Victor Anselmo Silva

*Redes Lógicas de Markov Aplicadas ao
Aprendizado de Classificadores Automáticos de
Dados*

São Paulo
2010

Victor Anselmo Silva

*Redes Lógicas de Markov Aplicadas ao
Aprendizado de Classificadores Automáticos de
Dados*

Dissertação apresentada à Escola Politécnica
da Universidade de São Paulo para obtenção
do Título de Mestre em Engenharia Mecâ-
nica.

São Paulo
2010

Victor Anselmo Silva

*Redes Lógicas de Markov Aplicadas ao
Aprendizado de Classificadores Automáticos de
Dados*

Dissertação apresentada à Escola Politécnica
da Universidade de São Paulo para obtenção
do Título de Mestre em Engenharia Mecâ-
nica.

Área de concentração:
Engenharia de Controle e Automação Mecâ-
nica

Orientador:
Prof. Dr. Fabio Gagliardi Cozman

São Paulo
2010

Dedicatória

A minha filha Beatriz, que me trouxe motivação e razão extras para tudo, a todo
momento.

Agradecimentos

São muitos os responsáveis por mais este alvo atingido. Meus agradecimentos são para todos. Em especial, agradeço:

- A Deus, pela vida;
- A minha companheira Verônica por todo amor e compreensão demonstrados com tanta dedicação, e a nossa filha pequena Beatriz, que nos trouxe alegria e renovou nosso ânimo com a vivacidade do mundo infantil;
- A minha mãe Marisa e ao meu irmão Heitor, pelo incentivo constante com amor e orações. E ao meu avô Samuel, pela preocupação com minha formação e pelo afeto demonstrado;
- Aos meus sogros Ricardo e Tania, pelo suporte nos momentos delicados e constante incentivo em todas as minhas empreitadas;
- Ao professor Fabio Cozman, sempre atencioso, pela oportunidade de pesquisa que me concedeu, pela paciência, orientação e compreensão nestes anos de trabalho;
- Aos membros do Laboratório de Tomada de Decisão, durante o período em que estive integralmente dedicado a esta dissertação: Rodrigo Polastro, Denis Mauá, Ricardo Shirota, Daniel Kikuti e José Ochoa Luna. Obrigado por suas contribuições e sugestões, técnicas, filosóficas, ou brincalhonas, quer em um almoço no bandeirão da USP, quer em uma pausa para um café na lanchonete;
- Ao amigo de graduação, pós-graduação e de república Fábio Henrique de Assis, pelas inúmeras conversas nas caminhadas rumo a Poli e na volta dela também ao fim do dia (e muitos deles dias longos). Pelas conversas sobre os temas mais diversos durante os cafês;
- Aos professores, membros da banca, Newton Maruyama e Leliane Barros, pelas sugestões pertinentes;
- À empresa Hewlett-Packard (HP) pela concessão da bolsa de mestrado.

“Há homens que lutam um dia e são bons.
Há outros que lutam um ano e são melhores.
Há os que lutam muitos anos e são muito bons.
Porém, há os que lutam toda a vida.
Esses são os imprescindíveis.”

Bertold Brecht

Resumo

Sistemas de computação têm se tornado maiores e mais complexos com o objetivo de lidar com a vasta quantidade de dados disponíveis. Uma tarefa decisiva em tais sistemas é classificar estes dados, bem como extrair informação útil destes. Nesta dissertação, testam-se as *redes lógicas de Markov* como linguagem para especificação e aprendizado de classificadores automáticos de dados. Esta linguagem combina fragmentos da lógica de primeira ordem e modelos probabilísticos gráficos (redes de Markov) em uma única representação. A junção destas duas técnicas permite a modelagem de conhecimento relacional através da lógica, e também de incertezas por meio de probabilidades e grafos não-direcionados.

Neste trabalho, classificadores são aprendidos segundo dois paradigmas de aprendizado de máquina: o supervisionado, foco desta dissertação, e também o aprendizado semi-supervisionado com restrições determinísticas. Para investigar a utilidade das redes lógicas de Markov no treinamento de classificadores, uma série de experimentos de aprendizado é desenvolvida a partir de bases de dados de treino reais disponíveis em repositórios na internet. Como ferramenta auxiliar nos experimentos, esta dissertação testa também o pacote *Alchemy*, que provê um conjunto de algoritmos para tarefas gerais de aprendizado de máquina e inferência probabilística em redes lógicas de Markov.

Para mensurar o desempenho dos classificadores aprendidos, três métricas tradicionais são empregadas: acurácia, precisão e revocação. Os resultados alcançados com classificadores semi-supervisionados com restrições indicam que a linguagem ainda não é própria para este paradigma de aprendizado. Por outro lado, o êxito obtido no desempenho dos classificadores treinados de forma supervisionada sugere que as redes lógicas de Markov são um formalismo lógico-probabilístico promissor para aplicações de classificação, e devem ser objeto de pesquisas futuras.

Palavras-chave: redes lógicas de Markov, classificadores, aprendizado supervisionado, semi-supervisionado com restrições.

Abstract

Computing systems have become larger and more complex in order to deal with the vast amount of available data. An important task in such systems is to classify these data, so as to extract useful information from them. In this dissertation, *Markov logic networks* are tested as a language to specify and learn automatic data classifiers. This language combines fragments of first-order logic and probabilistic graphical models (Markov networks), in a single representation. Together, both techniques allow one to model relational knowledge through a logic formalism, and uncertainty through probabilities and undirected graphs.

In this work, data classifiers are learned by two machine learning paradigms: the supervised, the main focus of this dissertation, and also the semisupervised learning under deterministic constraints. To investigate the usefulness of Markov logic networks in training data classifiers, a set of experiments is developed from real databases available in repositories at the internet. As a support tool for experiments, this dissertation tests also the *Alchemy* package, which provides a set of algorithms for general machine learning tasks and probabilistic inference in Markov logic networks.

To measure the performance of data classifiers, three traditional metrics are employed: accuracy, precision and recall. The results reached with semisupervised data classifiers indicate that the language is not yet suitable for learning based on this paradigm. On the other hand, the success achieved with classifiers trained in a supervised context suggests that Markov logic networks are a promising logical-probabilistic formalism to approach classification applications, and should be considered in future research.

Keywords: Markov logic networks, classifiers, supervised, semisupervised under constraints, learning.

Lista de Figuras

1.1	Exemplo de rede de Markov construída a partir de uma rede lógica de Markov.	p. 18
2.1	Exemplo de rede de Markov.	p. 34
2.2	Esquema ilustrativo de uma validação cruzada com 5-repetições.	p. 39
2.3	Rede de Markov construída após a instanciação da rede lógica de Markov da Tabela 2.2.	p. 43
2.4	Exemplo de rede de Markov proposicionalizada para um problema hipotético de classificação.	p. 46
3.1	Quantidade de instâncias de treino e teste para cada base de dados utilizada nos experimentos.	p. 71
3.2	Fluxograma de um processo de aprendizado de um classificador com RLMs.	p. 78
3.3	Desempenho do SMLC especificado segundo o <i>Modelo 1</i> de aprendizado para a base <i>Adult</i>	p. 81
3.4	Desempenho do SMLC especificado segundo o <i>Modelo 2</i> de aprendizado para a base <i>Adult</i>	p. 81
3.5	Desempenho do SMLC especificado segundo o <i>Modelo 3</i> de aprendizado para a base <i>Adult</i>	p. 82
3.6	Desempenho do SMLC especificado segundo o <i>Modelo 4</i> de aprendizado para a base <i>Adult</i>	p. 82
3.7	Desempenho do SMLC especificado segundo o <i>Modelo 1</i> de aprendizado para a base <i>Breast Cancer</i>	p. 82
3.8	Desempenho do SMLC especificado segundo o <i>Modelo 2</i> de aprendizado para a base <i>Breast Cancer</i>	p. 83
3.9	Desempenho do SMLC especificado segundo o <i>Modelo 3</i> de aprendizado para a base <i>Breast Cancer</i>	p. 83

3.10	Desempenho do SMLC especificado segundo o <i>Modelo 4</i> de aprendizado para a base <i>Breast Cancer</i>	p. 83
3.11	Desempenho do SMLC especificado segundo o <i>Modelo 1</i> de aprendizado para a base <i>Iris</i>	p. 84
3.12	Desempenho do SMLC especificado segundo o <i>Modelo 2</i> de aprendizado para a base <i>Iris</i>	p. 84
3.13	Desempenho do SMLC especificado segundo o <i>Modelo 3</i> de aprendizado para a base <i>Iris</i>	p. 84
3.14	Desempenho do SMLC especificado segundo o <i>Modelo 4</i> de aprendizado para a base <i>Iris</i>	p. 85
3.15	Comparação (parte 1) de desempenho do SMLC (<i>Modelo 3</i>) com métodos centrados em regras de associação.	p. 85
3.16	Comparação (parte 2) de desempenho do SMLC (<i>Modelo 3</i>) com métodos centrados em regras de associação.	p. 86
3.17	Comparação (parte 1) de desempenho do SMLC (<i>Modelo 3</i>) com métodos centrados em árvores de decisão.	p. 86
3.18	Comparação (parte 2) de desempenho do SMLC (<i>Modelo 3</i>) com métodos centrados em árvores de decisão.	p. 86
3.19	Comparação (parte 1) de desempenho do SMLC (<i>Modelo 3</i>) com métodos centrados principalmente em redes Bayesianas.	p. 87
3.20	Comparação (parte 2) de desempenho do SMLC (<i>Modelo 3</i>) com métodos centrados principalmente em redes Bayesianas.	p. 87
3.21	Desempenho global do classificador SMLC aprendido para as bases <i>Adult</i> , <i>Breast Cancer</i> e <i>Iris</i>	p. 88
3.22	Desempenho global do classificador SMLC, aprendido segundo o <i>Modelo 1</i> , <i>Modelo 2</i> , <i>Modelo 3</i> e <i>Modelo 4</i>	p. 89
3.23	Tempos de aprendizado do SMLC para (a) <i>Modelo 1</i> , (b) <i>Modelo 2</i> , (c) <i>Modelo 3</i> e (d) <i>Modelo 4</i> para as bases <i>Breast Cancer</i> , <i>Adult</i> e <i>Iris</i> . . .	p. 90
3.24	Desempenho do SMLC considerando diversos níveis de conhecimento prévio introduzido na modelagem do classificador com RLMs.	p. 94

3.25	Tempo de treinamento do SMLC para diversos níveis de conhecimento.	p.96
3.26	Desempenho do SSMLC aprendido para a base <i>Iris</i> para um cenário semi-supervisionado com 10% dos dados rotulados.	p.99
3.27	Desempenho do SSMLC aprendido para a base <i>Iris</i> para um cenário semi-supervisionado com 20% dos dados rotulados.	p.99
3.28	Comparação de valores de acurácia entre o SSMLC e outros métodos não-supervisionados, empregando-se a base <i>Iris</i>	p.100

Lista de Tabelas

2.1	Exemplo de uma base de conhecimento de primeira ordem. Os predicados Am , Fu e Ca são formas abreviadas de Amigos , Fumante e Cancer	p. 28
2.2	Exemplo de especificação de uma rede lógica de Markov.	p. 42
2.3	Exemplo de rede lógica de Markov para um problema hipotético de classificação.	p. 45
3.1	Bases de dados utilizadas nos testes de classificação.	p. 70

Lista de Siglas

BCPO Base de Conhecimento de Primeira Ordem

EM *Expectation Maximization*

LPO Lógica de Primeira Ordem

MAP *Maximum a Posteriori*

MCMC *Markov Chain Monte Carlo*

RM Rede de Markov

RLM Rede Lógica de Markov

SMLC *Supervised Markov Logic Classifier*

SSMLC *Semi-supervised Markov Logic Classifier*

WEKA *Waikato Environment for Knowledge Analysis*

Sumário

1	Introdução	p. 16
1.1	Motivação	p. 19
1.2	Objetivo e Principais Contribuições	p. 20
1.3	Organização do Texto	p. 21
2	Revisão da Literatura e Teoria	p. 23
2.1	Literatura Básica	p. 23
2.2	Teoria Básica	p. 26
2.2.1	Bases de Conhecimento de Primeira Ordem	p. 26
2.2.2	Fundamentos da Teoria de Probabilidades	p. 29
2.2.3	Redes de Markov	p. 33
2.2.4	Aprendizado de Máquina	p. 35
2.2.5	Classificadores e Métricas de Avaliação de Desempenho	p. 38
2.2.6	Redes Lógicas de Markov	p. 41
3	Redes Lógicas de Markov Aplicadas ao Aprendizado de Classificadores	p. 60
3.1	Especificando Classificadores com Redes Lógicas de Markov	p. 60
3.1.1	Classificadores Supervisionados	p. 60
3.1.2	Classificadores Semi-supervisionados com Restrições	p. 64
3.2	Metodologia, Experimentos e Discussões de Resultados	p. 67
3.2.1	Exemplo de Aprendizado de um Classificador com <i>Alchemy</i>	p. 71

3.2.2	Esquema Geral Proposto para Aprendizado de Classificadores com RLMS	p. 77
3.2.3	Resultados e Discussão de Testes de RLMS em Aprendizado de Classificadores Supervisionados	p. 79
3.2.4	Resultados e Discussão de Testes de RLMS em Aprendizado de Classificadores Semi-supervisionados com Restrições	p. 97
4	Conclusão e Trabalhos Futuros	p. 103
	Referências Bibliográficas	p. 107

1 *Introdução*

Nesta dissertação testam-se as *redes lógicas de Markov*, ou simplesmente *lógica de Markov*, como uma linguagem para especificação de classificadores automáticos de dados. Esta linguagem, proposta no trabalho de Richardson e Domingos (2006), combina fragmentos da lógica de primeira ordem e uma classe de modelos probabilísticos gráficos, chamada redes de Markov.

Com a lógica de primeira ordem, pode-se representar grande variedade de domínios de conhecimento, ao passo que com modelos gráficos probabilísticos pode-se tratar a incerteza de forma eficiente em tais domínios. Em muitas aplicações, estes dois aspectos podem ser extremamente relevantes. Por exemplo, os problemas de classificação coletiva e predição de *links* podem ser modelados com a linguagem¹ redes lógicas de Markov (RICHARDSON; DOMINGOS, 2006).

Em classificação coletiva, a meta é inferir probabilisticamente a classe de certo objeto considerando não só o conhecimento de seus atributos, mas também o conhecimento da classe de outros objetos relacionados a este. Neste contexto, torna-se conveniente especificar relações de dependência entre objetos de classes distintas, de modo que a informação que se tem sobre a classe de um deles auxilie na inferência da classe do outro.

Em predição de *links* também são realizadas inferências probabilísticas, no entanto, o objetivo é determinar se existe alguma relação entre dois objetos distintos a partir do conhecimento das propriedades de ambos e, possivelmente, a partir do conhecimento de outras relações de dependência diretas e/ou indiretas entre ambos.

Um exemplo disso seria, em um domínio acadêmico, descobrir se **Fabio** é o orientador de doutorado de **Rodrigo**, dado que se conhece algum outro relacionamento entre eles (uma publicação conjunta, por exemplo). Problemas como classificação coletiva e pre-

¹Talvez, referir-se a redes lógicas de Markov com o termo “linguagem” cause certo desconforto inicialmente, porém esta é a terminologia usual adotada em diversos trabalhos da literatura, dado que redes lógicas de Markov envolvem tanto lógica de primeira ordem, que é uma linguagem de representação de conhecimento, e modelos probabilísticos, os quais também são formas de representação empregando, no entanto, grafos que se assemelham a redes.

dição de *links* exemplificam situações em que a lógica de primeira ordem, associada ao tratamento de incertezas de forma probabilística, pode ser útil na modelagem de conhecimento.

Com redes lógicas de Markov pode-se especificar, de forma simples e compacta, relações entre objetos e também sentenças lógicas complexas. Além disso, o aspecto probabilístico é tratado em redes lógicas de Markov através de aprendizado de pesos que codificam probabilidades para as sentenças lógicas especificadas². Mais precisamente, uma rede lógica de Markov é um conjunto de sentenças (ou fórmulas) ponderadas, escritas em lógica de primeira ordem.

Quando aplicada a um domínio de dados (por exemplo, pessoas, imagens), uma rede lógica de Markov possibilita a construção de um grafo não-direcionado do tipo rede de Markov (PEARL, 1988), a partir do qual pode-se executar inferências probabilísticas para classificar um objeto deste domínio de dados.

