multicritérios. Em nossa solução, dada uma instância de teste x, o classificador correspondente $\mathcal{R}^*_x$ será composto pelas

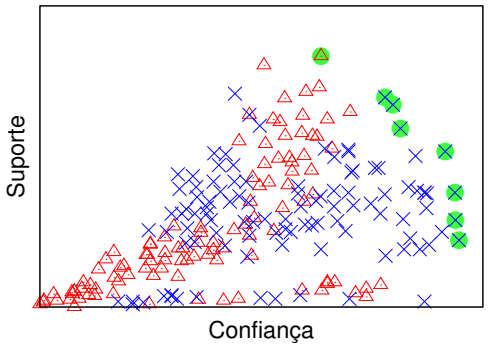


Figura 3.2: Cada ponto corresponde a uma regra em $\mathcal{R}_x$. As regras ótimas em $\mathcal{R}_x^*$ estão destacadas.

regras na fronteira de Pareto.

A figura 3.2 ilustra esta intuição com dados obtidos de um exemplo real. Cada ponto na figura corresponde a uma regra no conjunto $\mathcal{R}_x$. Ao olhar para a figura, não é evidente se as regras apontando para a classe azul (xis) são melhores que as regras apontando para classe vermelha (triângulo). Entretanto, se considerarmos apenas as regras na fronteira de Pareto, torna-se claro que as regras apontando para a classe azul, que é a classe correta, são as que mais se sobressaem. Além disso, enquanto o conjunto original $\mathcal{R}_x$ possui 210 regras, existem somente oito regras no conjunto $\mathcal{R}_x^*$, correspondendo a uma redução de 96% no tamanho do classificador.

3.3 Utilizando Fronteiras Adicionais

Em alguns casos, a fronteira de Pareto é composta por um conjunto muito pequeno de regras. Como consequência, o classificador $\mathcal{R}_x^*$ pode ser demasiado simplista, sendo arriscado realizar predições com essas regras. Isso ocorre porque algumas das regras do conjunto $\mathcal{R}_x^*$ podem ter sido geradas devido a ruídos na base. Nossa estratégia para contornar esse problema é utilizar fronteiras de Pareto adicionais. Ou seja, em vez de utilizar somente a primeira fronteira, também serão utilizadas as fronteiras subsequentes.

A figura 3.3 mostra um exemplo de três fronteiras de Pareto no espaço de critérios determinado pela confiança e o suporte. O número de fronteiras η a serem utilizadas pelo classificador é um parâmetro de entrada. Na prática, observamos que $\eta = 3$ é um bom número de fronteiras, pois, se a primeira fronteira for ruído, as duas próximas têm chance menor de o serem e suas regras irão favorecer a classe correta. O algoritmo 3, que chamamos de PE-LAC, Pareto-Eficiente - Classificação Associativa Sob Demanda

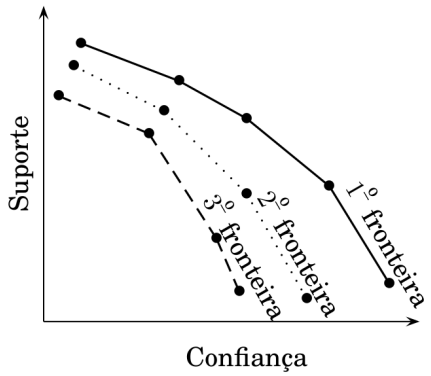


Figura 3.3: Fronteiras de Pareto.

(*Pareto-Efficient Lazy Associative Classification*), mostra os detalhes do processo inteiro de aprender classificadores interpretáveis utilizando um conjunto Pareto-Eficiente de regras. Foram adicionados dois parâmetros de entrada ao PE-LAC em relação ao LAC, que são o número de fronteiras de Pareto η a serem utilizadas para compor o classificador e o conjunto de dimensões $D^{\mathcal{CR}}$ que serão utilizadas para compor o espaço de utilidade.

Algoritmo 3 PE-LAC Pareto-Eficiente - Classificação Associativa Sob Demanda

Entrada: Conjunto de treino $\mathcal{S}$, conjunto de teste $\mathcal{T}$, suporte mínimo σ_{min}, confiança mínima θ_{min}, tamanho máximo da regra l, critério estatístico utilizado para ponderar os votos $\mathcal{CR}$, número de fronteiras η, dimensões utilizadas no filtro $D^{\mathcal{CR}}$.

1: **for all** instância $x_i \in \mathcal{T}$ **do**
2: $R_i \leftarrow$ ***induceRules***$(x_i, \mathcal{S}, \sigma_{min}, \theta_{min}, l)$ ▷ Apriori, Eclat ou DEclat
3: $F_i \leftarrow \emptyset$
4: **for** $i = 1$ **to** η **do**
5: $F_i \leftarrow F_i \cup$ ***paretoFrontier***$(\mathcal{R}_i - F_i, D^{\mathcal{CR}})$ ▷ utilizar BNL
6: **for all** classe $y_j \in \mathcal{Y}$ **do** ▷ $\mathcal{Y}$ conjunto de todas as classes
7: $s(x_i, y_j) \leftarrow \sum_{r \in F_i^{y_j}} \mathcal{CR}(r)$
8: $\mathcal{F}_s(x_i) = y_j \leftarrow \{y_j \in \mathcal{Y} : argmax(\ s(x_i, y_j)\ )\}$

3.4 Complexidade Computacional

A complexidade de tempo do LAC antes das alterações era de $\mathcal{O}(|\mathcal{T}| \times |\mathcal{S}| \times |\mathcal{Y}| \times \sum_{m=1}^{l} \binom{|x_i|}{m})$, sendo $\mathcal{T}$ a base de treino, $\mathcal{S}$ a base de teste, $\mathcal{Y}$ o conjunto de classes, l a restrição do maior tamanho de regra, x_i uma instância de teste e

$\sum_{m=1}^{l} \binom{|x_i|}{m}$ o número de regras geradas. Como mostrado na seção 2.5, o algoritmo BNL utilizado para calcular a fronteira de Pareto é linear no melhor caso e no caso médio. Logo, nesses casos, a complexidade do PE-LAC será: $\mathcal{O}(|\mathcal{T}| \times |\mathcal{S}| \times |\mathcal{Y}| \times \boldsymbol{\eta} \times \sum_{m=1}^{l} \binom{|x_i|}{m})$. Como η, normalmente, será uma constante pequena, ele pode ser retirado da fórmula e a complexidade do PE-LAC será igual à do LAC. No pior caso, o algoritmo BNL possui complexidade quadrática e a complexidade do PE-LAC passa a ser $\mathcal{O}(|\mathcal{T}| \times |\mathcal{S}| \times |\mathcal{Y}| \times \eta \times (\sum_{m=1}^{l} \binom{|x_i|}{m})^{\mathbf{2}})$.

Capítulo 4

Avaliação Experimental

Neste capítulo, analisamos empiricamente a estratégia PE-LAC em termos da acurácia na classificação e do tamanho do modelo (interpretabilidade). Vamos discutir a metodologia que utilizamos na avaliação dos modelos e, por fim, apresentar nossos resultados.

Tabela 4.1: Bases de Dados.

Base	#Atributos	#Instâncias	Frequência da Classe				
			0	1	2	3	4
UCI-austra	14	621	0,55	0,45	-	-	-
UCI-breast	10	629	0,66	0,34	-	-	-
UCI-cleve	13	272	0,55	0,45	-	-	-
UCI-crx	15	621	0,55	0,45	-	-	-
UCI-diabetes	8	691	0,64	0,36	-	-	-
UCI-german	20	900	0,70	0,30	-	-	-
UCI-heart	13	243	0,42	0,58	-	-	-
UCI-hepati	19	139	0,79	0,21	-	-	-
UCI-horse	22	331	0,36	0,64	-	-	-
UCI-ionosphere	34	315	0,65	0,35	-	-	-
Twitter-elections	[1;45]	66.643	0,30	0,70	-	-	-
Twitter-worldcup	[4;34]	3.215	0,80	0,20	-	-	-
Kaggle-titanic	8	891	0,62	0,38	-	-	-
Twitter-times	[8;29]	5.616	0,04	0,04	0,72	0,12	0,08

4.1 Metodologia de Avaliação

Nos experimentos, nós utilizamos 10 bases de referência do repositório da UCI (Newman et al. [1998]), três bases de aplicações de análise de sentimentos no Twitter (Silva et al. [2011a]) e uma base de dados do Kaggle (Goldbloom [2012]). A tabela 4.1 mostra as características dessas bases. As bases do repositório da UCI foram utilizadas para facilitar a comparação com outros algoritmos, pois elas já foram utilizadas em vários trabalhos (Veloso et al. [2006]; Letham et al. [2012]; Lucchese et al. [2010]). Essas bases são estruturadas, ou seja, todas as suas instâncias possuem um número fixo de atributos. Utilizamos as bases textuais coletadas do Twitter para avaliar bases com números variados de atributos. A base do Kaggle foi utilizada por possuir dados de fácil explicação. Nela, o objetivo é decidir se um dado passageiro do Titanic iria sobreviver ao desastre.

Os resultados reportados nos experimentos são médias de validação cruzada de cinco partes. A avaliação da classificação é verificada por meio das medidas convencionais: acurácia, precisão, revocação e F_1. A acurácia representa a proporção de instâncias de teste corretamente classificadas, considerando todas as classes. A precisão p é definida como a proporção de instâncias de teste corretamente classificadas de uma dada classe. A revocação r é definida como a proporção de instâncias corretamente classificadas com relação a todas as instâncias de uma dada classe. A medida F_1 é a combinação da precisão e da revocação determinada pela média harmônica $\frac{2pr}{p+r}$. Para obter uma única medida de F_1, vamos utilizar as médias Macro-F_1 e Micro-F_1 (Yang et al. [2002]). A medida Micro-F_1 corresponde à média das acurácias, enquanto a medida Macro-F_1 é a média dos valores F_1 em todas as classes. Por fim, a interpretabilidade do modelo será medida pelo número de regras utilizadas pelo classificador.

4.2 Testes de Hipótese

O objetivo da estratégia PE-LAC é conseguir uma redução significativa no conjunto de regras $\mathcal{R}$, mantendo taxas de acertos equivalentes (Micro-F_1 e Macro-F_1). Para verificar se o objetivo foi alcançado, utilizaremos três testes de hipóteses pareados, um paramétrico (Teste-T) e dois não paramétricos (Teste de Wilcoxon e Teste de Permutação), e dois testes de hipóteses entre grupos, um paramétrico (ANOVA) e um não paramétrico (Teste de Kruskal-Wallis). No teste de hipótese, são consideradas duas opções: $\boldsymbol{H_0}$, a hipótese nula e $\boldsymbol{H_1}$, a hipótese alternativa. A hipótese nula só é rejeitada se a chance de ela ocorrer for pequena (Wasserman [2010]). Para medir a evidência contra $\boldsymbol{H_0}$, será utilizado o p-valor. Ele é dado por p-valor $= \mathbb{P}(T > t_{obs}|H_o)$,

Tabela 4.2: Escala de p-valor tipicamente utilizada.

p-valor	Evidência
$< 0,01$	Evidência muito forte contra $\boldsymbol{H_0}$.
$0,05 - 0,10$	Fraca evidência contra $\boldsymbol{H_0}$.
$> 0,10$	Pouca ou nenhuma evidência contra $\boldsymbol{H_0}$.

sendo T uma estatística qualquer e t_{obs} o valor observado da estatística. Tipicamente pesquisadores têm utilizado a escala de p-valor mostrada na tabela 4.2. Quando $\boldsymbol{H_0}$ é verdadeira e é rejeitada, este erro é chamado erro do tipo I. O erro do tipo II acontece quando a hipótese $\boldsymbol{H_1}$ é verdadeira e $\boldsymbol{H_0}$ é aceita.

Os testes paramétricos (Teste-T e ANOVA) são mais poderosos que os não paramétricos (Teste de Wilcoxon, Teste de Permutação e Teste de Kruskal-Wallis) quando suas premissas são satisfeitas, caso contrário, os testes não paramétricos são melhores. As premissas dos testes paramétricos são relacionadas com a distribuição que gerou as estatísticas observadas. Em aprendizado de máquina, tipicamente, são avaliadas poucas bases (< 30) e as estatísticas utilizadas não são normalmente distribuídas. Por isso, é aconselhado utilizar os testes não paramétricos para comparar os classificadores (Demšar [2006]).

A tabela 4.3 apresenta algumas informações dos testes utilizados, dos quais os três primeiros são testes pareados e os dois últimos, testes de conjunto. Os testes pareados avaliam apenas um classificador contra outro e os de conjunto avaliam se em um conjunto de classificadores existe pelo menos um classificador que possui estatística diferente dos demais. Poderia ser utilizado o teste pareado combinando-se todos os classificadores dois a dois em vez do teste de conjunto, porém, isso eleva as chances de erro do tipo I (Demšar [2006]). Mais detalhes a respeito desses testes podem ser encontrados nos trabalhos de Demšar [2006] e Wasserman [2010].

4.3 Resultados

Nossa análise de resultados foi divida em duas partes. Na primeira, apresentamos resultados nos quais nós implementamos os modelos e executamos os experimentos. Nossa avaliação é baseada em uma comparação direta com o algoritmo original LAC (Veloso et al. [2006]). Além do LAC, também comparamos nosso filtro de regras com o filtro baseado no conjunto fechado de regras, que chamamos de CL-LAC (*Closed Lazy Associative Classification*), ou seja, comparamos a estratégia PE-LAC com uma estratégia baseada em representações condensadas. Vale a pena mencionar que a maio-

Tabela 4.3: Informações dos testes utilizados.

Teste	Informações
Teste-T Pareado	Necessita que a estatística utilizada siga uma distribuição normal. Melhor quando o tamanho da amostra é ≥ 30. Bem documentado e é o mais utilizado.
Teste de Wilcoxon	Não necessita que a estatística utilizada siga uma distribuição normal. Necessita que os dados pertençam à mesma população. Bom para amostras pequenas.
Teste de Permutação	Não necessita que a estatística utilizada siga uma distribuição normal. Bom para amostras pequenas.
ANOVA	Utilizado em comparação de um conjunto de classificadores. Necessita que a estatística utilizada siga uma distribuição normal. Requer que as variâncias dos classificadores sejam iguais.
Teste de Kruskal-Wallis	Utilizado em comparação de um conjunto de classificadores. Não necessita que a estatística utilizada siga uma distribuição normal.

ria dos resultados reportados neste trabalho também pode ser comparados com outros algoritmos, pois utilizamos as mesmas bases e a mesma metodologia do trabalho de Veloso et al. [2006]. Por fim, também avaliamos a efetividade da nossa estratégia quando aplicada em cadeia com o filtro de regras fechadas. Demos o nome a essa estratégia de CL-PE-LAC (*Closed Pareto-Efficient Lazy Associative Classification*).