Como exemplo de uma rede lógica de Markov, sejam duas relações em lógica de primeira ordem: $\text{PaiDe}(\mathbf{x}, \mathbf{y})$ com o significado “ $\mathbf{x}$ é pai de $\mathbf{y}$ ”, e $\text{Irmãos}(\mathbf{z}, \mathbf{w})$ com o significado “ $\mathbf{z}$ e $\mathbf{w}$ são irmãos”, em que $\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$ e $\mathbf{w}$ são variáveis que podem assumir valores de um domínio de pessoas. A partir destas relações, pode-se especificar uma sentença lógica associada a um peso α , aprendido a partir de dados ou informado por um especialista, o qual codifica a probabilidade de uma fórmula ser verdadeira. Uma rede lógica de Markov envolvendo esta fórmula seria

$$\alpha \quad \text{PaiDe}(\mathbf{x}, \mathbf{y}) \wedge \text{PaiDe}(\mathbf{x}, \mathbf{z}) \Rightarrow \text{Irmãos}(\mathbf{y}, \mathbf{z}),$$

a qual permite modelar o conhecimento de que se as pessoas $\mathbf{y}$ e $\mathbf{z}$ tem um mesmo pai $\mathbf{x}$ (o símbolo $\wedge$ significa uma conjunção entre as relações), então pode-se afirmar que $\mathbf{y}$ e $\mathbf{z}$ são irmãos. A Figura 1.1 ilustra a rede de Markov que pode ser construída a partir desta rede lógica de Markov, considerando um domínio de pessoas em que as variáveis $\mathbf{x}$, $\mathbf{z}$ e $\mathbf{w}$ assumem valores constantes $\mathbf{X}$, $\mathbf{Z}$ e $\mathbf{W}$. A substituição destes valores na fórmula do exemplo produz a sentença instanciada³

$$\alpha \quad \text{PaiDe}(\mathbf{X}, \mathbf{Z}) \wedge \text{PaiDe}(\mathbf{X}, \mathbf{W}) \Rightarrow \text{Irmãos}(\mathbf{Z}, \mathbf{W}),$$

que, através de um procedimento descrito na Seção 2.2.6, permite a construção da rede de Markov desta figura, em que os nós (por exemplo, $\text{PaiDe}(\mathbf{X}, \mathbf{Z})$) tem valor binário e

²Na Seção 2.2.6 descreve-se como estimar tais pesos e como convertê-los em valores de probabilidade.

³Uma sentença ou uma relação é instanciada quando suas variáveis são substituídas por constantes.

relações instanciadas. Em um grafo como este, pode-se executar inferência probabilística para saber, por exemplo, se X é pai de W , sabendo-se que X é pai de Z e que Z e W são irmãos. Em outras palavras, deseja-se classificar a relação $\text{PaiDe}(X, W)$ como verdadeira ou falsa. Este é um exemplo em que uma rede lógica de Markov pode ser usada para aprender um classificador.

Figura 1.1: Exemplo de rede de Markov construída a partir de uma rede lógica de Markov.

Ao testar as redes lógicas de Markov em aprendizado de classificadores, esta dissertação foca principalmente o paradigma supervisionado de aprendizado. Secundariamente, alguns testes em aprendizado semi-supervisionado com restrições determinísticas também são desenvolvidos.

No aprendizado supervisionado, treina-se um classificador a partir de um conjunto de objetos totalmente rotulados, i.e., objetos ou exemplos cuja classe a que pertencem está previamente especificada. No aprendizado não-supervisionado, não há qualquer forma de supervisão. O objetivo é descobrir padrões interessantes em um grupo de objetos sem no entanto, conhecer qualquer informação sobre seus rótulos.

No aprendizado semi-supervisionado com restrições determinísticas, tais rótulos estão pré-especificados apenas para uma parcela reduzida dos dados, ao passo que uma porção muito maior não está rotulada. Além desta parcela rotulada pequena, alguma supervisão é incorporada também sob a forma de restrições sobre pares de objetos, proporcionando conhecimento que pode ajudar na inferência de um rótulo de um objeto. Por exemplo, restrições determinando que dois objetos devem (ou não devem) ter rótulos iguais.

Como ferramenta auxiliar ao aprendizado de classificadores, emprega-se um pacote livremente disponível denominado *Alchemy* (KOK *et al.*, 2005), o qual oferece um conjunto de algoritmos para aprendizado de pesos e inferência probabilística em redes de Markov construídas a partir de redes lógicas de Markov. Esta ferramenta tem sido muito empregada em tarefas de aprendizado e inferência em domínios inerentemente incertos e com estrutura relacional complexa.

Além de utilizar o pacote *Alchemy* em experimentos de classificação, esta dissertação também tem como objetivo testá-lo e avaliá-lo mediante o desempenho de alguns de seus algoritmos em problemas de aprendizado. Para isso, duas formas de aprendizado de pesos

disponíveis no *Alchemy* são testadas: generativa e discriminativa. Também são testados três métodos de inferência implementados no pacote. Um deles baseia-se puramente no Método de Monte Carlo via Cadeias de Markov, mais conhecido como MCMC (*Markov Chain Monte Carlo*) (GILKS *et al.*, 1996). Outro combina fundamentos de MCMC e satisfatibilidade em lógica, esse método é conhecido como MC-SAT (POON; DOMINGOS, 2006). E o terceiro método infere um rótulo com base em estimativas de máximo posterior ou, mais comumente conhecido como *Maximum a Posteriori* (MAP), que calcula o valor mais provável de certa variável.

Para verificar o desempenho de redes lógicas de Markov no aprendizado de classificadores, as Seções 3.2.3 e 3.2.4 apresentam resultados e discussões para alguns experimentos desenvolvidos principalmente com classificadores aprendidos de forma supervisionada, e também alguns testes preliminares para aprendizado semi-supervisionado com restrições determinísticas.

Os domínios de dados empregados para testar os classificadores aprendidos estão livremente disponíveis em um repositório na internet, chamado *UCI Machine Learning Repository*⁴. Neste repositório, há diversas bases de dados reais próprias para a execução de testes de técnicas de aprendizado de máquina.

Os resultados alcançados no aprendizado supervisionado sugerem que o emprego de uma linguagem que combina lógica de primeira ordem e modelos probabilísticos gráficos para a construção de classificadores automáticos de dados deve ser tema de pesquisas futuras. No entanto, no campo semi-supervisionado com restrições, os resultados sinalizam que a lógica de Markov ainda não é totalmente adequada para o treinamento de classificadores.

1.1 Motivação

Esta dissertação é movida pelo interesse em investigar as redes lógicas de Markov, linguagem nova na área de aprendizado e que tem sido pouco explorada para a construção de classificadores. A flexibilidade desta linguagem e a sua capacidade de combinar lógica de primeira ordem e modelos probabilísticos gráficos em uma única representação podem colaborar para um tratamento adequado de muitos aspectos críticos em problemas reais de aprendizado.

Dentre esses aspectos, pode-se citar hipóteses de amostras de dados independentes e

⁴Endereço eletrônico: <http://archive.ics.uci.edu/ml/>

identicamente distribuídos, as quais proporcionam simplificações de modelagem e reduzem o custo computacional, mas colaboram para resultados menos apurados. Outro aspecto importante é a desconsideração de possíveis estruturas relacionais intrínsecas no domínio de conhecimento estudado.

Além disso, a lógica de Markov é uma linguagem proposta exatamente com o intuito de capturar conhecimento relacional em um domínio sem, no entanto, desconsiderar a existência de incerteza nesse conhecimento. Assim, esta linguagem parece adequada para aprendizado de classificadores em domínios cujos dados estão fortemente atrelados por dependências probabilísticas e também por estruturas relacionais.

Somando-se a isto, algoritmos para aprendizado e inferência a partir de redes lógicas de Markov têm sido implementados em uma ferramenta aberta chamada pacote *Alchemy* (KOK *et al.*, 2005), que tem sido utilizada com sucesso por pesquisadores da área de aprendizado estatístico relacional (RICHARDSON; DOMINGOS, 2006; SINGLA; DOMINGOS, 2007; GETOOR; TASKAR, 2007). Isto faz desta ferramenta uma opção relevante para utilização em problemas de classificação modelados com fragmentos relacionais da lógica de primeira ordem e também com probabilidades.

Desta forma, o emprego de lógica de Markov para aprendizado de classificadores mostra-se interessante, dadas as características desta linguagem, as quais possibilitam realizar a abordagem de problemas em diversos domínios de conhecimento, tanto aqueles puramente probabilísticos, como também os probabilístico-relacionais.

1.2 Objetivo e Principais Contribuições

Esta dissertação tem dois objetivos. O objetivo primário é testar a utilidade da lógica de Markov como linguagem para aprendizado de classificadores automáticos de dados em alguns domínios de conhecimento. O objetivo secundário é testar, também, o desempenho do pacote de algoritmos para aprendizado e inferência em redes lógicas de Markov chamado *Alchemy* (KOK *et al.*, 2005), empregando-o como ferramenta auxiliar no aprendizado de pesos e na inferência probabilística a partir dos classificadores treinados.

Em aprendizado supervisionado, algumas propostas de modelos de aprendizado são testadas por meio de experimentos com bases de dados reais disponíveis na internet. Quanto ao treinamento semi-supervisionado com restrições, o modelo testado é uma adaptação de um dos modelos propostos para o aprendizado supervisionado. Nesta adaptação, são incluídas fórmulas lógicas de restrições e relações de transitividade. Os classificadores

aprendidos em ambos os paradigmas têm seu desempenho mensurado por três métricas comumente empregadas na avaliação de métodos de classificação: acurácia, precisão e revocação.

As principais contribuições deste trabalho de mestrado são:

- **Contribuição 1:** o estudo de redes lógicas de Markov e a constatação de que esta recente linguagem lógico-probabilística é promissora para aprendizado de classificadores automáticos, especialmente os supervisionados, em diversos domínios de dados. Por outro lado, conclui-se também que na vertente semi-supervisionada com restrições, as redes lógicas de Markov ainda carecem de mecanismos mais desenvolvidos para o treinamento de classificadores;
- **Contribuição 2:** a constatação de que alguns algoritmos do pacote *Alchemy* para aprendizado de pesos, tanto de forma generativa como discriminativa, e para inferência probabilística, em especial os métodos MCMC e MC-SAT, são satisfatórios para alguns problemas de aprendizado supervisionado de classificadores modelados com redes lógicas de Markov.

Por fim, é importante também ressaltar alguns tópicos que esta dissertação não se propõe a tratar. Em primeiro lugar, foge ao escopo deste trabalho estender a lógica de Markov com novas implementações e/ou melhorias que possam elevar o desempenho desta linguagem em classificação. Esta dissertação também não compreende a implementação de novos algoritmos para o pacote *Alchemy*, nem promove adaptações nestes, visando um desempenho computacional superior. Tanto a lógica de Markov como o pacote *Alchemy* são apenas testados nesta dissertação, como arcabouços teórico e ferramental, respectivamente, para o aprendizado de classificadores.

1.3 Organização do Texto

No Capítulo 2 constam revisões sobre a literatura relevante no contexto de aprendizado de classificadores e no uso conjunto de lógica e probabilidades, e também sobre a teoria básica para compreensão deste trabalho. Nesta revisão teórica são apresentados os conceitos de lógica de primeira ordem, probabilidades, redes de Markov, aprendizado de máquina, classificadores, e redes lógicas de Markov. Uma breve explanação sobre o pacote *Alchemy* também é feita ao final deste capítulo.

O Capítulo 3 é a contribuição maior desta dissertação. Discute-se na Seção 3.1 como especificar classificadores com redes lógicas de Markov. A Seção 3.2 descreve a metodologia experimental realizada, os testes, resultados e as discussões destes.

O Capítulo 4 é a conclusão. Nele comenta-se também sobre algumas possibilidades de trabalhos futuros.

2 Revisão da Literatura e Teoria

Este capítulo faz duas revisões breves. A primeira delas, que consta na Seção 2.1, trata da literatura envolvida em classificação e relacionada à combinação de lógica e probabilidades. A intenção desta revisão não é descrever em profundidade o estado-da-arte nestas áreas, tampouco descrever em sua totalidade a enorme gama de métodos de classificação e técnicas lógico-probabilísticas para tratamento de domínios relacionais em que há incerteza nos dados. O principal objetivo é revisar alguns assuntos relevantes para este trabalho, relacionados com classificação em geral.

A segunda revisão, feita na Seção 2, é sobre a teoria básica de redes lógicas de Markov e trata de alguns conceitos considerados pertinentes ao tema desta dissertação.

2.1 Literatura Básica

Uma das técnicas que podem ser empregadas em classificação é a lógica proposicional ou lógica booleana (RUSSELL; NORVIG, 1995). Nela tem-se uma linguagem simples que consiste em símbolos de proposições e conectivos lógicos. Tal linguagem permite a manipulação de proposições que são verdadeiras ou falsas. Consequentemente, pode-se classificar, apenas por meio de uma inferência lógica, o valor de novas proposições a partir de uma base de conhecimento contendo sentenças em lógica proposicional.

Contudo, a principal desvantagem que a lógica proposicional apresenta é a restrição de que apenas “fatos” podem ser manipulados. A linguagem não oferece mecanismos para lidar com a existência de objetos e de relações entre estes objetos. Isto reduz drasticamente sua expressividade e capacidade de representar domínios relacionais complexos de forma concisa. Por consequência, aplicações tais como classificação com base em regras e em estruturas relacionais tornam-se muito restritas.

Para superar esta limitação, a lógica de primeira ordem oferece uma representação muito mais ampla, dada sua capacidade de modelar fatos, objetos, relações e funções

aplicadas a estes objetos (GENESERETH; NILSSON, 1987). Em lógica de primeira ordem, o conhecimento é modelado por meio de sentenças com valor lógico verdadeiro ou falso. Em outras palavras, não há tolerância a conhecimento incerto sobre proposições, objetos ou sobre relações. Existem muitos problemas reais em que a incerteza acerca de uma proposição ou relação é algo de extrema importância (RUSSELL; NORVIG, 1995).

Como exemplo destes problemas, em medicina, nem sempre se pode considerar um diagnóstico como verdade absoluta, mesmo quando os profissionais em questão são especialistas. Para contornar esta barreira da intolerância ao conhecimento incerto, o raciocínio com base em probabilidades é aplicado em diversas áreas, particularmente em problemas de classificação em medicina, tal como em diagnóstico de enfermidades com base em evidências (sintomas) observadas (ADAMS *et al.*, 1974).

Contudo, muitos sistemas fundamentados em probabilidades, criados para suporte à decisão em medicina, são vulneráveis a dados não-representativos provenientes de amostras de tamanho insuficiente (COWELL *et al.*, 1999). Consequentemente, os resultados de classificação (diagnóstico) nem sempre são seguros. Tais sistemas também não apresentam um formalismo adequado para tratar informações de independência que, em muitos casos, podem simplificar um modelo e reduzir a quantidade de parâmetros necessários para a realização de inferências. A desvantagem nisso é a necessidade de armazenar e processar enormes tabelas de dados sobre probabilidades.

Uma forma de superar estas dificuldades se dá pelo emprego de técnicas que usam probabilidades e grafos para aprendizado a partir de dados, tais como as redes Bayesianas (JENSEN, 2001) e as redes de Markov (PEARL, 1988), ambos amplamente aplicados em classificação de dados (HERBRICH, 2002). Estas abordagens grafo-probabilísticas são capazes de modelar o conhecimento sobre um domínio incerto codificando relações de independência e independência condicional, tornando as representações mais compactas e reduzindo o processamento computacional necessário, por exemplo, para a realização de inferências.

Entretanto, boa parte destas técnicas apóia-se em hipóteses de amostras de dados independentes e identicamente distribuídos para fins de simplificação de modelagem e de computação (DEVROYE *et al.*, 1996; MCLACHLAN; PEEL, 2000). Tal suposição em geral é errônea, e certamente reduz a acurácia dos classificadores gerados por estes métodos, dado que relações importantes de dependência são desprezadas. Assim, a literatura atual tem revelado novas metodologias de aprendizado que buscam minimizar a ou mesmo evitar premissas de independência entre variáveis.

Esta meta tem sido alcançada com o desenvolvimento de linguagens capazes de tratar o conhecimento através de representações não apenas probabilísticas, mas também baseadas no poder de representação e expressividade da lógica. As tentativas de combinar lógica e probabilidades em aprendizado podem ser observadas nos trabalhos de Nilsson (1986), Bacchus (1990) e Halpern (1990). Na mesma época aproximadamente, muitos autores começaram a empregar programas lógicos para especificar de forma compacta modelos gráficos probabilísticos como redes Bayesianas complexas. Esta combinação tornou-se conhecida como construção de modelos baseados em conhecimento (do inglês *Knowledge-based Model Construction*) (WELLMAN *et al.*, 1992).