Na segunda parte, apresentamos uma comparação da nossa estratégia com outros modelos apresentados na seção 2.6, mas, neste caso, nem os modelos comparados nem os experimentos foram realizados por nós. Apenas utilizamos os resultados reportados nos trabalhos correspondentes para comparar com os nossos.[1] Nos experimentos, foram utilizados o tamanho de regra máximo igual a 3 e suporte mínimo igual a 1 como filtros a priori nas execuções dos algoritmos de geração de regras. A menos que seja dito o contrário, os resultados envolvendo a quantidade de regras são médias dos tamanhos dos conjuntos de regras dos classificadores aprendidos para cada instância x_i. A seguir, vamos apresentar os experimentos que foram utilizados para determinar o número de fronteiras η a serem utilizadas e os critérios estatísticos que compõem o espaço de

[1] Os códigos desenvolvidos nos experimentos e as bases de dados utilizadas estarão disponíveis no Repositório de Aprendizado de Máquina da UFMG (Veloso & Dafe [2012]).

Tabela 4.4: PE-LAC: desempenho variando o número de fronteiras η.

	Objetivo	#Fronteiras (η)	UCI-austra	UCI-breast	UCI-cleve	UCI-crx	UCI-diabetes	UCI-german	UCI-heart	UCI-hepati	UCI-horse	UCI-ionosphere
Confiança	Micro-F_1(%)	1	82	95	80	82	69	71	83	82	78	91
		2	85	96	83	84	75	71	85	83	79	93
		3	86	97	83	86	78	72	85	83	82	94
		4	87	97	84	86	78	71	85	84	83	94
Confiança	#Regras	1	3	7	3	3	2	5	3	21	15	103
		2	5	8	5	6	6	6	5	24	17	107
		3	8	9	7	10	9	8	8	27	20	111
		4	11	11	10	13	12	10	10	31	23	115

utilidade $D^{\mathcal{CR}}$.

Número de Fronteiras

Nosso primeiro experimento tem como objetivo avaliar como o número de fronteiras η, utilizadas para compor o conjunto de regras a serem utilizadas pelo classificador, impacta na acurácia da classificação e no tamanho do classificador. Nesse experimento, nós utilizamos as bases de dados do repositório da UCI e variamos η de 1 a 4. Os resultados são mostrados na tabela 4.4. Para facilitar a análise, apenas mostramos os resultados no caso em que a confiança é utilizada como estatística para ponderar os votos das regras. Os piores números de Micro-F_1 aconteceram quando foi utilizada apenas uma fronteira, indicando que um conjunto pequeno de regras faz com que o classificador seja mais vulnerável a ruído. Pode-se perceber que, com o aumento do número de fronteiras, os valores da medida Micro-F_1 aumentam, ficando estáveis para $\eta = 3$. Além disso, ao aumentar o número de fronteiras, o número de regras também aumenta. Na prática, observamos que $\eta = 3$ é um bom número de fronteiras, pois, se a primeira fronteira for ruidoza, as duas próximas fronteiras possuem menor chance de possuírem ruídos, favorecendo a classe correta. Nos experimentos a seguir, iremos fixar $\eta = 3$.

Tabela 4.5: PE-LAC: desempenho para vários espaços de critérios (1=confiança, 2=suporte, 3=valor adicionado, 4=Yules'Q).

$\mathcal{D}$	Objetivo	Espaço de Critérios $D^{\mathcal{CR}}$										
		1,2	1,3	**1,4**	2,3	2,4	3,4	1,2,3	1,2,4	1,3,4	2,3,4	1,2,3,4
austra	#Regras	24	4	8	25	12	8	26	26	8	26	27
	Micro-F_1(%)	83	84	86	86	79	85	86	86	86	86	86
	Macro-F_1(%)	82	83	85	85	77	85	85	85	86	85	85
breast	#Regras	25	9	9	26	24	10	26	25	10	26	26
	Micro-F_1(%)	96	96	97	97	96	95	97	96	95	97	97
	Macro-F_1(%)	95	95	96	97	95	94	97	96	95	97	97
cleve	#Regras	32	5	7	34	23	7	37	36	7	35	37
	Micro-F_1(%)	79	82	83	81	82	81	81	80	82	80	81
	Macro-F_1(%)	78	81	83	80	81	80	80	79	81	79	80

Espaços de Critérios Estatísticos

Nosso segundo experimento tem como objetivo avaliar quais são os melhores critérios estatísticos para compor o espaço de utilidade $D^{\mathcal{CR}}$. A tabela 4.5 mostra os resultados para vários espaços de utilidade nas bases UCI-austra, UCI-breast e UCI-cleve. Foram apresentados valores de Macro-F_1, Micro-F_1 e o número de regras médio utilizado por instância. Utilizamos como opções de critérios para compor o espaço de utilidade a confiança, o suporte, o valor adicionado e Yules'Q. Para facilitar a análise, apenas mostramos resultados no caso que a confiança é utilizada como estatística para ponderar os votos das regras. De todas as configurações de espaço de utilidade, o melhor espaço foi composto pelos critérios confiança e Yules'Q. Juntos, esses dois critérios contêm todas as propriedades discutidas na seção 2.1.1. Existem outros espaços de utilidade que apresentaram valores equivalentes de Macro-F_1 e Micro-F_1, porém, eles possuem três ou quatro dimensões e, como esperado, suas fronteiras englobam um número maior de regras. Nos experimentos a seguir, iremos fixar $D^{\mathcal{CR}} = \{$ confiança, Yules'Q $\}$. As regras selecionadas nesse espaço de utilidade são aquelas em que os atributos possuem forte correlação positiva com as classes e possuem alta probabilidade a posteriori de prever a classe correta, dados os atributos.

Comparações com Outros Algoritmos

Nosso terceiro experimento tem como objetivo comparar a eficácia da nossa estratégia PE-LAC contra os algoritmos LAC e CL-LAC. Além disso, avaliamos a eficácia da utilização em cadeia do nosso filtro de regras Pareto-Eficiente com o filtro do con-

junto fechado de regras, algoritmo que chamamos de CL-PE-LAC. Os resultados de Macro-F_1, Micro-F_1 e o número de regras médio utilizado por instância são apresentados na tabela 4.6. A primeira coluna da tabela 4.6 indica qual foi o critério estatístico utilizado para ponderar as regras no momento de escolha do rótulo. As células escuras indicam quais algoritmos apresentaram os melhores resultados quando comparados com os outros. Uma análise rápida indica que os resultados obtidos pelo algoritmo PE-LAC estão de acordo com o nosso objetivo principal de aprender classificadores menores sem prejudicar suas taxas de acerto. Na maioria dos casos, nosso algoritmo PE-LAC conseguiu uma redução de até duas ordens de grandeza no número médio de regras quando comparado com o algoritmo LAC. Por exemplo, na base UCI-austra, a quantidade de regras foi reduzida de 210 para somente oito. O algoritmo CL-LAC apresentou um número de regras um pouco maior que o PE-LAC, porém, o CL-LAC apresentou piores resultados de Micro-F_1 e Macro-F_1 dentre os quatro algoritmos avaliados. Por fim, o algoritmo CL-PE-LAC gerou os menores modelos e obteve valores de Micro-F_1 e Macro-F_1 melhores do que o algoritmo CL-LAC. Isso indica que aplicar nosso filtro de Pareto em cadeia com o filtro do conjunto fechado foi uma boa solução. As regras selecionadas pelo algoritmo CL-PE-LAC são ótimas no espaço de utilidade $D^{\mathcal{CR}}$ e sem redundância com relação à cobertura das instâncias.