Atualmente, uma área emergente, denominada *aprendizado estatístico relacional*, engloba combinações de subconjuntos da lógica de primeira ordem e de probabilidades. Nela o objetivo é representar, raciocinar e aprender em domínios com estrutura relacional complexa e probabilística rica (GETOOR; TASKAR, 2007). Algumas linguagens lógico-probabilísticas que se incluem nesta nova área são os modelos probabilísticos relacionais (FRIEDMAN *et al.*, 1999), os programas lógico-estocásticos (MUGGLETON, 1996), os programas lógico-Bayesianos (KERSTING *et al.*, 2000), redes de dependência relacional (NEVILLE; JENSEN, 2004) e as redes lógicas de Markov (RICHARDSON; DOMINGOS, 2006), que podem ser vistas como uma generalização das linguagens lógico-probabilísticas citadas anteriormente.

As redes lógicas de Markov são a linguagem mais desenvolvida até o momento para aplicações de aprendizado de máquina, pois disponibilizam a lógica de primeira ordem como formalismo para modelagem de conhecimento intrinsecamente relacional e modelos gráficos probabilísticos do tipo redes de Markov, como formalismo para o tratamento da incerteza nesse conhecimento. Nessa linha, e considerando aplicações de classificação, não são muitas as publicações encontradas. No entanto, algumas podem ser citadas.

A primeira delas é o trabalho de Lippi e Frasconi (2009), no qual redes lógicas de Markov foram empregadas para classificação supervisionada de estruturas secundárias em proteínas. Neste trabalho, o problema foi modelado como uma tarefa de predição de *links*, na qual o interesse estava em estimar probabilidades para predicados-*query* representativos de possíveis “ligações” entre certos componentes de uma cadeia de proteínas. As conclusões reportadas pelo artigo afirmam que redes lógicas de Markov melhoraram significativamente a acurácia dos resultados.

Outro trabalho é o de Richardson e Domingos (2006), os precursores de redes lógicas de Markov. Na parte experimental deste artigo o objetivo era predizer, dado um universo

acadêmico de alunos e professores, que aluno era orientado por qual professor. Um modelo em redes lógicas de Markov foi conceitualizado a partir de diversos tipos de sentenças, tais como “estudantes não são professores”, “um estudante tem no máximo um orientador”, “no máximo um autor de uma publicação é um professor” ou ainda “estudantes em primeira fase do programa de pós-graduação não têm orientador”. Foram investigadas também técnicas puramente probabilísticas e puramente lógicas como alternativas de solução do problema. A linguagem Redes lógicas de Markov proporcionou os melhores resultados de classificação.

2.2 Teoria Básica

Por ser uma linguagem que combina lógica relacional e modelos probabilísticos gráficos, as redes lógicas de Markov pressupõem um conhecimento introdutório sobre lógica de primeira ordem, teoria de probabilidades e grafos, em especial, redes de Markov. Estes três assuntos são abordados nas Seções 2.2.1, 2.2.2 e 2.2.3, respectivamente.

Como este trabalho envolve a área de aprendizado de classificadores automáticos, é natural abordar o tema geral aprendizado e, também, a definição de um classificador. Estes dois assuntos são tratados nas Seções 2.2.4 e 2.2.5, respectivamente. Por fim, na Seção 2.2.6 são apresentados os conceitos de redes lógicas de Markov, e de aprendizado e inferência probabilística a partir desta linguagem. Como um dos objetivos desta dissertação é testar também o pacote *Alchemy* como ferramenta de suporte ao aprendizado e inferência em redes lógicas de Markov, suas funcionalidades principais são apresentadas ao final da Seção 2.2.6.

2.2.1 Bases de Conhecimento de Primeira Ordem

Nesta seção são introduzidos alguns conceitos relevantes sobre bases de conhecimento de primeira ordem, os quais dão suporte ao entendimento de lógica de primeira ordem, formalismo fundamental na especificação de classificadores por meio da linguagem redes lógicas de Markov.

Uma *base de conhecimento de primeira ordem* (BCPO) é definida como um conjunto de fórmulas em lógica de primeira ordem (LPO) (GENESERETH; NILSSON, 1987). Para a construção de fórmulas, podem ser empregados quatro tipos de símbolos: constantes, variáveis, funções e predicados. Constantes representam objetos em um domínio de interesse. Por exemplo, um domínio de pessoas: **Paulo**, **Marcos**, **Marcela**. Variáveis assumem

valores de objetos no domínio, e podem ser de um tipo específico. Por exemplo, uma variável **pessoa** pode assumir apenas valores do domínio: **Pedro**, **Ana**, **Caio**.

Funções representam mapeamentos de tuplas de objetos para objetos. Como exemplo, a função **PaiDe(filho)** indica quem é o pai de uma pessoa representada pela variável **filho**. Predicados especificam relações entre objetos de um domínio ou atributos de objetos. Por exemplo, o predicado **Amigos(pessoa₁, pessoa₂)** estabelece que duas pessoas, representadas pelas variáveis **pessoa₁** e **pessoa₂**, são amigas. Já o fato de uma pessoa ser fumante pode ser representado pelo predicado **Fumante(pessoa)**. Uma *interpretação* especifica quais objetos, funções e relações no domínio são representados por quais símbolos.

Em uma BCPO, um *termo* é qualquer expressão que representa um objeto no domínio. Um termo pode ser uma constante, uma variável, ou uma função aplicada a uma tupla de termos. Por exemplo, são termos: **Paulo**, **x** e **MaiorDivisorComum(x, y)**. Uma *fórmula atômica* (ou simplesmente *átomo*) é definida como um predicado aplicado a uma tupla de termos. Por exemplo, **Amigos(pessoa, PaiDe(Marcela))**. Fórmulas podem ser recursivamente construídas a partir de átomos usando conectores lógicos e quantificadores. Se F_1 e F_2 são fórmulas, também são fórmulas: $\neg F_1$ (negação), $F_1 \wedge F_2$ (conjunção), $F_1 \vee F_2$ (disjunção), $F_1 \Rightarrow F_2$ (implicação), $F_1 \Leftrightarrow F_2$ (equivalência), $\forall x F_1$ (quantificação universal) e $\exists x F_1$ (quantificação existencial). Pode-se usar parênteses para reforçar precedência.

Um *literal positivo* é um átomo, um *literal negativo* é um átomo negado. Fórmulas em uma BCPO formam implicitamente uma grande conjunção, de forma que uma BCPO pode ser vista como uma única grande fórmula. Um *termo proposicionalizado* é um termo que não contém qualquer variável. Um *átomo proposicionalizado* é um átomo em que todos os argumentos são termos proposicionalizados. Um *mundo possível* é composto por um conjunto de objetos, um conjunto de funções (mapeamentos de tuplas de objetos para objetos), e um conjunto de relações satisfeitas entre estes objetos; junto com uma interpretação, eles determinam o valor verdade de cada átomo proposicionalizado (GENESERETH; NILSSON, 1987).

Uma fórmula é *satisfazível* se existe pelo menos um mundo em que ela é verdadeira. O problema ordinário de inferência em LPO consiste em verificar se uma fórmula F é verdadeira em todos os mundos em que uma BCPO é verdadeira. Esta verificação é comumente feita por *refutação*: uma BCPO admite F se $BCPO \cup \neg F$ não pode ser satisfeita. Quando o objetivo é automatizar a inferência, convém converter as fórmulas

de uma BCPO para uma notação mais regular, chamada forma clausal ou *forma normal conjuntiva*.

Uma BCPO em forma normal conjuntiva é uma conjunção de *cláusulas*, em que uma cláusula é uma disjunção de literais. Toda BCPO pode ser convertida para forma clausal por meio de uma sequência definida de passos, sobre a qual o leitor encontra detalhes em Genesereth e Nilsson (1987). A Tabela 2.1, adaptada de Richardson e Domingos (2006), mostra uma BCPO simples e sua conversão para forma normal conjuntiva.

Tabela 2.1: Exemplo de uma base de conhecimento de primeira ordem. Os predicados *Am*, *Fu* e *Ca* são formas abreviadas de *Amigos*, *Fumante* e *Cancer*.

Conhecimento	Base de Conhecimento de Primeira Ordem	Forma Normal Conjuntiva
Fumar causa câncer.	$\forall x \text{ Fu}(x) \Rightarrow \text{Ca}(x)$	$\neg \text{Fu}(x) \vee \text{Ca}(x)$
Amigos de amigos são amigos.	$\forall x, y, z \text{ Am}(x, y) \wedge \text{Am}(y, z) \Rightarrow \text{Am}(x, z)$	$\neg \text{Am}(x, y) \vee \neg \text{Am}(y, z) \vee \text{Am}(x, z)$
Se duas pessoas são amigas, ambas fumam ou nenhuma fuma.	$\forall x, y \text{ Am}(x, y) \Rightarrow \text{Fu}(x) \Leftrightarrow \text{Fu}(y)$	$\neg \text{Am}(x, y) \vee \text{Fu}(x) \vee \neg \text{Fu}(y),$ $\neg \text{Am}(x, y) \vee \neg \text{Fu}(x) \vee \text{Fu}(y)$

Por exemplo, a fórmula $\forall x \text{ Fu}(x) \Rightarrow \text{Ca}(x)$ captura o conhecimento de que se uma pessoa x fuma, ela tem câncer. Apesar disso nem sempre ser verdade na vida real, essa hipótese foi assumida no modelo. As demais fórmulas deste exemplo também modelam relações que nem sempre se confirmam.

Após estabelecer uma BCPO, pode-se realizar inferência lógica nesta, que consiste em descobrir se uma fórmula vale em uma BCPO ou não. A inferência em LPO é semi-decidível (RUSSELL; NORVIG, 1995). Em poucas palavras, ser semi-decidível significa que há algoritmos capazes de afirmar que uma fórmula F é admitida por uma BCPO, mas nenhum algoritmo consegue afirmar que F não é admitida em uma BCPO.

Como consequência, as bases são com frequência construídas usando apenas um conjunto restrito da LPO com propriedades desejáveis. Por exemplo, *cláusulas de Horn*, que são cláusulas contendo um literal positivo no máximo. Algumas linguagens lógicas, como Prolog, baseiam-se apenas em lógica restrita a cláusulas de Horn.

Na maioria dos domínios é difícil estabelecer fórmulas não triviais que sejam sempre verdadeiras. As relações lógicas contruídas em geral captam apenas uma fração do conhecimento relevante. Por isso, apesar de sua expressividade, LPO pura tem aplicabilidade limitada em problemas práticos de inteligência artificial.

Em vista disso, pesquisadores tem direcionado esforços para o desenvolvimento de extensões para LPO. No caso mais simples envolvendo lógica proposicional, a limitação desta tem sido superada pela introdução de incerteza através modelos probabilísticos

gráficos. Na Seção 2.2.6 apresenta-se uma linguagem recente, que propõe uma extensão, também probabilística, para LPO.

2.2.2 Fundamentos da Teoria de Probabilidades

O conceito de probabilidade, abordado nesta seção com base principalmente no trabalho de Neapolitan (2003), inclui muitas abordagens filosóficas diferentes. Em destaque estão aquelas que entendem a probabilidade como uma fração, como uma frequência relativa, ou como um grau de crença.

Pensando em probabilidade como uma *fração*, emprega-se o *princípio da indiferença* para os cálculos necessários, segundo o qual eventos elementares são considerados equiprováveis se não há motivos claros para esperar ou preferir a ocorrência de um ou de outro evento. Conforme esse princípio, quando se consideram n eventos elementares a probabilidade de cada um deles é a fração $1/n$. Normalmente, valores de probabilidades são determinados dessa forma em jogos de chance. Um exemplo é a probabilidade de obter um rei de copas, denotado por “rc”, em um baralho de 52 cartas: $P(\{rc\}) = 1/52$.

Contudo, há situações em que a probabilidade que se deseja determinar não pode ser computada com o auxílio do princípio da indiferença. Por exemplo, considera-se o arremesso de um dado com seis faces, viciado, devido a alguma assimetria em sua estrutura. O resultado do arremesso pode ser qualquer uma das seis faces, entretanto, por consequência da assimetria, não se pode atribuir com base no princípio da indiferença a probabilidade de $1/6$ a cada um dos eventos possíveis.

Diante de situações como esta, uma probabilidade também pode ser interpretada como uma *frequência relativa*, a qual se aplica quando experimentos podem ser repetidos inúmeras vezes. Tal abordagem determina que se um experimento pode ser refeito infinitas vezes, a probabilidade de qualquer um dos eventos é um limite. Por exemplo, se $\Omega = \{um, dois, três, quatro, cinco, seis\}$ é o conjunto dos eventos prováveis, n é o número de experimentos realizados, e n_{dois} é o número de ocorrências do evento $\{dois\}$, a probabilidade deste é dada por

$$P(\{dois\}) = \lim_{n \rightarrow \infty} \frac{n_{dois}}{n}.$$

Por fim, uma probabilidade também pode significar um *grau de crença*. Nesse caso, não é possível obter o valor dessa probabilidade a partir de uma fração, fundamentada no princípio de indiferença, tampouco pode-se estimar esse valor como uma frequência

relativa, baseando-se para isso no cálculo de um valor limite a partir inúmeras repetições de um experimento.

Uma situação exemplo em que isso aconteceria seria, em um jogo de futebol, uma tentativa de estabelecer quão provável seria uma vitória do São Paulo Futebol Clube sobre o Esporte Clube Corinthians Paulista. O princípio da indiferença nesse caso claramente não se aplica; logo, esse valor de probabilidade certamente não é oriundo de uma fração. Também não se pode repetir a partida entre os times inúmeras vezes, sob as mesmas condições, o que permite concluir que essa probabilidade também não é dada por uma frequência relativa.

Assim, a probabilidade estabelecida dessa forma denota apenas uma crença com relação às chances do São Paulo vencer, arbitrada talvez com base em alguma informação extra como qualidade técnica e física dos jogadores ou resultados dos últimos confrontos entre os dois times. O valor de probabilidade assim estabelecido é chamado *grau de crença* ou *probabilidade subjetiva* (SAVAGE, 1972).

Então, dado o evento $\{\text{Vitória do São Paulo}\}$, determinar o valor de probabilidade $P(\{\text{Vitória do São Paulo}\}) = 0.80$ pode significar, por exemplo, que a crença que um torcedor apresenta sobre a vitória do São Paulo vale 80%. Porém, para um outro torcedor, essa crença, diga-se, probabilidade, pode ser 25%. Não há o torcedor certo, ou o errado. Apenas não há um valor comum de probabilidade, acerca do qual todos os torcedores concordam.

Axiomas da Teoria de Probabilidades

Em 1933, o matemático russo Andrei Nikolaevich Kolmogorov lançou as bases axiomáticas da probabilidade e desenvolveu uma teoria que constituiu um enorme avanço na área. Os três axiomas fundamentais estabelecidos por Kolmogorov (1950) são apresentados a seguir.

Axiomas 2.1 *Seja (Ω, P) um espaço de probabilidade, com espaço amostral Ω e medida de probabilidade P . Então:*

1. $P(\Omega) = 1$.
2. $0 \leq P(E) \leq 1$ para todo $E \subseteq \Omega$.
3. Considerando certa família de eventos $E_1, E_2, \dots$, contidos em Ω , e disjuntos, então $P(E_1 \cup E_2 \cup \dots) = \sum_i P(E_i)$.

Estes três itens são chamados *axiomas da teoria de probabilidade*, e fornecem medidas de probabilidades variando de 0, para um evento impossível, a 1, para um evento cujo acontecimento é certo.

Variáveis Aleatórias e Distribuições de Probabilidades

Dado um espaço de probabilidade (Ω, P) finito, uma *variável aleatória* X é uma função sobre Ω . Em outras palavras, uma variável aleatória estabelece um valor único para cada elemento (*outcome*) em um espaço amostral. O conjunto de valores que uma variável aleatória pode assumir é chamado *espaço* de X . Uma variável é dita discreta se seu espaço é finito ou contável, e contínua se seu espaço é infinito ou incontável.

Como exemplo de variável aleatória, suponha que um evento e seja a seleção de uma pessoa em uma população Ω . A idade dessa pessoa poderia ser denotada por uma variável $X(e)$. Essa mesma população poderia ser utilizada para definir outra variável, Y , com a qual $Y(e)$ poderia denotar o peso dessa pessoa, truncado para valores inteiros. Uma variável aleatória induz em seu espaço uma *função de massa de probabilidade* $P(X = x)$, em que x é qualquer elemento no espaço de X . Costuma-se usar apenas $P(x)$, em vez de $P(X = x)$.

Para o caso em que há mais de uma variável aleatória definida sobre um espaço Ω , por exemplo X e Y , a função de massa de ambas torna-se $P(x, y)$, a qual chama-se *função de massa conjunta de probabilidade*. Este conceito estende-se também para três ou mais variáveis. Dada uma função conjunta de probabilidade, pode-se determinar a função de massa de qualquer uma de suas variáveis aleatórias por meio de uma *marginalização*.