Uma análise mais profunda é necessária para verificar se o objetivo de reduzir o número de regras sem prejudicar a eficácia do classificador foi alcançado. Para isso, realizamos três testes de significância pareados (Teste-T, Wilcoxon e Permutação), mostrados na tabela 4.7. Foram avaliadas três estatísticas nos testes (Micro-F_1, Macro-F_1 e o número médio de regras), na comparação entre os algoritmos PE-LAC e LAC em cada critério estatístico (confiança, suporte, valor adicionado e Yules'Q). Os valores utilizados nos testes foram retirados da tabela 4.6. As hipóteses nulas dos testes são:

1. Os algoritmos LAC e PE-LAC aprendem classificadores com valores de Micro-F_1 equivalentes.
2. Os algoritmos LAC e PE-LAC aprendem classificadores com valores de Macro-F_1 equivalentes.
3. Os algoritmos LAC e PE-LAC aprendem classificadores compostos pela mesma quantidade de regras.

As hipóteses nulas 1 e 2 foram aceitas para todos os critérios estatísticos (i.e., os algoritmos LAC e PE-LAC aprendem classificadores com valores de Micro-F_1 e Macro-F_1 equivalentes), exceto para o suporte. No caso do suporte, existe uma chance fraca de que os classificadores aprendidos pelo LAC e o PE-LAC não possuam valores equiva-

lentes de Micro-F_1 e Macro-F_1. No caso, o algoritmo PE-LAC conseguiu os melhores valores. Isso ocorreu porque o suporte isoladamente não é um bom critério estatístico, pois ele apenas analisa a frequência de ocorrência da regra e não o seu poder de discriminação. Acreditamos que o nosso algoritmo PE-LAC apresentou resultados melhores que o LAC quando o suporte é utilizado para ponderar as regras porque o PE-LAC utilizou um espaço de utilidade composto pelos critérios de confiança e Yules'Q, sendo que estes critérios complementaram as propriedades do suporte. Em todos os testes, a hipótese nula 3 foi rejeitada, o que nos faz aceitar a hipótese alternativa, na qual os algoritmos LAC e PE-LAC aprendem classificadores compostos por quantidades diferentes de regras, indo de acordo com nossas expectativas. Vale observar que, apesar de o Teste-T ser paramétrico e não ser ideal para a situação em que existem poucos resultados, ele apresentou resultados semelhantes aos testes não paramétricos (Teste de Wilcoxon e Teste de Permutação).

Além dos testes pareados, também utilizamos testes de grupos (ANOVA e Kruskal-Wallis) para verificar as seguintes hipóteses:

1. O algoritmo LAC apresenta o mesmo desempenho em uma determinada métrica (Micro-F_1, Macro-F_1 ou número médio de regras) independentemente do critério estatístico utilizado para ponderar as regras na etapa de votação.
2. O algoritmo PE-LAC apresenta o mesmo desempenho em uma determinada estatística (Micro-F_1, Macro-F_1 ou número médio de regras) independentemente do critério estatístico utilizado para ponderar as regras na etapa de votação.
3. Ambos os algoritmos, LAC e PE-LAC, apresentam o mesmo desempenho em uma determinada métrica (Micro-F_1, Macro-F_1 ou número médio de regras) independentemente do critério utilizado para ponderar as regras na etapa de votação.

Os resultados dos testes de conjunto são mostrados na tabela 4.8. A hipótese nula 1 apresentou um chance alta de ser rejeitada ao considerar as métricas Micro-F_1 e Macro-F_1. Isso ocorreu nos experimentos em que se utilizou o LAC, pois o suporte sozinho não é um bom critério para ponderar regras, como explicado anteriormente. Já a hipótese nula 2 foi aceita ao considerar as métricas Micro-F_1 e Macro-F_1. Isso reforça nossa premissa de que, ao utilizar um espaço de utilidade no algoritmo PE-LAC, os critérios estatísticos complementam as propriedades uns dos outros. As hipóteses nulas 1 e 2 foram aceitas com 100% de certeza ao considerar o número de regras, porque o conjunto de regras dos algoritmos LAC ou PE-LAC não sofre modificação com a mudança da escolha do critério estatístico utilizado para ponderar as regras no momento da votação. Os resultados da hipótese nula 3 foram de acordo com os resultados da hipótese nula 1 ao

(a) LAC, #Regras=72 (b) PE-LAC, #Regras=4

Figura 4.1: Regras geradas para um passageiro sobrevivente do Titanic.

considerar as métricas Micro-F_1 e Macro-F_1. Ou seja, ela apresenta uma hipótese alta de ser rejeitada. Esse resultado era esperado, pois, se a hipótese nula 1 for rejeitada, a 3 também deve ser, devido ao fato de a hipótese 3 englobar os resultados das hipóteses 1 e 2. Já ao considerar o número de regras utilizadas pelos algoritmos LAC e PE-LAC, a hipótese nula 3 confirma nossa expectativa de que eles são estatisticamente diferentes. Como nos testes pareados, apesar de o teste paramétrico ANOVA não ser indicado para essa situação, ele apresentou resultados semelhantes ao teste não paramétrico de Kruskal-Wallis.

Apresentamos um exemplo para visualizar o ganho de interpretabilidade dos classificadores aprendidos pelo PE-LAC em relação ao LAC, mostrado na figura 4.1. Nesse exemplo, o objetivo é classificar se um dado passageiro do Titanic sobreviveu à tragédia. As regras em azul são votos indicando que ele sobreviveu e as em preto, que ele faleceu. O tamanho das regras é proporcional à sua confiança. A figura 4.1a mostra o conjunto de regras $\mathcal{R}_x$ gerado pelo algoritmo LAC e a figura 4.1b mostra o conjunto de regras $\mathcal{R}_x^*$ gerado pelo algoritmo PE-LAC. O algoritmo LAC gerou 72 regras para esta instância, sendo possível perceber uma predominância das regras que votam que o passageiro sobreviveu, o que de fato aconteceu. Porém, é difícil analisar o que levou a essa decisão por parte do classificador. Já o algoritmo PE-LAC selecionou apenas quatro das 72 regras utilizas pelo algoritmo LAC (uma redução de 94% no número de regras). Diante dessa redução da complexidade do modelo, é mais fácil compreender porque o classificador decidiu que o passageiro pertence ao grupo dos sobreviventes. No caso, as características marcantes foram ele estar hospedado em uma cabine, ter um irmão a bordo e ter por volta de 49 anos.