Como exemplo, considerando uma função de massa conjunta qualquer $P(x, y)$, pode-se obter $P(x)$ por meio da expressão $P(x) = \sum_y P(x, y)$, em que $\sum_y$ significa uma soma com y percorrendo todos os valores possíveis no espaço de Y . Nesse caso, $P(x)$ é chamada *função de massa marginal* de X . Esse processo pode ser aplicado também a funções com três ou mais variáveis. Por exemplo, $P(x, y) = \sum_z P(x, y, z)$ ou $P(y) = \sum_{x,z} P(x, y, z)$.

Probabilidade Condicional e Regra de Bayes

Um dos conceitos mais importantes em teoria de probabilidades é o chamado *probabilidade condicional*, cuja definição é dada a seguir.

Definição 2.1 *Sejam E e F eventos tal que $P(F) \neq 0$. A probabilidade condicional de E*

dado F , simbolizada por $P(E|F)$, é dada por:

$$P(E|F) = \frac{P(E \cap F)}{P(F)}. \quad (2.1)$$

Intuitivamente, pode-se interpretar a probabilidade condicional como sendo uma fração, isto é, a porção de elementos em F que também estão em E . Apresentando de outra forma, seja n o número de elementos no espaço amostral, n_F o número de elementos em F , e n_{EF} o número de elementos em $E \cap F$. Então:

$$P(E|F) = \frac{P(E \cap F)}{P(F)} = \frac{n_{EF}/n}{n_F/n} = \frac{n_{EF}}{n_F}, \quad (2.2)$$

que corresponde ao percentual de ocorrências em F que também estão em E . Em outras palavras, $P(E|F)$ corresponde a probabilidade de E ocorrer dado que se sabe que F ocorreu.

Outro conceito relevante ao tratamento de probabilidades é *independência*, definido a seguir.

Definição 2.2 *Dois eventos E e F são independentes se $P(E|F) = P(E)$, se $P(F) > 0$.*

Esse conceito pode ser melhor compreendido exercendo-se o raciocínio a seguir: dois eventos E e F são independentes se qualquer informação conhecida *a priori* sobre F não promove qualquer alteração no conhecimento que se tem sobre E , de forma que a probabilidade de E dado F é apenas a probabilidade de E .

Da união dos conceitos de probabilidade condicional e de independência, tem-se a definição de independência condicional.

Definição 2.3 *Dois eventos E e F são condicionalmente independentes dado um evento G se $P(G) \neq 0$ e um dos seguintes itens vale:*

1. $P(E|F \cap G) = P(E|G)$, se $P(F \cap G) > 0$.
2. $P(E|G) = 0$ ou $P(F|G) = 0$.

Dizer que E e F são condicionalmente independentes dado um terceiro evento G , significa que a ocorrência do evento G torna os eventos E e F independentes.

Para exemplificar os conceitos de independência e independência condicional, consideram-se três eventos, representados pela notação: $E = \{Rc, Re, rc\}$, $F = \{Rc, Rp, rc\}$ e

$G = \{Rc, Re, Rp, Ro\}$ em que R significa “Rei”, r significa “rainha”, e p, o, e, c são as iniciais dos naipes de um baralho de 52 cartas. Pode-se então determinar os valores de probabilidade $P(E) = 3/52$, $P(E|F) = 2/3$, do que se conclui que E e F não são independentes. Porém, sabendo-se que $P(E|G) = 1/2$ e $P(E|F \cap G) = 1/2$, conclui-se que E e F se tornam independentes quando há condicionamento em G .

Fundamentando-se em probabilidades condicionais, a *regra de Bayes*, desenvolvida por Thomas Bayes e publicada em 1763, tem sido usada como uma forma de computar probabilidades de eventos de interesse a partir de valores conhecidos de probabilidades. Considerando dois eventos E e F , esta regra é dada pela fórmula

$$P(E|F) = \frac{P(F|E)P(E)}{P(F)}, \quad (2.3)$$

a qual vale se $P(F) \neq 0$.

Nela, $P(E|F)$ é a probabilidade posterior, isto é, a probabilidade condicional do evento E após a observação de certa evidência, no caso, o evento F , e $P(F|E)$ é a verossimilhança, que é também uma probabilidade condicional, porém, considerando o evento F como um parâmetro da regra de Bayes. Em outras palavras, ao passo que uma probabilidade condicional permite prever um evento baseando-se em um parâmetro de um modelo estatístico como a regra de Bayes, a verossimilhança proporciona a predição deste parâmetro dado um evento. O termo $P(E)$ é chamado de probabilidade anterior, que representa a crença acerca do evento E , antes que qualquer outro evento tenha sido observado. Há ainda o fator de normalização, dado por $P(F)$, que também é uma probabilidade anterior.

A regra de Bayes é frequentemente empregada para calcular a probabilidade de uma hipótese dadas evidências. A regra pode ser entendida como uma forma de especificar como uma pessoa idealmente racional se comportaria acerca de certo acontecimento mediante a observação de evidências relacionadas ao mesmo. Por exemplo, certos sintomas de determinada doença podem ser observados em um paciente. Nesta situação, a regra de Bayes pode ser empregada para calcular a probabilidade de que um diagnóstico esteja correto dadas as observações dos sintomas.

2.2.3 Redes de Markov

Uma rede de Markov (RM) é um grafo não-direcionado usado para especificar um modelo compacto de uma distribuição de probabilidades para um conjunto de variáveis $X = (X_1, \dots, X_n)$ (PEARL, 1988). Uma RM codifica relações de independência entre

as variáveis. Influência probabilística “movimenta-se” de um nó para outro ao longo das arestas da rede. Esta influência pode ser bloqueada quando os nós imediatamente vizinhos de um nó X são evidenciados, isto é, quando há condicionamento nestes nós. Diz-se então que o nó X é independente dos demais nós da rede, e ao conjunto de nós imediatamente vizinhos de X dá-se o nome de cobertor de Markov.

A Figura 2.1 ilustra uma rede com oito variáveis. Nesta figura, descreve-se o diagnóstico para tuberculose para oito pacientes representados pelas variáveis. Pacientes que estiveram em contato estão conectados por uma aresta. As arestas indicam as possibilidades para transmissão da doença. Por exemplo, o paciente E esteve em contato com os pacientes C , G e H , mas não com o paciente D .

Figura 2.1: Exemplo de rede de Markov.

Em uma RM, um clique é um subconjunto máximo de nós em que cada nó está conectado a todos os demais pertencentes a esse subconjunto, e um estado é composto pelos valores assumidos pelos nós que compõem o clique. Uma RM possui uma função potencial ϕ_k para cada clique no grafo. Essa função deve ser real e não-negativa para qualquer um dos estados do clique. A distribuição conjunta associada a uma RM é dada pelo produto desses potenciais, cuja formulação é

$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}}), \quad (2.4)$$

em que $X = x$ denota um valor assumido por cada variável X_i , $x_{\{k\}}$ é o estado do k -ésimo clique, e $Z = \sum_{x \in \chi} \prod_k \phi_k(x_{\{k\}})$ é a constante de normalização, a qual garante $\sum_{x \in \chi} P(X = x) = 1$.

Uma distribuição conjunta em uma RM é comumente representada como um modelo log-linear. Nela, cada função potencial para cada clique é substituída por uma função exponencial. Este modelo log-linear é dado por

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_j w_j f_j(x) \right), \quad (2.5)$$

em que $f_j(x)$ é uma função real de estado $x_{\{k\}}$ do j -ésimo clique, e w_j é o peso desta função para este clique.

Como exemplo de cálculo, seja a sub-rede definida por E , F , G e H , na Figura 2.1, composta por dois cliques, o primeiro formado pelos nós E , G e H , e o segundo por F e H . Se f_1 tem peso $w_1 = 1.5$, e vale $f_1(\neg E \wedge G \wedge \neg H) = 1$ neste estado; f_2 tem peso $w_2 = 2.8$, e vale $f_2(F \wedge \neg H) = 1$ neste estado. Pode-se então calcular $P(\neg E \wedge F \wedge G \wedge \neg H)$, isto é, a probabilidade conjunta de uma situação em que os pacientes E e H não estão infectados e E , F e G estão. A constante de normalização Z é a soma dos produtos $w_j f_j(x)$ para todos os estados possíveis de ambos os cliques.

2.2.4 Aprendizado de Máquina

O foco em aprendizado de máquina é aprender automaticamente a reconhecer padrões complexos e tomar decisões inteligentes, ambos a partir de dados. Na literatura de aprendizado de máquina, os métodos de aprendizado podem ser divididos em três grupos: métodos não-supervisionados, supervisionados e semi-supervisionados (CHAPELLE *et al.*, 2006).

Em *aprendizado não-supervisionado* a meta é encontrar estruturas interessantes em um conjunto de n dados observados, geralmente representado por $X = (x_1, \dots, x_n)$, em que x_i pertence a um espaço χ , para todo $i \in [n] = \{1, \dots, n\}$. Neste tipo de aprendizado, os dados não apresentam qualquer informação prévia, tais como classes ou alguma forma inicial de segmentação ou agrupamento.

Em geral, assume-se a hipótese de que os dados da amostra X são independentes e identicamente distribuídos, sendo obtidos de uma distribuição qualquer sobre um espaço χ . Tem-se argumentado que o problema de aprendizado não-supervisionado consiste em fundamentalmente estimar uma função densidade de probabilidade que provavelmente tenha gerado X (DEVROYE, 1987). No entanto, há também formas “fracas” deste tipo de aprendizado, tais como estimação de quantis, *clustering*, detecção de distorções e redução de dimensionalidade.

Por outro lado, no *aprendizado supervisionado* o objetivo é aprender uma função de mapeamento de x para y , dado um conjunto de dados de treino constituído de pares (x_i, y_i) . Neste tipo de aprendizado os y_i 's $\in \Upsilon$ são chamados rótulos das observações x_i . Novamente, faz-se necessária a hipótese de que os pares (x_i, y_i) são independentes e identicamente distribuídos, e amostrados a partir de alguma distribuição com valores em

$\chi \times \Upsilon$. Tarefas de aprendizado supervisionado são bem definidas, considerando que uma função de mapeamento pode ser avaliada por meio de seu desempenho em predições em conjuntos de dados de teste. Quando os rótulos assumem valores reais, o aprendizado supervisionado é chamado de regressão. Quando os rótulos são discretos, chama-se de classificação.

O *aprendizado semi-supervisionado* é uma abordagem intermediária entre o aprendizado não-supervisionado e o supervisionado. Somando-se aos dados não-rotulados, aos métodos semi-supervisionados provê-se também alguma informação supervisionada. Em geral, são dados alguns rótulos associados a algumas observações. Assim, um conjunto de dados observados $X = (x_i)_{i \in [n]}$ pode ser dividido em duas partes: as observações $X_l = (x_1, \dots, x_l)$ para as quais os rótulos $Y_l = (y_1, \dots, y_l)$ são dados, e as observações $X_u = (x_{l+1}, \dots, x_{l+u})$, cujos rótulos são desconhecidos.

Esta é a forma padrão de aprendizado semi-supervisionado. Contudo, o uso conjunto, em aprendizado semi-supervisionado, de quantidades reduzidas de dados rotulados estendidos por uma quantidade extremamente maior de instâncias não-rotuladas tem sido controverso. Há trabalhos na literatura que favorecem a adição de dados não rotulados (CASTELLI, 1994; CASTELLI; COVER, 1995) e, também evidências contrárias a isso (NIGAM *et al.*, 2000; COZMAN; COHEN, 2002).

Para fugir deste impasse, formas alternativas de supervisão parcial têm sido empregadas, tais como a especificação de restrições sobre pares de observações, por exemplo, estabelecendo que dois pares (x_1, y_1) e (x_2, y_2) devem (ou não devem) ter o mesmo rótulo. Fornecer supervisão parcial sob a forma de rótulos pode não ser uma tarefa fácil, mesmo em aplicações semi-supervisionadas.

Segundo Cohen *et al.* (2004), a investigação de novas metodologias para aprendizado com supervisão parcial tem sido encorajada principalmente devido a situações em que a determinação destes rótulos é um processo custoso e que requer grande esforço humano. Somando-se a isto, nem sempre se dispõe de um especialista capaz de especificar rótulos, mas sim de usuários sem conhecimento profundo do domínio tratado, os quais podem prover algum tipo inovador de supervisão.

Um número considerável de trabalhos envolvendo aprendizado semi-supervisionado baseia-se em restrições sobre pares de dados como opção de supervisão, em detrimento de rótulos. A terminologia básica para designar este tipo de restrição tem sido *must-link* e *cannot-link*. No primeiro deles estabelece-se que duas observações devem apresentar o mesmo rótulo e no outro, o contrário, duas observações não podem apresentar o mesmo

rótulo. Outros termos de valor similar também tem sido empregados, tais como restrições “positivas” ou restrições de “equivalência” (para dados de mesmo rótulo), e restrições “negativas” ou restrições de “não-equivalência” (para dados de rótulos distintos).

Como exemplos de trabalhos envolvendo restrições, ??) desenvolveram um modelo gráfico para representar restrições sobre pares de dados, as quais foram representadas como variáveis aleatórias. Neste modelo, a meta era classificar os dados segundo um grupo de rótulos pré-conhecidos. Para auxiliar o processo de classificação, alguma supervisão foi fornecida sob a forma de restrições probabilísticas.

O termo probabilístico foi usado devido à incerteza sobre o conhecimento disponível para especificar a variável aleatória de uma restrição, ou seja, não se pode afirmar com certeza se dois pares (x_1, y_1) e (x_2, y_2) têm (ou não têm) o mesmo rótulo. O modelo de ??) gerou resultados melhores quando comparado com outras versões do mesmo modelo sem, no entanto, capturar a incerteza nas restrições com probabilidades, e demonstrou-se robusto a restrições erroneamente especificadas

No trabalho de Shental *et al.* (2003), foi desenvolvida uma metodologia para incorporação de restrições de equivalência, especificadas por especialistas, no algoritmo de *clustering* EM (*Expectation Maximization*) (DEMPSTER *et al.*, 1977). O objetivo era estimar um modelo estatístico para uma amostra de dados, cujos rótulos eram completamente desconhecidos. O algoritmo EM estendido por restrições propiciou resultados superiores aos apresentados por dois outros algoritmos, também com restrições incorporadas, o *K-means* e o algoritmo de acoplamento completo, na maioria dos experimentos.

Outra contribuição do trabalho de Shental *et al.* (2003) foi a constatação de que as versões com restrições do EM e *K-means* saíram-se superiores em desempenho às versões destes sem restrições. No entanto, uma elevação de desempenho com a adição de restrições nem sempre acontece, pois, para a versão com restrições do algoritmo de acoplamento completo, Shental *et al.* (2003) relata resultados inferiores, em comparação com a implementação original deste algoritmo.

Na proposta de Basu *et al.* (2004), tem-se um modelo probabilístico semi-supervisionado que combina uma técnica de *clustering* baseada em medidas de distância entre dados, e campos ocultos aleatórios de Markov para modelagem de restrições sobre pares de dados. As restrições são fixadas por um especialista. Neste modelo, a meta é novamente classificação de dados de uma amostra em grupos pré-definidos.

Basicamente, o trabalho de Basu *et al.* (2004) desenvolve um função objetivo cuja

meta é uma minimização da distância entre dados dentro de um grupo (dados com mesmo rótulo) evitando, ao máximo, violar restrições do tipo *must* e *cannot-links*. A incorporação de restrições beneficiou o algoritmo de minimização deste trabalho de duas formas. Na primeira delas, as restrições propiciaram ao algoritmo um “chute” inicial mais acurado para a localização *a priori* de cada grupo. Na segunda forma, as restrições atuaram como guias na inserção de um dado em certo grupo.

A literatura ainda apresenta inúmeros trabalhos que tratam de aprendizado com restrições e em aplicações semi-supervisionadas. O leitor interessado pode consultar diversas publicações nessa área, tais como os trabalhos de Cohen *et al.* (2004), Bilenko *et al.* (2004), Yu e Shi (2004), Kamvar *et al.* (2003) e Wagstaff *et al.* (2001).

2.2.5 Classificadores e Métricas de Avaliação de Desempenho

Esta seção apresenta a definição tradicional de classificador, que remete unicamente ao modo supervisionado de aprendizado. Contudo, esta dissertação usa a expressão “classificador” de uma forma mais abrangente. Neste trabalho, a palavra classificador é empregada tanto para classificadores treinados de forma puramente supervisionada, como também para aqueles treinados de forma não-supervisionada (que são tradicionalmente conhecidos como métodos de *clustering*) ou semi-supervisionada. Além de conceitos sobre classificadores, esta seção define também as métricas de avaliação de desempenho de classificadores, acurácia, precisão e revocação, comumente empregadas em problemas de classificação.