Por fim, iremos apresentar uma comparação dos algoritmos PE-LAC e CL-PE-LAC com o estado da arte em algoritmos associativos para classificação, mencionados na seção 2.6, que têm como objetivos serem precisos e interpretáveis. Os resultados apresentados na tabela 4.9 foram reportados nos respectivos trabalhos, portanto, os testes, em uma mesma base, foram executados em instâncias diferentes. Os resultados dos algoritmos PE-LAC e CL-PE-LAC foram retirados da tabela 4.6 ao utilizar o critério estatístico Yules'Q para ponderar a utilidade das regras. Os algoritmos PE-LAC e CL-PE-LAC aprendem um conjunto de regras local para cada instância e os outros algoritmos, um conjunto global baseado na base de treino. Devido a isso, a quantidade de regras reportadas para os algoritmos PE-LAC e CL-PE-LAC são números médios de regras geradas por instância, enquanto que, nos outros algoritmos, a quantidade de regras representa o tamanho do conjunto global de regras. O nosso algoritmo PE-LAC obteve bons valores de Macro-F_1 e Micro-F_1 quando comparado aos outros, porém, apresentou um número grande de regras nas bases UCI-hepati e UCI-ionosphere. O problema do número elavado de regras nessas bases foi solucionado pelo nosso outro algoritmo CL-PE-LAC, que o ataca ao remover as regras redundantes. O algoritmo PANDA gerou um conjunto de regras razoável, de tamanho 15 em todas as bases, porém, apresentou os piores resultados de Macro-F_1. O algoritmo GA-RULES gerou apenas duas regras na base UCI-breast, porém, apresentou o pior valor de Micro-F_1 nesta base (67% contra 97% do CL-PE-LAC). Isso indica que o algoritmo GA-RULES reduziu o tamanho do conjunto de regras além do que deveria. O algoritmo Krimp gerou os maiores conjuntos de regras, porém, não foi o que apresentou os melhores valores de Micro-F_1. Por fim, os algoritmos CL-PE-LAC, BLM e ORC foram os que apresentaram os melhores compromissos entre o tamanho do modelo versus sua acurácia. O problema do algoritmo ORC é devido ao seu desempenho computacional. Enquanto o algoritmo LAC gastou $1,9$ segundo para classificar as instâncias da base UCI-breast, conforme apresentado por Veloso et al. [2006], o algoritmo ORC gastou quatro horas, sendo que a base UCI-breast é considerada pequena. O algoritmo ORC apresenta elevado tempo computacional porque sua modelagem é baseada em programação inteira mista. O algoritmo BLM não reportou dados de tempo de execução. Já nossos algoritmos PE-LAC e CL-PE-LAC, na prática, apenas acrescentam um fator constante ao tempo de execução do LAC, ou seja, o número de fronteiras utilizadas, sendo que o LAC já mostrou ser eficiente em problemas reais e tem sido utilizado com sucesso no Observatório da Web[2], por exemplo, para acompanhar a evolução da dengue por meio de mensagens do Twitter.

[2]http://observatorio.inweb.org.br/

Tabela 4.6: Resultados de acurácia e tamanho do modelo/interpretabilidade. Células mais escuras indicam resultados melhores.

	Objetivo	Algoritmo	UCI-austra	UCI-breast	UCI-cleve	UCI-crx	UCI-diabetes	UCI-german	UCI-heart	UCI-hepati	UCI-horse	UCI-ionosphere	Twitter-elections	Twitter-worldcup	Kaggle-titanic	Twitter-times
Qualquer	#Regras	LAC	210	110	181	239	72	419	181	377	502	1181	201	197	70	1286
		CL-LAC	30	22	28	32	16	42	28	39	45	69	28	28	18	110
		PE-LAC	8	9	7	10	9	8	8	27	20	111	49	60	7	108
		CL-PE-LAC	6	4	7	6	6	7	6	8	6	4	6	8	5	9
Confiança	Micro-F_1(%)	LAC	84	97	82	83	70	70	84	79	74	90	94	96	75	95
		CL-LAC	81	96	79	80	65	70	81	79	65	89	87	96	71	83
		PE-LAC	86	97	83	86	78	72	85	83	82	94	96	96	77	96
		CL-PE-LAC	83	97	81	81	70	70	83	82	79	91	95	95	73	92
	Macro-F_1(%)	LAC	83	97	81	82	57	41	83	44	64	88	92	93	70	70
		CL-LAC	80	96	78	79	40	41	79	44	41	86	82	92	63	35
		PE-LAC	85	96	83	86	74	50	84	65	78	93	95	93	75	77
		CL-PE-LAC	82	97	80	79	57	46	82	55	75	89	93	93	67	59
Suporte	Micro-F_1(%)	LAC	80	93	80	80	65	70	81	79	66	79	72	96	70	72
		CL-LAC	72	89	75	72	64	70	74	79	64	75	70	80	65	72
		PE-LAC	86	95	82	85	69	70	82	78	83	92	89	81	78	82
		CL-PE-LAC	79	94	77	77	64	70	75	79	79	79	77	77	68	75
	Macro-F_1(%)	LAC	78	92	79	78	40	41	80	44	65	72	46	92	58	17
		CL-LAC	67	87	73	68	39	41	69	44	39	63	41	45	47	17
		PE-LAC	86	94	81	85	65	53	81	49	81	91	87	77	74	34
		CL-PE-LAC	77	93	75	75	89	41	72	44	77	72	69	69	54	22
Valor Adicionado	Micro-F_1(%)	LAC	86	97	84	86	73	69	86	77	74	91	94	94	77	96
		CL-LAC	85	96	85	84	72	66	84	78	73	90	92	94	72	94
		PE-LAC	87	96	84	86	77	72	86	81	80	91	96	95	79	96
		CL-PE-LAC	84	97	81	84	74	66	83	82	79	92	94	94	74	96
	Macro-F_1(%)	LAC	86	97	83	86	72	67	85	72	73	90	94	91	76	76
		CL-LAC	85	96	84	83	72	64	84	72	73	89	91	90	71	72
		PE-LAC	87	96	83	86	75	65	85	71	78	90	95	92	77	78
		CL-PE-LAC	83	97	81	83	73	64	82	73	78	91	93	93	73	77
Yules'Q	Micro-F_1(%)	LAC	86	97	83	86	75	73	85	81	74	91	96	95	78	97
		CL-LAC	86	97	84	85	75	67	86	81	71	88	94	93	75	95
		PE-LAC	86	97	83	86	74	72	85	81	82	94	96	95	78	96
		CL-PE-LAC	83	97	81	81	74	67	84	83	79	90	95	95	74	93
	Macro-F_1(%)	LAC	85	97	82	85	73	70	84	74	73	89	95	92	76	80
		CL-LAC	85	97	83	85	74	65	85	74	70	86	94	89	72	74
		PE-LAC	85	96	83	85	73	57	84	68	79	93	95	92	76	77
		CL-PE-LAC	83	97	80	80	73	65	83	73	77	88	94	94	71	65

Tabela 4.7: Testes pareados: LAC vs. PE-LAC.

$\mathcal{CR}$	p-valor(%)								
	Micro-F_1 (1)			Macro-F_1 (2)			#Regras (3)		
	Teste-T	Wilc.	Perm.	Teste-T	Wilc.	Perm.	Teste-T	Wilc.	Perm.
Confiança	42	38	42	29	31	28	0	0	0
Suporte	13	7	12	16	12	15	0	0	0
Valor Adi.	63	63	63	84	77	84	0	0	0
Yules'Q	86	96	86	82	100	81	0	0	0

Tabela 4.8: Testes de grupos: LAC, PE-LAC e ambos.

Hipótese	p-valor(%)					
	Micro-F_1		Macro-F_1		#Regras	
	ANOVA	Kruskal-W.	ANOVA	Kruskal-W.	ANOVA	Kruskal-W.
(1)	9	8	1	5	100	100
(2)	46	46	30	65	100	100
(3)	10	10	0	14	0	0

Tabela 4.9: Comparação com outros algoritmos a partir de resultados reportados.