Informalmente, um classificador é uma função que devolve um valor discreto de rótulo para um objeto, dados os atributos que descrevem as características deste objeto. De acordo com Devroye *et al.* (1996), define-se um *classificador* como um mapeamento da forma $g(x) : \mathbb{R}^d \rightarrow \{1, 2, \dots, M\}$, em que x é um vetor de dimensão d , representativo das observações, e $\{1, 2, \dots, M\}$ é o conjunto finito dos valores que um rótulo representado por uma variável y pode assumir. Uma observação x é erroneamente classificada quando $g(x) \neq y$. Para avaliar um classificador, uma métrica usual é a probabilidade de erro dada por

$$e_g = P(y \neq g(x)). \quad (2.6)$$

O classificador considerado ótimo é aquele que minimiza a probabilidade de uma

classificação errônea. Este classificador é dado por

$$\begin{aligned} g^* &= \arg \min_g e_g \\ &= \arg \min_g P(y \neq g(x)). \end{aligned} \quad (2.7)$$

O problema de encontrar g^* é chamado de *problema de Bayes*, e g^* é chamado *classificador de Bayes*. A probabilidade de erro obtida com esse classificador é mínima, sendo o erro chamado *erro de Bayes*. Na maioria das ocasiões, a distribuição $P(x, y)$ que associa valores de probabilidade para os pares (*observação, rótulo*) é desconhecida, consequentemente g^* também é. O erro de Bayes neste caso não pode ser determinado. No lugar do erro de Bayes calcula-se então um erro empírico, dado por

$$\hat{e}_g = \frac{1}{N} \sum_{i=1}^N I_{y \neq g(x)}(x, y), \quad (2.8)$$

em que N é o número de observações do problema, e $I_{y \neq g(x)}(x, y)$ é uma função indicadora que devolve 1, se x foi corretamente classificado, e 0 caso contrário. Quando $P(x, y)$ não está disponível, pode-se estimá-la e o classificador é construído usando esta estimativa. Outra possibilidade é construí-lo diretamente a partir de dados coletados.

Em geral, um classificador é aprendido (ou treinado) utilizando-se apenas uma parte das observações disponíveis. Esta porção de dados empregada para treinar o classificado é chamada de conjunto de *dados de treino*. A parte restante, o conjunto de *dados de teste*, é usada para testar o classificador, aplicando-o para rotular estes dados de teste e, a partir disso, estimar a probabilidade de erro.

O uso de dados de teste é essencial para a detecção de *overfitting*, situação na qual o classificador treinado apresenta um excelente desempenho quando aplicado aos dados de treino, mas falha na predição de rótulos de novos dados. Na ausência de um conjunto de dados projetado unicamente para testar o classificador, usa-se um procedimento chamado *validação cruzada com K-repetições* (*K-fold cross validation*). Este procedimento usa parte dos dados disponíveis para treinar o classificador, e uma parte diferente para testá-lo. Segmentando-se os dados em K conjuntos de tamanho similar, por exemplo, quando $K = 5$, a Figura 2.2 ilustra o cenário da validação cruzada.

1	2	3	4	5
Treino	Treino	Teste	Treino	Treino

Figura 2.2: Esquema ilustrativo de uma validação cruzada com 5-repetições.

Para a k -ésima parte (terceira parte nesta figura), treina-se o classificador com o conjunto resultante da união das outras $K - 1$ partes dos dados, e calcula-se o erro do classificador quando este é testado com a k -ésima parte. Este processo é repetido para $k = 1, 2, \dots, K$. Após isso, combina-se as K estimativas do erro de predição.

Como métricas de avaliação de resultados de classificação, as medidas de acurácia, precisão e revocação são comumente empregadas. Para definí-las adequadamente, faz-se necessário apresentar o conceito de *matriz de confusão*. Seja R o número de rótulos de classe de um domínio, a matriz de confusão M de um classificador de dados deste domínio é uma matriz $R \times R$ cujos elementos $M_{i,j}$ são determinados pelo número de vezes que um objeto originalmente pertencente ao rótulo de classe i é classificado como j . Nesta matriz, os elementos $M_{i,i}$ da diagonal principal indicam quantos objetos foram corretamente classificados, ao passo que os demais elementos indicam quantos erros foram cometidos pelo classificador.

As medidas de precisão e revocação foram inicialmente empregadas em tarefas de recuperação de informação para medir qualidade e abrangência de um conjunto de documentos retornados por uma busca (MANNING *et al.*, 2008). Em aplicações de classificação, tais medidas informam a qualidade e abrangência em relação a um rótulo de classe i , sendo determinadas pelas equações

$$precisão(i) = \frac{M_{i,i}}{\sum_{j=1}^R M_{j,i}} \quad (2.9)$$

$$revocação(i) = \frac{M_{i,i}}{\sum_{j=1}^R M_{i,j}}. \quad (2.10)$$

Para tornar estes conceitos mais intuitivos, pode-se escrevê-los como

$$precisão(i) = \frac{\text{Número de instâncias corretamente rotuladas como } i}{\text{Número de instâncias rotuladas como } i}$$

$$revocação(i) = \frac{\text{Número de instâncias corretamente rotuladas como } i}{\text{Número de instâncias que realmente têm rótulo } i}.$$

O numerador de ambas as métricas é o mesmo, e refere-se à quantidade de instâncias corretamente classificadas com um rótulo i . Em outras palavras, nada mais é do que o número de acertos. A diferença está no denominador. A precisão avalia um classificador com relação ao número de acertos dentre o total de instâncias que são classificadas com rótulo i , independentemente se este rótulo é o correto ou não. Já a revocação avalia um

classificador com relação ao número de acertos dentre o total de instâncias que realmente pertencem ao rótulo i .

A acurácia de um classificador avalia seu desempenho de forma geral, considerando todos os rótulos. Ela é dada pela equação

$$acurácia = \frac{\sum_{i=1}^R M_{i,i}}{\sum_{i=1}^R \sum_{j=1}^R M_{i,j}}, \quad (2.11)$$

cujas intuição pode ser expressada por

$$acurácia = \frac{\text{Número total de instâncias corretamente rotuladas}}{\text{Número total de instâncias}}.$$

O numerador desta equação refere-se a todas as instâncias corretamente classificadas, e denominador, obviamente, refere-se à quantidade total de instâncias. A acurácia é, simplesmente, o percentual global de acertos de um classificador, isto é, dentre todas as instâncias classificadas, que porção está corretamente rotulada.

2.2.6 Redes Lógicas de Markov

Nesta seção, apresenta-se uma linguagem de representação de conhecimento denominada *rede lógica de Markov* (RLM), ou simplesmente lógica de Markov, que combina probabilidades codificadas em modelos gráficos do tipo redes de Markov e relações de dependência representadas em LPO. As principais referências para este assunto são os trabalhos de Richardson e Domingos (2006), Singla e Domingos (2007).

Definição, Conceitos e Exemplos

Em inteligência artificial, probabilidades e lógica oferecem duas relevantes formas para abordagem de problemas reais. Por um lado, modelos probabilísticos gráficos permitem um tratamento adequado para incerteza. Por outro, a lógica, em especial a LPO, permite a representação concisa de uma grande variedade de domínios de conhecimento. Assim, a junção destas duas áreas em uma única representação tem sido objeto de estudo na literatura de inteligência artificial.

Diversos domínios de conhecimento contêm estruturas relacionais complexas, as quais podem ser compactamente modeladas com LPO. Por exemplo, representar a transitividade de uma relação binária em um domínio com n objetos requer n^3 fórmulas em lógica

proposicional, mas apenas uma em LPO: $\forall x, y, z \ R(x, y) \wedge R(y, z) \Rightarrow R(x, z)$. Exatamente como uma rede de Markov pode ser vista como uma extensão probabilística da lógica proposicional, uma RLM é uma extensão probabilística da LPO.

A idéia embutida em RLMs pode ser melhor esclarecida com a interpretação a seguir. Considerando uma BCPO como um conjunto de restrições binárias sobre um conjunto de mundos possíveis, se um mundo qualquer infringir uma fórmula sequer desta BCPO, tal mundo terá probabilidade de ocorrência igual a zero. Com o intuito de superar essa limitação, a meta no emprego de RLMs é “suavizar” essas restrições, permitindo que, ao infringir uma fórmula, um mundo se torne menos provável, mas não impossível. Assim, quanto menos fórmulas forem violadas, mais provável este mundo será.

Para acoplar o aspecto probabilístico em LPO, obtendo-se com isso uma RLM, associa-se um número real denominado *peso* a cada fórmula de uma BCPO, o qual codifica a incerteza sobre a validade desta fórmula em um mundo. Quanto maior o peso, maior probabilidade de ocorrência de um mundo que satisfaz a fórmula atrelada a este peso, em comparação com um mundo em que tal fórmula é falsa. Intuitivamente, o peso de uma fórmula é simplesmente a chance (*odds*) de um mundo onde esta é verdadeira, contra um mundo onde esta é falsa.

Dado que qualquer fórmula em LPO pode ser convertida para forma clausal (GENESERETH; NILSSON, 1987) (vide Seção 2.2.1), pode-se definir uma RLM, em poucas palavras, como um conjunto de cláusulas de primeira ordem ponderadas. A Tabela 2.2 exemplifica como especificar uma RLM. Os pesos de cada fórmula podem ser estabelecidos com auxílio de um especialista, ou podem ser aprendidos a partir de dados de treino. Na Seção 2.2.6, contempla-se como pesos podem ser aprendidos a partir de dados.

Tabela 2.2: Exemplo de especificação de uma rede lógica de Markov.

Rede Lógica de Markov	Pesos
$\forall x \text{ Fumante}(x) \Rightarrow \text{Cancer}(x)$	1.5
$\forall x, y \text{ Amigos}(x, y) \Rightarrow \text{Fumante}(x) \Leftrightarrow \text{Fumante}(y)$	1.1

À medida que os pesos aumentam, uma RLM se aproxima de uma BCPO puramente lógica. No limite de todos os pesos tendendo ao infinito e assumindo valores iguais, uma RLM representa uma distribuição uniforme sobre os mundos que satisfazem a BCPO. Alguns parágrafos adiante há um exemplo de cálculo de probabilidades que elucida a capacidade de RLMs estenderem a LPO.

Quando se fixa um conjunto de constantes possíveis para cada variável que compõe

as fórmulas de uma RLM, pode-se instanciar tais fórmulas. A partir destas instanciações, pode-se contruir uma rede de Markov, cujos nós são átomos instanciados (ou átomos, simplesmente). A Figura 2.3 ilustra a rede de Markov contruída a partir da RLM da Tabela 2.2, considerando as constantes **Ana**, denotada por **A**, e **Paulo**, denotada por **P**. Nesta figura, **Amigos(A,P)** e **Fumante(A)** são exemplos de átomos instanciados, bem como os demais nós são instanciados.

Figura 2.3: Rede de Markov construída após a instanciação da rede lógica de Markov da Tabela 2.2.

Uma RLM pode ser vista como modelo para construção de redes de Markov. Dados conjuntos diferentes de constantes, uma mesma RLM originará diferentes redes de Markov. O tamanho destas redes pode variar muito, dependendo da quantidade de constantes envolvidas no problema. No entanto, todas as redes de Markov geradas terão certa regularidade em estrutura e parâmetros, dada pela RLM geratriz. Por exemplo, todas as proposicionalizações de uma mesma fórmula terão mesmo peso. A rede de Markov gerada a partir da instanciação das fórmulas de uma RLM chama-se *rede de Markov proposicionalizada*.

Formalmente, uma rede lógica de Markov L é um conjunto de pares (F_i, w_i) , em que F_i é uma fórmula em lógica de primeira ordem e w_i é um número real chamado peso. Junto com um conjunto finito de constantes $C = \{c_1, \dots, c_{|C|}\}$, uma rede lógica de Markov define uma rede de Markov proposicionalizada $M_{L,C}$ contendo

1. Um nó binário para cada proposicionalização de cada predicado que aparece em L . O valor deste nó é 1 se o predicado proposicionalizado é verdadeiro, e 0 caso contrário;
2. Uma função característica que atribui um valor para cada proposicionalização de cada fórmula F_i em L . O valor desta função é 1 se a fórmula instanciada é verdadeira, e 0 caso contrário.

A partir da definição de RLM e da Equação (2.5), a distribuição de probabilidades sobre mundos possíveis x 's, especificada por uma rede de Markov proposicionalizada $M_{L,C}$,

é dada por

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_i^F w_i n_i(x) \right), \quad (2.12)$$

em que $Z = \sum_x \exp \left(\sum_i^F w_i n_i(x) \right)$ é chamada função de partição, F é o número de fórmulas na RLM, $n_i(x)$ é o número de instanciações verdadeiras de F_i em um mundo x e w_i é o peso associado à fórmula F_i .

A estrutura do grafo da $M_{L,C}$ é estabelecida com auxílio da definição de RLM e com base em algumas regras de construção explicadas a seguir. As proposicionalizações de cada fórmula são obtidas simplesmente pela substituição de suas variáveis por constantes em todas as combinações possíveis.

Por exemplo, na Figura 2.3, os átomos **Fumante(A)** e **Cancer(A)** são formas proposicionalizadas oriundas da fórmula $\forall x \text{ Fumante}(x) \Rightarrow \text{Cancer}(x)$, considerando a constante **A**. Similarmente, **Fumante(P)** e **Cancer(P)** originam-se desta mesma fórmula, porém, aplicando-a à constante **P**. O clique definido pelos átomos **Fumante(A)**, **Fumante(P)** e **Amigos(A,P)** provém da fórmula $\forall x, y \text{ Amigos}(x, y) \Rightarrow \text{Fumante}(x) \Leftrightarrow \text{Fumante}(y)$, proposicionalizada com **A** e **P**.

Como as variáveis devem ser substituídas por constantes em todas as combinações possíveis, um segundo clique deve ser construído, ainda com **Fumante(A)** e **Fumante(P)**, mas agora envolvendo **Amigos(P,A)**. Semanticamente, os átomos **Amigos(A,P)** e **Amigos(P,A)** são iguais, mesmo assim ambos devem ser adicionados à rede, para evitar perda de informação de condicionamento, caso, por exemplo, **Amigos(P,A)** seja observado, em vez de **Amigos(A,P)**. A ausência de um destes átomos poderia prejudicar a inferência a partir desta rede.

Deve ser posicionada uma aresta entre dois nós da $M_{L,C}$ se os predicados instanciados correspondentes a estes nós aparecerem juntos em pelo menos uma proposicionalização de uma fórmula na L . Consequentemente, os átomos em cada fórmula proposicionalizada formam um clique, não necessariamente máximo, na $M_{L,C}$. Como exemplo, na Figura 2.3, os predicados relacionados aos átomos **Fumante(A)**, **Fumante(P)** e **Amigos(A,P)** aparecem juntos na fórmula proposicionalizada $\text{Amigos}(A,P) \Rightarrow \text{Fumante}(A) \Leftrightarrow \text{Fumante}(P)$, logo deve-se colocar uma aresta para cada combinação dois a dois destes átomos. Semelhantemente, deve-se posicionar arestas para $\text{Amigos}(P,A) \Rightarrow \text{Fumante}(P) \Leftrightarrow \text{Fumante}(A)$, aproveitando os átomos que já estão na rede.

Após construir a rede de Markov da Figura 2.3, ela pode ser usada para deduzir diversas informações sobre relacionamentos de amizade e sobre hábitos de fumo. Por

exemplo, seria possível classificar Paulo como um fumante ou não-fumante, dado que ele é amigo de Ana, que fuma. Ou ainda, seria possível diagnosticar (classificar) Ana como cancerosa, dado que ela é amiga de Paulo, que fuma. Genericamente, observando-se alguns átomos desta rede, inúmeras informações podem ser inferidas probabilisticamente.

Isso confere às redes lógicas de Markov adequação a problemas de classificação automática, dado que uma RLM associada a um conjunto de constantes é uma rede de Markov ordinária, passível de inferência probabilística para classificar um objeto, e cujas variáveis aleatórias (nós) são átomos representativos de relações lógicas entre objetos. A grande vantagem das redes lógicas de Markov está na capacidade de especificar relações de dependência não apenas por meio de arestas que conectam nós em uma rede de Markov proposicionalizada, mas principalmente pela especificação de relações de dependência em LPO.

Na $M_{L,C}$, um estado é representado pelo conjunto de valores, 0 ou 1 (verdadeiro ou falso, para o átomo, denotado por um nó), que cada nó pode assumir. Cada estado da $M_{L,C}$ representa um mundo possível da rede. É interessante, portanto, acompanhar um exemplo de um problema hipotético de classificação com RLMs, que envolve cálculo de probabilidades de um mundo (estado) da $M_{L,C}$. Neste exemplo também explica-se, para um caso restrito, como uma RLM pode ser uma extensão da LPO.

Exemplo Hipotético de Classificação

Deseja-se especificar um classificador em RLM. O conhecimento disponibilizado por um especialista para auxiliar a especificação do classificador permite escrever duas fórmulas em lógica de Markov. Além disso, tal especialista também é capaz de atribuir um peso a cada fórmula. A seguir, a Tabela 2.3 apresenta o conhecimento disponível, as fórmulas que compõem a RLM e os pesos.