Objetivo	Algoritmo	UCI-breast	UCI-heart	UCI-hepati	UCI-ionosphere	Kaggle-titanic
#Regras	PE-LAC*	9	8	27	111	7
	CL-PE-LAC*	4	6	8	4	5
	PANDA	15	15	15	15	15
	Krimp	30	79	-	164	-
	GA-RULES	2	-	-	-	-
	BLM	4	-	-	-	4
	ORC	13	-	-	-	3
Macro-F_1(%)	PE-LAC	97	85	81	94	78
	CL-PE-LAC	97	84	83	90	74
	PANDA	70	53	63	63	73
Micro-F_1(%)	PE-LAC	96	84	68	93	76
	CL-PE-LAC	97	83	73	88	71
	Krimp	94	62	-	91	-
	GA-RULES	67	-	-	-	-
	BLM	95	-	-	-	79
	ORC	95	-	-	-	79

Capítulo 5

Conclusão e Trabalhos Futuros

Neste trabalho, combinamos fundamentos de classificação associativa, critérios estatísticos, eficiência de Pareto e parcimônia com o objetivo de aprender classificadores interpretáveis e precisos. Utilizamos como base o algoritmo LAC, que representa o estado da arte em classificação associativa, e, seguindo o princípio da Navalha de Occam, mostramos que este algoritmo gera classificadores precisos, porém, desnecessariamente complexos. Então, como alternativa, propusemos o algoritmo PE-LAC, capaz de aprender classificadores mais simples que o LAC e com acurácia equivalente. O algoritmo PE-LAC utiliza um filtro a posteriori para selecionar as regras que irão compor o classificador. Esse filtro avalia cada regra em um espaço composto por um conjunto de critérios estatísticos, sendo que cada critério corresponde a uma dimensão. Exemplos de critérios estatísticos avaliados neste trabalho são confiança, suporte, valor adicionado e Yules'Q. Nossa premissa foi de que, ao combinar os critérios em um espaço de utilidade, suas propriedades se complementam. Para selecionar o conjunto ótimo de regras nesse espaço, foi aplicado um conceito central da economia conhecido como eficiência de Pareto.

Nos experimentos utilizando dados de referência assim como dados de aplicações recentes, ambos os algoritmos LAC e PE-LAC apresentaram acurácias equivalentes; porém, o algoritmo PE-LAC gerou classificadores até duas ordens de grandeza menores do que o LAC. Essa conclusão foi verificada por meio de testes de significância pareados e de conjunto. Nós também comparamos o algoritmo PE-LAC com o algoritmo CL-LAC, que filtra as regras com base no conjunto fechado. Nos resultados, o algoritmo CL-LAC apresentou as menores taxas de acerto e a quantidade média de regras utilizadas por instância foi um pouco maior do que a quantidade de regras utilizadas pelo PE-LAC. Uma estratégia alternativa foi aplicar o filtro de Pareto em cadeia com o filtro do conjunto fechado, que chamamos de CL-PE-LAC. O algoritmo CL-PE-LAC remove as

regras redundantes selecionadas pelo algoritmo PE-LAC. Ele gerou a menor quantidade de regras por instância, com taxas de acerto competitivas em relação ao algoritmo PE-LAC. Porém, em alguns casos, a taxa de acerto do algoritmo CL-PE-LAC foi menor do que a do PE-LAC, devido ao primeiro ter gerado um conjunto de regras excessivamente pequeno.

Além de comparar nossa solução com os algoritmos LAC e CL-LAC, também comparamos nossa estratégia com outras cinco: PANDA (decomposição de matrizes), Krimp (ganho de informação e MDL), GA-RULES (algoritmo genético), BLM (modelos bayesianos) e ORC (otimização inteira mista). Os algoritmos que apresentaram quantidade de regras e acurácia equivalentes aos algoritmos PE-LAC e CL-PE-LAC foram o BLM e ORC. Os outros algoritmos apresentaram taxas de acerto inferiores e o algoritmo Krimp gerou um número grande de regras. O problema do algoritmo ORC é que ele é inviável na prática, pois é baseado em programação inteira mista. Enquanto o algoritmo LAC gastou $1,9$ segundo para classificar as instâncias da base UCI-breast, o algoritmo ORC gastou 4 horas. O algoritmo BLM não reportou dados de tempo de execução. Já nossos algoritmos PE-LAC e CL-PE-LAC, na prática, apenas acrescentam um fator constante ao tempo de execução do LAC, ou seja, o número de fronteiras utilizadas, sendo que o LAC já mostrou ser eficiente em problemas reais.

Como primeiro trabalho futuro, nós pretendemos relaxar a restrição de dominância de Pareto para adicionar outras regras ao classificador final, além das regras da fronteira. Para alcançar esse objetivo, iremos utilizar outra teoria da economia conhecida como compensação de Kaldor-Hicks. A teoria da compensação de Kaldor-Hicks utiliza o mesmo espaço de utilidade da teoria de eficiência de Pareto, sendo que ela inclui no conjunto ótimo as soluções que são Pareto-Candidatas, além das soluções que compõem a fronteira de Pareto. Uma solução é Pareto-Candidata se ela puder compensar uma das suas dimensões de utilidade retirando de outra na qual ela foi melhor até se igualar a uma solução na fronteira de Pareto (Alchian [1988]). Ao utilizar a teoria de Kaldor-Hicks para selecionar as regras que irão compor o classificador, esperamos eliminar a necessidade de utilizar mais de uma fronteira para evitar sobreajustamento. Como segundo trabalho futuro, iremos propor um modo de estimar o tamanho de uma classe de equivalência das regras de associação. O resultado numérico da votação do CL-LAC, baseado no conjunto fechado, será igual ao do LAC se para cada regra utilizada pelo CL-LAC sua utilidade for multiplicada pelo tamanho do conjunto de equivalência que ela representa. Portanto, se for possível estimar o tamanho da classe de equivalência de maneira eficiente, o algoritmo CL-LAC apresentará uma acurácia idêntica à do LAC, utilizando um conjunto de regras menor e podendo ser mais eficiente computacionalmente. Além do algoritmo CL-LAC, essa solução beneficiará todos

os algoritmos que utilizam como base o conjunto fechado de regras, como, por exemplo, o algoritmo utilizado em recuperação da informação proposto por Pôssas et al. [2005]. Normalmente, os algoritmos que optam por utilizar o conjunto fechado trocam eficiência por acurácia. Se conseguirmos uma boa estimativa do tamanho da classe de equivalência, essa troca não será mais necessária.

Referências Bibliográficas

Agrawal, R.; Imielinski, T. & Swami, A. (1993). Database Mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5(6):914--925. ISSN 1041-4347.

Agrawal, R. & Srikant, R. (1994). Fast Algorithms for Mining Association Rules in Large Databases. Em *Proceedings of the International Conference on Very Large Data Bases*, pp. 487--499, Santiago de Chile, Chile. VLDB Endowment.

Alchian, A. (1988). *Proberty Rights*. Macmillan [u.a.].

Alpaydin, E. (2004). *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press. ISBN 0262012111.

Bayardo, Jr., R. J. & Agrawal, R. (1999). Mining the Most Interesting Rules. Em *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 145--154, San Diego, California, USA. ACM.

Bell, R. M.; Koren, Y. & Volinsky, C. (2010). A Perspective on the Netflix Prize. *Chance*, 23(1):24.

Blumer, A.; Ehrenfeucht, A.; Haussler, D. & Warmuth, M. (1987). Occam's razor. *Information Proceedingsessing Letters*, 24(6):377–380.