Tabela 2.3: Exemplo de rede lógica de Markov para um problema hipotético de classificação.

Conhecimento	Rede Lógica de Markov	Pesos
O atributo de um objeto determina o valor de seu rótulo de classe.	$\forall \mathbf{x}, \mathbf{v}, \mathbf{c} \ S(\mathbf{x}, \mathbf{v}) \Rightarrow \mathbf{R}(\mathbf{x}, \mathbf{c})$	1.9
Se dois objetos têm o mesmo atributo, então eles têm o mesmo rótulo de classe.	$\forall \mathbf{x}_1, \mathbf{x}_2, \mathbf{v}, \mathbf{c} \ S(\mathbf{x}_1, \mathbf{v}) \wedge S(\mathbf{x}_2, \mathbf{v}) \Rightarrow \mathbf{R}(\mathbf{x}_1, \mathbf{c}) \wedge \mathbf{R}(\mathbf{x}_2, \mathbf{c})$	1.3

Para construir a rede de Markov proposicionalizada deste exemplo, é preciso estabelecer quais valores as variáveis $\mathbf{x}$, $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{c}$ e $\mathbf{v}$ podem assumir. Tais valores são as constantes

do problema. Os conjuntos a seguir especificam esses valores: $\mathbf{x} = \{X1, X2\}$, $\mathbf{x}_1 = \{X1\}$ e $\mathbf{x}_2 = \{X2\}$ são os objetos que devem ser classificados, $\mathbf{c} = \{C1, C2\}$ são os rótulos de classe, e $\mathbf{v} = \{V1, V2\}$ são os valores possíveis para o atributo S. Em conjunto com estas constantes, as fórmulas em lógica de Markov originam doze fórmulas proposicionalizadas, listadas a seguir.

$$\begin{aligned}
&S(X1, V1) \Rightarrow R(X1, C1) \\
&S(X1, V2) \Rightarrow R(X1, C1) \\
&S(X1, V1) \Rightarrow R(X1, C2) \\
&S(X1, V2) \Rightarrow R(X1, C2) \\
&S(X2, V1) \Rightarrow R(X2, C1) \\
&S(X2, V2) \Rightarrow R(X2, C1) \\
&S(X2, V1) \Rightarrow R(X2, C2) \\
&S(X2, V2) \Rightarrow R(X2, C2) \\
&S(X1, V1) \wedge S(X2, V1) \Rightarrow R(X1, C1) \wedge R(X2, C1) \\
&S(X1, V2) \wedge S(X2, V2) \Rightarrow R(X1, C1) \wedge R(X2, C1) \\
&S(X1, V1) \wedge S(X2, V1) \Rightarrow R(X1, C2) \wedge R(X2, C2) \\
&S(X1, V2) \wedge S(X2, V2) \Rightarrow R(X1, C2) \wedge R(X2, C2).
\end{aligned}$$

A Figura 2.4 apresenta a rede de Markov proposicionalizada construída a partir destas fórmulas, segundo a definição de RLMs e com base nas regras de construção explicadas anteriormente.

Figura 2.4: Exemplo de rede de Markov proposicionalizada para um problema hipotético de classificação.

Recordando, os nós de uma rede de Marvok proposicionalizada são binários. A rede da Figura 2.4 tem oito nós. Isso gera 2^8 estados, isto é, 256 mundos possíveis. Obviamente, foge ao escopo deste exemplo apresentar todos os cálculos de probabilidade destes mundos. Contudo, considerando apenas uma subrede, que compreende os átomos $S(X1, V1)$ e $R(X1, C2)$ originados da fórmula $S(X1, V1) \Rightarrow R(X1, C2)$, pode-se exemplificar o cálculo de algumas probabilidades. Nesta subrede há quatro mundos possíveis, dados por

$$\begin{aligned} x_1 &= \{S(X1, V1), R(X1, C2)\} \\ x_2 &= \{S(X1, V1), \neg R(X1, C2)\} \\ x_3 &= \{\neg S(X1, V1), R(X1, C2)\} \\ x_4 &= \{\neg S(X1, V1), \neg R(X1, C2)\}. \end{aligned}$$

A formulação para a obtenção das probabilidades de cada um destes mundos nesta subrede é dada pela Equação (2.12),

$$P(X = x) = \frac{1}{Z} \exp \left(\sum_i^F w_i n_i(x) \right).$$

Neste caso, como se analisa apenas a subrede gerada por uma única fórmula em LPO, dada por $\forall \mathbf{x}, \mathbf{v}, \mathbf{c} \ S(\mathbf{x}, \mathbf{v}) \Rightarrow R(\mathbf{x}, \mathbf{c})$, tem-se $F = 1$. Logo a Equação 2.12 assume a forma

$$P(X = x) = \frac{e^{w n(x)}}{e^{w n(x_1)} + e^{w n(x_2)} + e^{w n(x_3)} + e^{w n(x_4)}}.$$

Para determinar $n(x)$, basta responder a pergunta: quantas proposicionalizações da fórmula $\forall \mathbf{x}, \mathbf{v}, \mathbf{c} \ S(\mathbf{x}, \mathbf{v}) \Rightarrow R(\mathbf{x}, \mathbf{c})$ com peso $w = 1.9$, são verdadeiras no mundo x ? Recordando que tal fórmula possui forma normal conjuntiva $\neg S(\mathbf{x}, \mathbf{v}) \vee R(\mathbf{x}, \mathbf{c})$, basta testar esta cláusula em cada um dos mundos para obter os $n(x)$'s. Por exemplo, para a subrede considerada, a cláusula $\neg S(X1, V1) \vee R(X1, C2)$ vale em $x_1 = \{S(X1, V1), R(X1, C2)\}$, $x_3 = \{\neg S(X1, V1), R(X1, C2)\}$ e $x_4 = \{\neg S(X1, V1), \neg R(X1, C2)\}$, o que permite concluir que $n(x_1) = n(x_3) = n(x_4) = 1$ para estes mundos. Porém, esta cláusula não vale para o mundo $x_2 = \{S(X1, V1), \neg R(X1, C2)\}$, o que leva a $n(x_2) = 0$. Então, a probabilidade do mundo x_2 , por exemplo, é

$$P(x_2) = \frac{1}{3e^w + 1}.$$

E para os demais mundos,

$$P(x_1) = P(x_3) = P(x_4) = \frac{e^w}{3e^w + 1}.$$

Se $w > 0$, o efeito do peso é tornar o mundo que é inconsistente com a sentença lógica $\forall \mathbf{x}, \mathbf{v}, \mathbf{c} \ S(\mathbf{x}, \mathbf{v}) \Rightarrow R(\mathbf{x}, \mathbf{c})$, isto é, o mundo x_2 (que não satisfaz a cláusula $\neg R(X1, C2) \vee S(X1, V1)$),

menos provável que os outros três.

Estes cálculos simplificados feitos para ilustrar a determinação de probabilidades de mundos podem ser realizados de maneira análoga, quando se considera toda a rede de Markov da Figura 2.4.

Finalmente, a subrede considerada (ou toda a rede, caso estejam determinadas as probabilidade de todos os seus mundos) pode ser usada para classificar um objeto. Por exemplo, pode-se prever a probabilidade do rótulo de classe do objeto X_1 ser C_2 , dado que $S(X_1, V_1)$ está observado. Classificar significa calcular a probabilidade condicional $P(R(X_1, C_2)|S(X_1, V_1))$, o que pode ser feito aplicando-se a regra de Bayes.

$$\begin{aligned}
 P(R(X_1, C_2)|S(X_1, V_1)) &= \frac{P(S(X_1, V_1)|R(X_1, C_2))P(R(X_1, C_2))}{P(S(X_1, V_1))} \\
 &= \frac{P(\{R(X_1, C_2), S(X_1, V_1)\})}{\sum_R P(\{R(X_1, C_2), S(X_1, V_1)\})} \\
 &= \frac{P(\{R(X_1, C_2), S(X_1, V_1)\})}{P(\{R(X_1, C_2), S(X_1, V_1)\}) + P(\{\neg R(X_1, C_2), S(X_1, V_1)\})} \\
 &= \frac{P(x_1)}{P(x_1) + P(x_2)} \\
 &= \frac{\frac{e^w}{3e^w+1}}{\frac{e^w}{3e^w+1} + \frac{1}{3e^w+1}} \\
 &= \frac{1}{1 + e^{-w}}.
 \end{aligned}$$

Dado que $w = 1.9$, obtém-se $P(R(X_1, C_2)|S(X_1, V_1)) = 0.87$, ou seja, a probabilidade de X_1 ter rótulo de classe C_2 é 87%, dado que X_1 tem valor de atributo V_1 . Como se trata de um exemplo de classificação binária, i.e., uma decisão entre dois rótulos de classe apenas, o classificador rotula o objeto X_1 com C_2 , pois $P(R(X_1, C_2)|S(X_1, V_1)) = 0.87 > 0.5$.

Conforme comentado no início deste exemplo, uma RLM é uma extensão da LPO. Isto pode ser demonstrado de forma simples para o caso restrito do exemplo apresentado, detalhes desta habilidade de RLMs para generalizar LPO podem ser estudados no trabalho de Richardson e Domingos (2006). O resultado da classificação no exemplo foi uma probabilidade condicional, cujo valor, $P(R(X_1, C_2)|S(X_1, V_1)) = \frac{1}{1+e^{-w}}$, é função apenas do peso da fórmula $\forall x, v, c \ S(x, v) \Rightarrow R(x, c)$. Para w alto, tem-se $P(R(X_1, C_2)|S(X_1, V_1))$ tendendo a 1, o que recupera o aspecto puramente lógico da LPO para esta fórmula. Isto confere às RLMs a possibilidade de representar não só modelos puramente probabilísticos, o que pode ser alcançado com a rede de Markov proposicionalizada, mas também modelos

puramente lógicos.

Além de classificação, muitas outras tarefas de aprendizado estatístico relacional podem ser concisamente formuladas na linguagem de RLMs. Nos parágrafos a seguir, cinco problemas relevantes são exemplificados: classificação coletiva, predição de *links*, clustering baseado em *links*, e modelagem de redes sociais e identificação de objetos. Tais problemas são propostos por Richardson e Domingos (2006).

Classificação Coletiva

A meta em problemas de classificação usuais é prever a classe de certo objeto dados seus atributos. O problema de classificação coletiva considera também as classes de objetos relacionados. Os atributos podem ser representados em RLMs como predicados da forma $A(\mathbf{x}, \mathbf{v})$, em que A é um atributo, $\mathbf{x}$ é um objeto, e $\mathbf{v}$ é o valor de A em $\mathbf{x}$. O atributo de classe pode ser representado por $C(\mathbf{x}, \mathbf{r})$, em que $\mathbf{r}$ é o rótulo de classe de $\mathbf{x}$. A tarefa de classificação é simplesmente o problema de inferir o valor lógico de $C(\mathbf{x}, \mathbf{r})$ para todos os $\mathbf{x}$ e $\mathbf{r}$ de interesse, dados os valores observados dos atributos $A(\mathbf{x}, \mathbf{v})$.

O problema de classificação usual é o caso especial em que $C(\mathbf{x}_i, \mathbf{v})$ e $C(\mathbf{x}_j, \mathbf{v})$ são independentes para todo $\mathbf{x}_i$ e $\mathbf{x}_j$, dados os $A(\mathbf{x}, \mathbf{v})$ conhecidos. Em classificação coletiva, o cobertor de Markov de $C(\mathbf{x}_i, \mathbf{v})$ inclui outros $C(\mathbf{x}_j, \mathbf{v})$, mesmo após condicionamento nos $A(\mathbf{x}, \mathbf{v})$ conhecidos. Relações entre objetos são representadas por predicados da forma $R(\mathbf{x}_i, \mathbf{x}_j)$. Uma generalização interessante é, por exemplo, a possibilidade de que $C(\mathbf{x}_i, \mathbf{v})$ e $C(\mathbf{x}_j, \mathbf{v})$ sejam indiretamente independentes através de predicados não-observados, incluindo talvez os próprios predicados de relação $R(\mathbf{x}_i, \mathbf{x}_j)$.

Predição de *Links*

O objetivo em predição de links é determinar se existe uma relação entre dois objetos de interesse, a partir das propriedades destes objetos e de outras relações conhecidas. Por exemplo, saber se um **Artigo** foi escrito por certo **Professor**. A formulação deste problema em RLMs é idêntica àquela de classificação coletiva, com a única diferença de que agora, a meta é inferir o valor de $R(\mathbf{x}_i, \mathbf{x}_j)$ para todos os pares de objeto de interesse, em vez de $C(\mathbf{x}, \mathbf{v})$.

Clustering Baseado em Links

O objetivo em problemas de *clustering* é agrupar objetos com atributos similares. Em *clustering* probabilístico, assume-se um modelo generativo $P(X) = \sum_C P(X|C)P(C)$, em que X é um objeto, C é a variável representativa de um grupo, e $P(C|X)$ é o grau de pertinência de X ao grupo C . Em *clustering* baseado em *links*, objetos são agrupados de acordo com seus relacionamentos e, possivelmente, de acordo com seus atributos. Por exemplo, objetos que possuem alguma relação inerente são mais propensos a pertencer a um mesmo grupo.

Este problema pode ser abordado com RLMs postulando-se um predicado não-observado $C(\mathbf{x}, \mathbf{r})$ com o significado “ $\mathbf{x}$ pertence ao grupo $\mathbf{r}$ ”, e tendo fórmulas na RLM envolvendo este predicado e aqueles observados (e.g., $R(\mathbf{x}_i, \mathbf{x}_j)$ para relacionamentos e $A(\mathbf{x}, \mathbf{v})$ para atributos). O agrupamento pode então ser executado aprendendo-se pesos para as fórmulas na RLM, e os graus de pertinência aos grupos para cada objeto são dados pelas probabilidades inferidas para os predicados $C(\mathbf{x}, \mathbf{r})$, condicionados naqueles observados.

Modelagem de Redes Sociais

Redes sociais são grafos em que os nós representam atores sociais (e.g., pessoas) e arcos representam relações entre estes atores (e.g., amizade). A análise de redes sociais foca a construção de modelos envolvendo atores, suas propriedades e relacionamentos. Por exemplo, a probabilidade de dois atores formarem um relacionamento pode depender da similaridade de seus atributos. Reciprocamente, dois atores conectados por algum relacionamento podem ter certas propriedades em comum.

Os modelos em análise social são em geral redes de Markov, e podem ser representados de forma compacta por fórmulas como $\forall \mathbf{x}, \mathbf{y} R(\mathbf{x}, \mathbf{y}) \Rightarrow A(\mathbf{x}) \Leftrightarrow A(\mathbf{y})$, na qual $\mathbf{x}$ e $\mathbf{y}$ são atores, $R(\mathbf{x}, \mathbf{y})$ é uma relação entre eles, $A(\mathbf{x})$ representa um atributo de $\mathbf{x}$. Um peso associado a esta fórmula captura a força da correlação entre a relação e a similaridade de atributo. Por exemplo, um modelo estabelecendo que amigos tendem a ter hábitos de fumo similares pode ser representado por $\forall \mathbf{x}, \mathbf{y} \text{ Amigos}(\mathbf{x}, \mathbf{y}) \Rightarrow \text{Fumante}(\mathbf{x}) \Leftrightarrow \text{Fumante}(\mathbf{y})$.

Identificação de Objetos

A problema de identificação de objetos se propõe a determinar quais registros em uma base de dados referem-se a mesma entidade no mundo real. Por exemplo, saber quais registros em uma base de dados bibliográficos representam a mesma publicação.

Este problema é importante para muitas companhias, agências governamentais e projetos científicos de larga escala.

Um maneira de modelá-lo com RLMs seria definir um predicado de igualdade $\text{Iguais}(\mathbf{x}, \mathbf{y})$ com o significado “ $\mathbf{x}$ representa a mesma entidade do mundo real assim como $\mathbf{y}$ ”. Este predicado seria aplicado tanto a registros de uma base de dados como também a campos (e.g., $\text{Iguais}(\text{ICML}, \text{Intl. Conf. on Mach. Learn.})$), que significa que os objetos dados por ICML e Intl. Conf. on Mach. Learn. referem-se à *International Conference on Machine Learning*.

As dependências entre registros iguais e campos iguais poderiam ser representadas por fórmulas como $\text{Iguais}(\mathbf{x}, \mathbf{y}) \Leftrightarrow \text{Iguais}(\mathbf{f}_i(\mathbf{x}), \mathbf{f}_i(\mathbf{y}))$, em que $\mathbf{x}$ e $\mathbf{y}$ são registros e $\mathbf{f}_i(\mathbf{x})$ é uma função que retorna o valor do i th campo do registro $\mathbf{x}$. Como esta modelagem permite a propagação de informação de uma igualdade de objetos (i.e., uma proposicionalização do predicado $\text{Iguais}(\mathbf{x}, \mathbf{y})$) para outra, via campos que aparecem em ambos os pares de registros, ela também realiza identificação coletiva de objetos.