Börzsönyi, S.; Kossmann, D. & Stocker, K. (2001). The Skyline Operator. Em *Proceedings of the International Conference on Data Engineering*, pp. 421--430, Washington, DC, USA. IEEE Computer Society.

Breiman, L. (2001). Statistical Modeling: the two cultures. *Statistical Science*, 16(3):199--215.

Breiman, L.; Friedman, J.; Olshen, R. & Stone, C. (1984). *Classification and Regression Trees*. Wadsworth Intl.

Calders, T. & Goethals, B. (2007). Non-Derivable Itemset Mining. *Data Mining and Knowledge Discovery*, 14(1):171--206.

Chang, A.; Bertsimas, D. & Rudin, C. (2012). An Integer Optimization Approach to Associative Classification. Em Bartlett, P.; Pereira, F.; Burges, C.; Bottou, L. & Weinberger, K., editores, *Advances in Neural Information Processing Systems*, pp. 269--277.

Cheng, H.; Yan, X.; Han, J. & Hsu, C.-W. (2007). Discriminative Frequent Pattern Analysis for Effective Classification. Em *Proceedings of the International Conference on Data Engineering*, pp. 716–725, Istanbul, Turkey. IEEE.

Corne, D.; Knowles, J. & Oates, M. (2000). The Pareto Envelope-Based Selection Algorithm for Multi-objective Optimisation. Em *Proceedings of Parallel Problem Solving from Nature*, pp. 839–848, Dortmund, Germany.

Cortes, C. & Vapnik, V. (1995). Support-Vector Networks. *Machine Learning*, 20(3):273–297.

Cover, T. & Hart, P. (1967). Nearest Neighbor Pattern Classification. *IEEE Transactionson Information Theory*, 13(1):21–27.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *The Journal of Machine Learning Research*, 7:1--30. ISSN 1532-4435.

Domingos, P. (1999). The Role of Occam's Razor in Knowledge Discovery. *Data Mining and Knowledge Discovery*, 3(4):409–425.

Domingos, P. & Pazzani, M. (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning*, 29(2-3):103–130.

Fayyad, U. M. & Irani, K. B. (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. Em *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 1022–1029, Chambéry, France.

Fidelis, M. V.; Lopes, H. S.; Freitas, A. A. & Grossa, P. (2000). Discovering Comprehensible Classification Rules with a Genetic Algorithm. Em *Proceedings of the Evolutionary Computation*, pp. 805--810, La Jolla, CA, USA.

Gage, B.; Waterman, A.; Shannon, W.; Boechler, M.; Rich, M. & Radford (2001). Comparing Hospitals on Stroke Care: the need to account for stroke severity. *American Medical Association*, 285(1):2864--2870.

Gehrke, J.; Ganti, V.; Ramakrishnan, R. & Loh, W. (1999). Boat-Optimistic Decision Tree Construction. Em *Proceedings of the ACM SIGMOD International Conference on Management of Data Conference*, pp. 169–180, Philadelphia, USA.

Godfrey, P.; Shipley, R. & Gryz, J. (2007). Algorithms and Analyses for Maximal Vector Computation. *Very Large Databases*, 16(1):5--28. ISSN 1066-8888.

Goldbloom, A. (2012). Titanic: machine learning from disaster. `http://www.kaggle.com`.

Grünwald, P. & Langford, J. (2007). Suboptimal Behavior of Bayes and MDL in Classification Under Misspecification. *Machine Learning*, 66(2-3):119--149. ISSN 0885-6125.

Hahsler, M. & Chelluboina, S. (2013). Visualizing Association Rules: Introduction to the R-extension Package arulesViz. `http://cran.r-project.org/web/packages/arulesViz/`.

Hata, I.; Veloso, A. & Ziviani, N. (2013). Learning Accurate and Interpretable Classifiers Using Optimal Multi-Criteria Rules. *Information and Data Management*, 4(3):204--219.

Jerome, H. F. & Bogdan, E. P. (2008). Predictive Learning Via Rule Ensembles. *Data Mining and Knowledge Discovery*, 2:916–954.

Joachims, T. (2006). Training Linear SVMs in Linear Time. Em *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 217–226, Philadelphia, USA.

Jovanoski, V. & Lavrac, N. (2001). Classification Rule Learning with APRIORI-C. Em *Proceedings of the Portuguese Conference on Artificial Intelligence*, pp. 44--51, London, UK. Springer-Verlag.

Kirsch, A.; Mitzenmacher, M.; Pietracaprina, A.; Pucci, G.; Upfal, E. & Vandin, F. (2009). An Efficient Rigorous Approach for Identifying Statistically Significant Frequent Itemsets. Em *Proceedings of the ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 117--126, New York, NY, USA. ACM.

Lattimore, T. & Hutter, M. (2011). No Free Lunch versus Occam's Razor in Supervised Learning. *Computing Research Repository*, abs/1111.3846.

Lenca, P.; Meyer, P.; Vaillant, B. & Lallich, S. (2008). On Selecting Interestingness Measures for Association Rules: user oriented description and multiple criteria decision aid. *European Journal of Operational Research*, 184(2):610--626.

Leondes, C. (2002). *Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century*. Academic Press.

Letham, B.; Rudin, C.; McCormick, T. H. & Madigan, D. (2012). Building Interpretable Classifiers with Rules using Bayesian Analysis. Department of Statistics Technical Report tr609, University of Washington.

Letham, B.; Rudin, C.; McCormick, T. H. & Madigan, D. (2013). An interpretable stroke prediction model using rules and bayesian analysis. Em *Proceedings of the Artificial Intelligence*, Bellevue, Washington, USA.

Li, J.; Li, H.; Wong, L.; Pei, J. & Dong, G. (2006). Minimum Description Length Principle: generators are preferable to closed patterns. Em *Proceedings of the Artificial Intelligence*, pp. 409--414, Boston, Massachusetts, USA. AAAI Press.

Li, W.; Han, J. & Pei, J. (2001). CMAR: accurate and efficient classification based on multiple class-association rules. Em *Proceedings of the International Conference on Data Mining*, pp. 369--376, San Jose, California, USA. IEEE Computer Society.

Lim, W.; van der Eerden, M.; Laing, R.; Boersma, W.; Karalus, N.; Town, G.; Lewis, S. & Macfarlane, J. (2003). Defining Community Acquired Pneumonia Severity on Presentation to Hospital: an international derivation and validation study. *Thorax*, 58(5):377--382.

Liu, B.; Hsu, W. & Ma, Y. (1998). Integrating Classification and Association Rule Mining. Em *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 80–86, New York, USA.

Lowd, D. & Domingos, P. (2005). Naive Bayes Models for Probability Estimation. Em *Proceedings of the International Conference on Machine Learning*, pp. 529–536, Los Angeles, USA.

Lucchese, C.; Orlando, S. & Perego, R. (2010). Mining Top-K Patterns from Binary Datasets in presence of Noise. Em *Proceedings of the SIAM International Conference on Data Mining*, Columbus, OH, USA.

Madigan, D.; Mosurski, K. & Almond, R. (1997). Graphical Explanation in Belief Networks. *Computational and Graphical Statistics*, 160:181.

Marchand, M. & Sokolova, M. (2005). Learning with Decision Lists of Data-Dependent Features. *Journal of Machine Learning Research*, 6:427–451.