Por exemplo, casar duas referências bibliográficas pode determinar que “ICML” e “MLC” representam a mesma conferência, o que por sua vez pode ajudar a casar outros pares de referências em que um deles contenha “ICML” e o outro “MLC”. RLMs também permitem que informação adicional seja incorporada de maneira simples, modular e uniforme. Por exemplo, $\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \text{ Iguais}(\mathbf{x}, \mathbf{y}) \wedge \text{Iguais}(\mathbf{y}, \mathbf{z}) \Rightarrow \text{Iguais}(\mathbf{x}, \mathbf{z})$ representa uma relação de transitividade que poderia ser adicionada.

Aprendizado de Pesos em RLMs

O aprendizado em RLMs se dá pela estimação dos pesos, o qual pode ser feito tanto de forma generativa como discriminativa. Em uma abordagem generativa, os pesos podem ser estimados a partir da maximização da função de verossimilhança, dada pela Equação (2.12), de uma ou mais BCPOs proposicionalizadas, também denominadas bases de dados relacionais. Por uma questão de simplicidade de equacionamento, a formulação a seguir é feita para uma base relacional apenas.

Assume-se também a hipótese de mundo fechado (GENESERETH; NILSSON, 1987), a qual estabelece que todos os átomos proposicionalizados que não aparecem em uma base proposicionalizada possuem valor falso. Caso a hipótese adotada fosse de mundo aberto, em que o valor dos átomos que não aparecem na base é desconhecido, i.e., faltante (*missing*), seria necessário empregar um algoritmo tal como o EM (DEMPSTER *et al.*,

1977), para estimar os valores não-observados.

Se houver n átomos proposicionalizados, a base relacional é na verdade interpretada como um vetor $x = (x_1, \dots, x_l, \dots, x_n)$, em que x_l é o valor verdade do l th átomo proposicionalizado ($x_l = 1$ se o átomo aparece na base, e $x_l = 0$ caso contrário). Dada uma base, os pesos de uma RLM podem ser aprendidos, em princípio, empregando-se métodos padrão de otimização. Se a i th fórmula da BCPO possui $n_i(x)$ proposicionalizações verdadeiras no vetor x , então pela Equação (2.12), o gradiente do logaritmo da função de verossimilhança (daqui por diante denotada por log-verossimilhança) com relação à variável peso é

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w(X = x) &= n_i(x) - \sum_{x'} P_w(X = x') n_i(x') \\ &= n_i(x) - E_w[n_i(x)], \end{aligned} \quad (2.13)$$

em que a soma é realizada sobre todos os vetores possíveis, i.e., bases possíveis x' , e $P_w(X = x')$ é a própria verossimilhança $P(X = x')$, porém computada com o emprego da estimativa atual para o vetor $w = (w_1, \dots, w_i, \dots)$ contendo os pesos para cada fórmula proposicionalizada. De uma forma mais simples, o i th componente do gradiente é simplesmente a diferença entre o número de proposicionalizações verdadeiras da i th fórmula e seu valor esperado considerando a estimativa atual de pesos. Porém, a contagem do número de proposicionalizações de uma fórmula em uma base relacional é uma tarefa computacionalmente intratável, ainda que a fórmula seja uma cláusula simples (a prova deste resultado pode ser consultada em Richardson e Domingos (2006, Proposição 6.1)).

Um segundo ponto inconveniente na Equação (2.13) é o fato de que computar o valor esperado do número de proposicionalizações verdadeiras, i.e., a parcela $E_w[n_i(x)]$, é também uma tarefa computacionalmente intratável, dado que isto requer a execução de inferência sobre o modelo. Além disto, algoritmos de otimização eficientes também defrontam-se com a barreira de ter de computar a função log-verossimilhança dada pela Equação (2.12), e consequentemente a função de partição Z .

Em vista destas dificuldades, uma alternativa mais eficiente e muito empregada em áreas como estatística espacial, modelagem de redes sociais e processamento de linguagem é a otimização de uma função pseudo-verossimilhança (BESAG, 1975),

$$P_w^*(X = x) = \prod_{l=1}^n P_w(X_l = x_l | MB_x(X_l)), \quad (2.14)$$

em que $MB_x(X_l)$ é o estado do cobertor de Markov de X_l nos dados. O gradiente da

pseudo-log-verossimilhança é

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w^*(X = x) &= \sum_{l=1}^n [n_i(x) - P_w(X_l = 0 | MB_x(X_l)) n_i(x_{[X_l=0]}) \\ &\quad - P_w(X_l = 1 | MB_x(X_l)) n_i(x_{[X_l=1]})], \end{aligned} \quad (2.15)$$

em que $n_i(x_{[X_l=0]})$ é o número de proposicionalizações verdadeiras da i th fórmula quando se fixa $X_l = 0$ e o restante dos dados permanecem inalterados, e similarmente para $n_i(x_{[X_l=1]})$. Computar a Equação (2.15) ou a Equação (2.14) não requer inferência sobre o modelo. Para otimizá-las, Richardson e Domingos (2006) usam um algoritmo do tipo *quasi*-Newton denominado L-BFGS, comumente aplicado em problemas de otimização de larga escala (LIU *et al.*, 1989).

Em muitas aplicações de inferência, são conhecidos *a priori* as variáveis que serão observadas, i.e., evidências, e as variáveis acerca das quais deve-se inferir um valor de probabilidade (*query variables*). O objetivo neste contexto é classificar corretamente estas últimas dadas as evidências. De acordo com o artigo de Singla e Domingos (2005), a formulação básica para aprendizado nesta situação, o qual é dito discriminativo, é apresentada a seguir.

Se uma base relacional for usada para treinar um classificador for particionada em um conjunto X de átomos que são evidências e um conjunto Y de átomos para serem classificados (*query atoms*), a verossimilhança condicional de Y dado X é

$$\begin{aligned} P(y|x) &= \frac{1}{Z_x} \exp \left(\sum_{i \in F_Y} w_i n_i(x, y) \right) \\ &= \frac{1}{Z_x} \exp \left(\sum_{j \in G_Y} w_j g_j(x, y) \right), \end{aligned} \quad (2.16)$$

em que F_Y é o conjunto de todas as cláusulas de uma RLM com ao menos uma proposicionalização envolvendo um átomo para ser classificado, $n_i(x, y)$ é o número de proposicionalizações verdadeiras da i th cláusula envolvendo átomos para serem classificados, G_Y é o conjunto de cláusulas proposicionalizadas na RLM envolvendo átomos para serem classificados, e $g_j(x, y) = 1$ se a j th cláusula é verdadeira na base de treino, e 0 caso contrário. Maximiza-se então o gradiente da log-verossimilhança condicional

$$\begin{aligned} \frac{\partial}{\partial w_i} \log P_w(y|x) &= n_i(x, y) - \sum_{y'} P_w(y'|x) n_i(x, y') \\ &= n_i(x, y) - E_w[n_i(x, y)]. \end{aligned} \quad (2.17)$$

Como no caso generativo, computar a parcela $E_w[n_i(x, y)]$ é intratável. Contudo, ela pode ser aproximada pela contagem $n_i(x, y_w^*)$ no valor corrente $y_w^*(x)$ do máximo *a posteriori* (MAP), o que é uma boa aproximação se a maior parte da massa de probabilidade de $P_w(y|x)$ estiver concentrada ao redor de $y_w^*(x)$. Desta forma, computar a Equação (2.17) requer apenas inferência do tipo MAP para encontrar $y_w^*(x)$, o que é extremamente mais rápido do que o cálculo total da inferência condicional para $E_w[n_i(x, y)]$. A generalização em detalhes desta formulação para RLMs arbitrárias foge ao escopo deste trabalho, mas pode ser acompanhada em Singla e Domingos (2005).

Inferência Probabilística em RLMs

A proposicionalização de uma RLM origina uma RM que pode ser usada para realização de inferências. Podem ser respondidas questões como “Qual a probabilidade da fórmula F_1 ser satisfeita dado que a fórmula F_2 é satisfeita?”. Se F_1 e F_2 são duas fórmulas em lógica de primeira ordem, C é um conjunto finito de constantes incluindo as constantes que aparecem em F_1 ou F_2 , e L é uma RLM, então

$$\begin{aligned} P(F_1|F_2, L, C) &= P(F_1|F_2, M_{L,C}) \\ &= \frac{P(F_1 \wedge F_2|M_{L,C})}{P(F_2|M_{L,C})} \\ &= \frac{\sum_{x \in \chi_{F_1} \cap \chi_{F_2}} P(X = x|M_{L,C})}{\sum_{x \in \chi_{F_2}} P(X = x|M_{L,C})}, \end{aligned} \quad (2.18)$$

em que χ_{F_i} é o conjunto dos mundos em que F_i é satisfeita, e $P(X = x|M_{L,C})$ é dada pela Equação (2.12).

Saber se uma BCPO satisfaz uma fórmula F em LPO, i.e., se F é verdadeira em todos os mundos em que BCPO é verdadeira, é o mesmo que verificar se $P(F|L_{KB}, C_{KB,F}) = 1$, em que L_{KB} é a RLM obtida associando-se pesos iguais e infinitos a todas as fórmulas na BCPO, e $C_{KB,F}$ é o conjunto de todas as constantes que aparecem na BCPO ou em F . Esta verificação pode ser feita computando-se $P(F|L_{KB}, C_{KB,F})$ pela Equação (2.18), com $F_2 = True$.

No entanto, computar a Equação (2.18) diretamente é algo intratável, a menos que o domínio considerado seja reduzido. Como estimar probabilidades em uma RLM proposicionalizada requer inferência probabilística, a qual é #P-completa, e inferência lógica, a qual é NP-completa mesmo em domínios finitos, resultados melhores são dificilmente alcançados. A princípio, $P(F_1|F_2, L, C)$ pode ser aproximada por meio de algoritmos baseados no método de Monte Carlo combinado com cadeias de Markov, que rejeitam todos

os estados em que F_2 não é satisfeita, contando-se o número de amostras em que F_1 é satisfeita. Contudo, mesmo esta abordagem pode ser muito lenta para fórmulas arbitrárias. No trabalho de Richardson e Domingos (2006) propõe-se um algoritmo de inferência para o caso em que F_1 e F_2 são conjunções de literais proposicionalizados.

Apesar desta abordagem ser menos geral do que a Equação (2.18), uma conjunção de termos é na prática a forma mais frequente para a qual uma probabilidade dever ser estimada. Somando-se a isso, o algoritmo de Richardson e Domingos estima de forma mais eficiente tal probabilidade, quando comparado à aplicação direta da Equação (2.18). De forma simplificada, o algoritmo é composto por duas etapas. Na primeira delas, retorna-se o subconjunto mínimo M de nós da RM proposicionalizada, os quais são necessários para calcular $P(F_1|F_2, L, C)$. Na segunda etapa, realiza-se inferência neste subconjunto, com os nós relacionados a F_2 fixados nos valores que aparecem na BCPO proposicionalizada para a fórmula F_2 . A implementação deste algoritmo usa a amostragem de Gibbs (GILKS *et al.*, 1996).

A etapa crucial na inferência em RLMs com o algoritmo de Gibbs consiste em amostrar um átomo proposicionalizado por vez, dado o seu cobertor de Markov. O cobertor de Markov de um átomo proposicionalizado é o conjunto de átomos proposicionalizados que aparecem em alguma instanciação de uma fórmula em que este átomo aparece. Considerando o contexto de RLMs, esta definição de cobertor de Markov tem um significado diferente daquele apresentado na Seção 2.2.3, sobre redes de Markov. A probabilidade de certo átomo X_l proposicionalizado com o valor x_l , quando seu cobertor de Markov B_l encontra-se no estado b_l é calculada por

$$P(x_l|b_l) = \frac{\exp(\sum_{f_i \in F_l} w_i f_i(X_l = x_l, B_l = b_l))}{\exp(\sum_{f_i \in F_l} w_i f_i(X_l = 0, B_l = b_l)) + \exp(\sum_{f_i \in F_l} w_i f_i(X_l = 1, B_l = b_l))}, \quad (2.19)$$

em que F_l é o conjunto de fórmulas proposicionalizadas em que X_l aparece, w_i é o peso associado a i th fórmula e $f_i(X_l = x_l, B_l = b_l)$ é o valor (0 ou 1) da função correspondente a i th fórmula instanciada quando $X_l = x_l$ e $B_l = b_l$. Em palavras, a probabilidade estimada de uma conjunção de literais proposicionalizados é, simplesmente, a fração de amostras nas quais os átomos proposicionalizados não verdadeiros.

Contudo, estimar tais probabilidades em RLMs apenas com o emprego de métodos puramente probabilísticos fundamentados em MCMC ou em propagação de crença, os quais são adequados ao tratamento de relações de dependência probabilísticas, pode levar a aproximações pobres, que acontecem devido à presença de relações de dependência (fórmulas em lógica de primeira ordem) determinísticas ou quase-determinísticas (i.e., quando um peso de uma fórmula em uma RLM tende ao infinito). Por outro lado, métodos

lógicos de inferência, tais como os testes de satisfatibilidade, são inaplicáveis em contextos probabilísticos. Com o objetivo acoplar lógica e probabilidades em um único método de inferência para a melhorar as estimativas de probabilidades em RLMs proposicionalizadas, Poon e Domingos (2006) desenvolveram o algoritmo de inferência MC-SAT, largamente empregado em RLMs, o qual combina idéias de MCMC e de satisfatibilidade.

Esta combinação o torna capaz de lidar com dependências tanto probabilísticas como determinísticas ou quase-determinísticas empregando para isso o algoritmo de Wei *et al.* (2004), denominado *SampleSAT*, como uma subrotina para “pular” eficientemente entre regiões isoladas e quase-isoladas de probabilidade não-zero. Poon e Domingos (2006) realizaram experimentos os quais mostraram que o MC-SAT apresenta desempenho superior à amostragem de Gibbs, em diversos problemas de portes e graus de determinismo diferentes.

Pacote *Alchemy*

Esta seção tem como objetivo apresentar de forma resumida algumas funcionalidades principais do pacote *Alchemy*, empregadas nesta dissertação para aprendizado e teste de classificadores especificados com RLMs. Para detalhes, consultar Kok *et al.* (2005).

O pacote *Alchemy* provê uma série de algoritmos para aprendizado estatístico relacional e inferência lógica probabilística, a partir de RLMs. Basicamente, ele permite a execução de três tipos de tarefas: aprendizado de estrutura de uma RLM, aprendizado de pesos e inferência probabilística. Neste trabalho, apenas o aprendizado de pesos e inferência são testados.

No aprendizado de pesos, a meta é aprender os parâmetros de um modelo (pesos) dada uma base de dados de treino consistindo de átomos proposicionalizados. Na inferência o objetivo é estimar a probabilidade ou estado mais provável de átomos *query*, dada uma base de treino de testes contendo átomos-evidência proposicionalizados, isto é, os átomos observados. No *Alchemy* há dois comandos básicos, um para aprendizado e outro para inferência: **learnwts**, para aprender pesos, e **infer**, para executar inferência. Tais comandos requerem argumentos de entrada. Para **learnwts**, tem-se

```
-i <string>      Arquivos de entrada, extensão .mln.  
-o <string>      Arquivo de saída, extensão .mln.  
-t <string>      Arquivos com base de dados de treino, extensão .db.
```

Para realizar inferência, o comando **infer** requer uma RLM com um peso para cada

fórmula (os pesos podem ser aprendidos ou fixados por um especialista), informações de condicionamento (evidências) e um conjunto de átomos *query* (em geral, predicados). Os argumentos básicos para o comando `infer` são

```
-i <string>           Arquivos de entrada, extensão .mln.
-r <string>           Arquivo de saída contendo os resultados de inferência, extensão .mln.
-e <string>           Arquivos com as observações, extensão .db.
-q <string>           Átomos query (separados por vírgula, sem espaço).
```

Tanto para o comando de aprendizado de pesos, como para o comando de inferência, existem outros argumentos possíveis, mas opcionais. Para uma lista completa destes argumentos, pode-se consultar a referência mencionada ao início desta seção

Um arquivo com extensão `.mln` consiste de duas partes básicas: declarações e fórmulas. A seção de declarações precisa conter ao menos um predicado, já a seção de fórmulas pode conter zero ou mais fórmulas. Enumerar as constantes (i.e., os valores que cada variável pode assumir) de cada tipo usadas no arquivo `.mln` ou `.db` é opcional. Se não houver enumeração, o conjunto de constantes é obtido de ambos os arquivos. Um arquivo `.db` consiste de um conjunto de átomos proposicionalizados, um por linha.