MD, E. M. A.; Marc Cohen, M.; Peter J. L. M. Bernink, M.; Carolyn H. McCabe, B.; Thomas Horacek, M.; Gary Papuchis, M.; Branco Mautner, M.; Ramon Corbalan, M.; David Radley, M. & Eugene Braunwald, M. (2000). The TIMI Risk Score for Unstable Angina Non-ST Elevation MIA Method for Prognostication and Therapeutic Decision Making. *American Medical Association*, 284(1):835--842.

Meira, W. & Zaki, M. (2011). *Fundamentals of Data Mining Algorithms*. Draft.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edição. ISBN 0070428077, 9780070428072.

Mramor, M.; Leban, G.; Demsar, J. & Zupan, B. (2007). Visualization-Based Cancer Microarray Data Classification Analysis. *Bioinformatics*, 23(16):2147–2154.

Nannen, V. (2003). The Paradox of Overfitting. Dissertação de mestrado, Rijksuniversiteit Groningen, the Netherlands.

Newman, D.; Hettich, S.; Blake, C. & Merz, C. (1998). UCI Repository of Machine Learning Databases. `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Pak, A. & Paroubek, P. (2010). Twitter as a Corpus for Sentiment Analysis and Opinion Mining. Em *Proceedings of The International Conference on Language Resources and Evaluation*, pp. 1320–1326, Valletta, Malta.

Palda, F. (2011). *Pareto's Republic and the new Science of Peace*. Cooper-Wolfling.

Piatetsky-Shapiro, G. (1991). Discovery, analysis and presentation of strong rules. Em Piatetsky-Shapiro, G. & Frawley, W. J., editores, *Knowledge Discovery in Databases*, pp. 229--248. AAAI Press.

Pôssas, B.; Ziviani, N.; Meira, Jr., W. & Ribeiro-Neto, B. (2005). Set-Based Vector Model: an efficient approach for correlation-based ranking. *Transactions on Information Systems*, 23(4):397--429. ISSN 1046-8188.

Quinlan, J. (1993). *C4.5: programs for machine learning*. Morgan Kaufmann, San Francisco, USA.

Quinlan, J. R. (1986). Induction of Decision Trees. *Machine Learning*, 1(1):81--106. ISSN 0885-6125.

Ribeiro, M. T.; Lacerda, A.; Moura, E.; Hata, I.; Veloso, A. & Ziviani, N. (2014). Multi-Objective Pareto-Efficient Approaches for Recommender Systems. *Transactions on Intelligent Systems and Technology*, 5(1):Forthcoming Issue.

Rivest, R. (1987). Learning Decision Lists. *Machine Learning*, 2(3):229–246.

Shmueli, G. (2010). To Explain or to Predict? *Statistical Science*, 25(3):289--310.

Silva, I. S.; Gomide, J.; Veloso, A.; Meira, Jr., W. & Ferreira, R. (2011a). Effective Sentiment Stream Analysis with Self-Augmenting Training and Demand-Driven Projection. Em *Proceedings of the Special Interest Group on Information Retrieval*, pp. 475--484, Beijing, China. ACM.

Silva, R.; Gonçalves, M. A. & Veloso, A. (2011b). Rule-Based Active Sampling for Learning To rank. Em *Proceedings of European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 240--255, Berlin, Heidelberg. Springer-Verlag.

Simon, G. J.; Kumar, V. & Li, P. W. (2011). A Simple Statistical Model and Association Rule Filtering for Classification. Em *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 823--831, San Diego, California, USA. ACM.

Sober, E.; Forster, M.; Hausman, D.; Lewis, P.; Lewontin, R. & Stee, M. (2002). Instrumentalism, Parsimony and the Akaike Framework. *Philosophy of Science*, 69(s3):S112--S123.

Tan, P.-N.; Kumar, V. & Srivastava, J. (2002). Selecting the Right Interestingness Measure for Association Patterns. Em *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 32--41, Edmonton, Alberta, Canada. ACM.

Thomas, J. J. & Cook, K. A. (2005). *Illuminating the Path: the research and development agenda for visual analytics* . National Visualization and Analytics Ctr. ISBN 0769523234.

Vaillant, B.; Lallich, S. & Lenca, P. (2006). Modeling of the Counter-Examples and Association Rules Interestingness Measures Behavior. Em *Proceedings of the International Conference on Data Mining*, pp. 132–137, Las Vegas, USA. CSREA Press.

Vellido, A.; Martín-Guerrero, J. D. & Lisboa, P. J. (2012). Making Machine Learning Models Interpretable. Em *Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium.

Vellido, A.; Martín-Guerrero, J. D.; Rossi, F. & Lisboa, P. J. G. (2011). Seeing is Believing: the importance of visualization in real-world machine learning applications. Em *Proceedings of The European Symposium on Artificial Neural Networks*, pp. 219–226, Bruges, Belgium.

Veloso, A. & Dafe, J. (2012). UFMG respository of machine learning. `https://code.google.com/p/machine-learning-dcc-ufmg/`.

Veloso, A. & Meira, W. (2011). *Demand-Driven Associative Classification*. Springer Briefs in Computer Science. Springer. ISBN 9780857295248.

Veloso, A.; Meira Jr., W. & Zaki, M. J. (2006). Lazy Associative Classification. Em *Proceedings of the Sixth International Conference on Data Mining*, pp. 645--654, Hong Kong, China. IEEE Computer Society.

Veloso, A.; Zaki, M. J.; Jr., W. M. & Gonçalves, M. A. (2009). The Metric Dilemma: Competence-Conscious Associative Classification. Em *Proceedings of the SIAM International Conference on Data Mining*, pp. 918–929. SIAM.

Vreeken, J.; Leeuwen, M. & Siebes, A. (2011). Krimp: mining itemsets that compress. *Data Mining and Knowledge Discovery*, 23(1):169--214. ISSN 1384-5810.

Wasserman, L. (2010). *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated. ISBN 1441923225, 9781441923226.

Wilkins, D. C. & Ma, Y. (1994). The Refinement of Probabilistic Rule Sets: sociopathic interactions. *Artificial Intelligence*, 70(1-2):1--32. ISSN 0004-3702.

Yahia, S. B.; Hamrouni, T. & Nguifo, E. M. (2006). Frequent Closed Itemset Based Algorithms: a thorough structural and analytical survey. *SIGKDD Explorations Newsletter*, 8(1):93--104.

Yang, Y.; Slattery, S. & Ghani, R. (2002). A Study of Approaches to Hypertext Categorization. *Journal of Intelligent Information Systems*, 18(2-3):219–241.

Yin, X. & Han, J. (2003). CPAR: classification based on predictive association rules. Em *Proceedings of the Third SIAM International Conference on Data Mining*, San Francisco, USA.

Zahálka, J. & Zelezný, F. (2011). An Experimental Test of Occam's Razor in Classification. *Machine Learning*, 82(3):475–481.

Zaki, M. J. & Gouda, K. (2003). Fast Vertical Mining Using Diffsets. Em *Proceedings of the SIGKDD International Conference on Knowledge Discovery and Data ining*, pp. 326--335, New York, NY, USA. ACM.

Zaki, M. J. & Hsiao, C.-J. (2005). Efficient Algorithms for Mining Closed Itemsets and Their Lattice Structure. *Transactions on Knowledge and Data Engineering*, 17(4):462--478.

Zaki, M. J.; Parthasarathy, S.; Ogihara, M. & Li, W. (1997). New Algorithms for Fast Discovery of Association Rules. Relatório técnico, University of Rochester, Rochester, NY, USA.

Ziviani, N. (1993). *Projeto de Algoritmos: com implementações em Pascal e C*. CENGAGE. ISBN 9788522110506.