Em um arquivo `.mln`, pode-se especificar fórmulas arbitrárias em LPO. A sintaxe para símbolos lógicos é: `!` (negação), `^` (E), `v` (OU), `=>` (implicação), `<=>` (se e somente se), `Forall` (quantificação universal), e `Exist` (quantificação existencial). A precedência de operadores é: negação > E > OU > implicação > se e somente se > `Forall` = `Exist`. Operadores com mesma precedência são avaliados da esquerda para a direita. Parênteses podem ser usados para reforçar precedência. Quantificação no nível mais externo pode ser omitida, i.e., variáveis livres são interpretadas como universalmente quantificadas no nível mais externo. Quantificadores podem ser aplicados a mais de uma variável de uma vez (e.g., `Forall x,y`).

Como exemplo de especificação de um modelo em RLMs para aprendizado e inferência com *Alchemy*, deseja-se modelar uma distribuição multinomial dada pelo resultado do lançamento de um dado de seis faces. No arquivo de entrada `.mln` pode-se enumerar as saídas e as faces de um dado, na seção de declaração, tal como em

```
throw=(1,...,20)
face=(1,...,6)
```

e a saída de cada lançamento como o predicado

`Outcome(throw,face)`

Adicionalmente, deseja-se modelar o fato de que cada lançamento tem exatamente uma saída. Isto requer uma regra estabelecendo que no mínimo uma saída precisa ocorrer para cada lançamento, e outra estabelecendo que no máximo uma saída pode ocorrer, conforme exemplificado em

`Exist f Outcome(t,f).`
`Outcome(t,f) ^ f != f' => !Outcome(t,f').`

Estas duas regras devem ser modeladas como restrições fortes, o que é denotado pelo ponto final situado ao final de cada fórmula. Restrições fortes são fórmulas com valor lógico verdadeiro. Em outras palavras, em uma RLM, tais fórmulas são sempre verdadeiras.

Porém, caso se queira especificar um modelo em que diversas fórmulas fortes (i.e., restrições fortes) são necessárias, o código em LPO pode tornar-se muito denso, poluído e de difícil leitura. Para remover este inconveniente, o *Alchemy* permite declarar este tipo de fórmula de uma maneira mais simples e compacta. No lugar das duas sentenças lógicas anteriores, pode-se declarar o predicado `Outcome` com o operador `!` colocado após a variável `face`. Internamente ao *Alchemy*, esta notação significa que cada lançamento do dado tem apenas uma saída. A nova construção é

`Outcome(throw,face!)`

Para executar inferência probabilística neste modelo, considerando `Outcome` como predicado *query*, o comando `infer -ms -i multinomial.mln -r multinomial.result -e empty.db -q Outcome` pode ser empregado.

Neste comando, `-ms` é uma opção sem argumento que aciona um tipo especial de algoritmo de inferência chamado MC-SAT, desenvolvido por Poon e Domingos (2006), `multinomial.mln` é um arquivo de entrada contendo a RLM do exemplo com fórmulas e pesos aprendidos, `multinomial.result` é um arquivo de saída que contém as probabilidades marginais para cada proposicionalização de `Outcome`, `empty.db` é um arquivo de evidências (no caso, um arquivo vazio, i.e., sem evidências). O resultado desta inferência será uma probabilidade aproximadamente igual para cada proposicionalização de `Outcome`.

Considera-se agora um dado viciado, em que cada face não apresenta a mesma probabilidade. Em RLMs, isto requer o aprendizado de um peso diferente para cada proposici-

onalização da variável `face`. Consequentemente, seria necessário escrever seis fórmulas a partir da cláusula `Outcome(t,f)`, com `f` variando de 1 a 6. A sintaxe do pacote *Alchemy* oferece uma notação amigável e enxuta para isso, usando o operador `+`, tal como em

$$\text{Outcome}(t, +f)$$

Ao encontrar esta notação, o pacote *Alchemy* produz as cláusulas para as quais pesos podem ser aprendidos, isto é,

$$\begin{aligned} &\text{Outcome}(t, 1) \\ &\text{Outcome}(t, 2) \\ &\vdots \\ &\text{Outcome}(t, 6) \end{aligned}$$

Considerando um arquivo `biased-die.db` contendo os dados de treino gerados a partir de um dado viciado, que produz como saída apenas a face 1 ou a 6 com igual probabilidade, usa-se o operador `+` para aprender pesos para cada face através do comando `learnwts -d -withEM -i multi-biased.mln -o learned.mln -t biased-die.db -ne Outcome`.

A opção `-d(-g)` significa que o aprendizado será conduzido de forma discriminativa (generativa), `-withEM` permite executar o aprendizado empregando ao algoritmo EM (*Expectation-Maximization*) (DEMPSTER *et al.*, 1977) para estimar probabilisticamente dados faltantes, `-ne` significa que `Outcome` deve ser visto como não-evidência durante o aprendizado, em outras palavras, este é o predicado *query*, que não aparecerá no arquivo de evidências.

Como métodos de inferência, *Alchemy* suporta dois tipos básicos: Monte Carlo via Cadeias de Markov (MCMC) e MAP/MPE. A implementação atual do pacote contém três algoritmos do tipo MCMC: amostragem de Gibbs (GILKS *et al.*, 1996) (opção `-p`), MC-SAT (POON; DOMINGOS, 2006) (opção `-ms`) e *simulated tempering* (MARINARI; PARISI, 1992) (opção `-simtp`). Para inferência do tipo MCMC, as probabilidades de átomos *query* são escritas em um arquivo de saída `.result` especificado. Quando se executa inferência do tipo MAP, pode-se usar duas opções: `-m` ou `-a`. A primeira simplesmente retorna em um arquivo de saída os átomos proposicionalizados verdadeiros, na segunda, os verdadeiros e falsos são retornados.

3 Redes Lógicas de Markov Aplicadas ao Aprendizado de Classificadores

Este capítulo contém as contribuições desta dissertação. A principal delas é a aplicação de redes lógicas de Markov em aprendizado. A segunda contribuição são os testes e avaliação de alguns algoritmos do pacote *Alchemy*.

Na Seção 3.1, discute-se como se pode especificar um classificador com lógica de Markov e, à medida que esta discussão se desenvolve, algumas alternativas de especificação de classificadores são propostas para realização de experimentos.

A Seção 3.2 contém metodologia experimental, os resultados e a discussão destes. Além destas etapas, há também um exemplo de aplicação do pacote *Alchemy* em aprendizado, e um fluxograma geral com instruções para treinamento de classificadores com lógica de Markov.

3.1 Especificando Classificadores com Redes Lógicas de Markov

A modelagem de classificadores com RLMS requer cuidado, dada a flexibilidade e a forma compacta oferecidas pela LPO para modelar conhecimento de domínio. Em vista disso, as Seções 3.1.1 e 3.1.2 tratam de algumas questões de modelagem, respectivamente, nos paradigmas supervisionado e semi-supervisionado com restrições.

3.1.1 Classificadores Supervisionados

Esta seção tem duas metas principais. A primeira é desenvolver uma discussão sobre possíveis modelos de especificação de um classificador com RLMS. A segunda é defi-

nir, à medida que esta discussão se desenvolve, um conjunto de modelos para teste em aprendizado com RLMs.

Especificação de Modelos de Classificador na Ausência de Conhecimento Prévio

Em aplicações usuais de classificação, por exemplo, diagnóstico médico, reconhecimento de dígitos manuscritos e reconhecimento facial (HASTIE *et al.*, 2001), a meta é determinar o rótulo ou classe de certo objeto, a partir de um conjunto de características relevantes que, em geral, são os atributos deste objeto. Em aplicações como estas, aprender um classificador de forma supervisionada requer uma base de dados totalmente rotulada. Os rótulos necessários são, frequentemente, especificados a partir de um especialista, isto é, uma pessoa determina tais rótulos fundamentando-se no seu conhecimento do domínio de dados estudado.

Além de especificar rótulos, um especialista pode também influenciar a modelagem do problema de classificação. No caso de redes lógicas de Markov, ele pode fornecer conhecimento prévio sob a forma de relações que capturam a estrutura relacional intrínseca do domínio estudado. Entretanto, na ausência de qualquer conhecimento sobre um domínio para o qual se deseja treinar um classificador pode-se, intuitivamente, interpretar uma tarefa de classificação, por exemplo, como um processo de diagnóstico, em que sintomas são atributos de um objeto, e o diagnóstico é a classe inferida com base nestes sintomas.

Com base neste raciocínio, e partindo-se da hipótese de que se deseja construir um classificador para uma aplicação em que a priori não se conhece adequadamente o domínio, uma alternativa inicial de modelagem é a especificação de uma fórmula que denote um processo de diagnóstico. Em redes lógicas de Markov, essa fórmula pode ser escrita considerando dois predicados. O primeiro, representado por $\text{Classe}(\mathbf{x}, \mathbf{r})$, relaciona certa classe $\mathbf{r}$ a um objeto $\mathbf{x}$. Se este predicado tiver valor lógico verdadeiro quando instanciado com constantes quaisquer, por exemplo $\mathbf{X1}$ e $\mathbf{R1}$, pode-se afirmar que a classe de $\mathbf{X1}$ é $\mathbf{R1}$. O segundo, dado por $\text{Atributo}_i(\mathbf{x}, \mathbf{v}_i)$, relaciona o objeto $\mathbf{x}$ ao valor $\mathbf{v}_i$ assumido pelo i -ésimo atributo deste objeto. À sentença em RLM que traduz este raciocínio dá-se o nome *Modelo 1*.

$$\text{Modelo 1} - \forall \mathbf{x}, \mathbf{r} \text{ Atributo}_1(\mathbf{x}, \mathbf{v}_1) \wedge \dots \wedge \text{Atributo}_n(\mathbf{x}, \mathbf{v}_n) \Rightarrow \text{Classe}(\mathbf{x}, \mathbf{r}),$$

no qual tem-se o predicado de classe implicando em uma conjunção de predicados de atributos relevantes para o diagnóstico. Esta forma de especificar um classificador remete

ao processo de diagnóstico médico, no qual uma enfermidade, representada pelo atributo de classe, determina um conjunto de sintomas, denotados pelos atributos. Segundo Russell e Norvig (1995), *regras de diagnóstico* permitem descobrir causas ocultas a partir de efeitos observados. Este conceito é reproduzido nesta formulação, em que a classe é a causa oculta, e os atributos são os efeitos observados. Acreditando-se que sentenças lógicas como esta podem capturar alguma relação de dependência entre uma classe e seus atributos, propõe-se testá-la, em aprendizado, através de alguns experimentos.

Em uma segunda abordagem, cogita-se a possibilidade de uma classe ser o fator chave para definir um conjunto de valores para os atributos de um objeto. A causa oculta não é mais a classe, como discutido para o *Modelo 1*, mas sim os seus atributos. Russell e Norvig (1995) afirmam que contextos como esse permitem a construção de *regras de causa*, nas quais um grupo de propriedades ocultas causa a percepção de uma classe. O modelo que se propõe investigar através de experimentos, chamado de *Modelo 2*, implementa este tipo de regra, sendo representado por

$$\textit{Modelo 2} - \forall \mathbf{x}, \mathbf{r} \text{ Classe}(\mathbf{x}, +\mathbf{r}) \Rightarrow \text{Atributo}_1(\mathbf{x}, +\mathbf{v}_1) \wedge \dots \wedge \text{Atributo}_n(\mathbf{x}, +\mathbf{v}_n),$$

na qual tem-se uma conjunção de predicados de atributos implicando certo predicado de classe. Formulações como esta são úteis para capturar o conhecimento de que uma classe de certo objeto justifica certa configuração (conjunto de valores) de seus atributos.

Uma terceira alternativa considera uma formulação, chamada de *Modelo 3*, definida por um conjunto de fórmulas concisas, cada uma relacionando o predicado de classe a um único atributo. Novamente, esta alternativa baseia-se em regra de diagnóstico. Com esse modelo, pode-se investigar o desempenho de um classificador especificado não só considerando o predicado de classe como uma causa oculta, mas também envolvendo fórmulas que implicitamente tornam os atributos independentes entre si (já que estes não mais estão conectados através de uma relação de conjunção, como acontece no *Modelo 1* e *Modelo 2*). Este modelo investiga a qualidade de predição que cada atributo tem, quando considerado individualmente em cada fórmula. As sentenças representativas do *Modelo 3* são

$$\textit{Modelo 3} \left\{ \begin{array}{l} \forall \mathbf{x}, \mathbf{r} \text{ Atributo}_1(\mathbf{x}, +\mathbf{v}_1) \Rightarrow \text{Classe}(\mathbf{x}, +\mathbf{r}) \\ \forall \mathbf{x}, \mathbf{r} \text{ Atributo}_2(\mathbf{x}, +\mathbf{v}_2) \Rightarrow \text{Classe}(\mathbf{x}, +\mathbf{r}) \\ \vdots \\ \forall \mathbf{x}, \mathbf{r} \text{ Atributo}_n(\mathbf{x}, +\mathbf{v}_n) \Rightarrow \text{Classe}(\mathbf{x}, +\mathbf{r}), \end{array} \right.$$

Intuitivamente, um modelo em que cada fórmula lógica tem apenas dois predicados

envolvidos parece, com relação à quantidade de pesos que devem ser aprendidos, uma tarefa computacionalmente menos complexa, comparando-se com modelos com um única fórmula envolvendo uma grande conjunção de predicados, tal como no *Modelo 1* e *Modelo 2*.

A quarta e última abordagem é similar ao *Modelo 3*, entretanto, considera-se agora regras de causa em que cada predicado de classe implica o predicado de atributo. Esta formulação estuda a influência de uma classe sobre certo atributo. Quanto maior o peso aprendido para cada uma destas fórmulas, mais influente na predição é a classe. O conjunto de sentenças que especifica o *Modelo 4* é dado por

$$\text{Modelo 4} \left\{ \begin{array}{l} \forall \mathbf{x}, \mathbf{r} \text{ Classe}(\mathbf{x}, +\mathbf{r}) \Rightarrow \text{Atributo}_1(\mathbf{x}, +\mathbf{v}_1) \\ \forall \mathbf{x}, \mathbf{r} \text{ Classe}(\mathbf{x}, +\mathbf{r}) \Rightarrow \text{Atributo}_2(\mathbf{x}, +\mathbf{v}_2) \\ \vdots \\ \forall \mathbf{x}, \mathbf{r} \text{ Classe}(\mathbf{x}, +\mathbf{r}) \Rightarrow \text{Atributo}_n(\mathbf{x}, +\mathbf{v}_n). \end{array} \right.$$

Esta seção discutiu formas alternativas para especificação de um classificador com RLMS, principalmente em contextos de aprendizado supervisionado em que não há disponibilidade de informação prévia sobre a estrutura relacional do domínio, isto é, quando não se sabe exatamente que tipo de sentença lógica deve-se especificar. Foram propostos quatro modelos de especificação, cujos testes em aprendizado supervisionado constam na Seção 3.2.3.

Especificação de Modelos de Classificador na Presença de Conhecimento Prévio

Até aqui os modelos propostos foram especificados com base apenas em regras gerais de diagnóstico ou causa. Apenas conhecimento causal, representado pelas regras de causa adotadas na modelagem, foi embutido. Nenhuma das fórmulas lógicas dos modelos refletia exatamente o comportamento ou relacionamento intrínseco dos dados. Em vista disso, convém testar as redes lógicas de Markov também como linguagem para especificação de modelos de aprendizado em domínios naturalmente relacionais.

Por exemplo, em um domínio de relacionamentos familiares, informações como “Elizabeth é a mãe de Carlos” e “Carlos é o pai de Guilherme”, e regras como “A avó de certa pessoa é a mãe do pai desta pessoa” podem ser expressadas em LPO e, conseqüentemente, especificadas em um modelo de aprendizado em RLM. A partir de uma base de dados de

treino, tal modelo pode ser usado para treinar um classificador, e aplicá-lo em problemas de descobrimento de parentesco. Para elucidar como este problema pode ser abordado, Russell e Norvig (1995) propõem algumas sentenças em LPO, reproduzidas a seguir, as quais encerram conhecimento sobre um domínio de pessoas, por exemplo.

“O marido de uma pessoa é o esposo masculino dessa pessoa”

$$\forall x, y \text{ Husband}(x, y) \Leftrightarrow \text{Male}(x) \wedge \text{Spouse}(x, y)$$

“A mãe de uma pessoa é um ancestral feminino desta pessoa”

$$\forall x, y \text{ MotherOf}(x) = y \Leftrightarrow \text{Female}(y) \wedge \text{Parent}(y, x)$$

“Uma relação de irmandade é simétrica”

$$\forall x, y \text{ Siblings}(x, y) \Leftrightarrow \text{Siblings}(y, x)$$

Problemas como este permitem a especificação de sentenças lógicas consistentes com o domínio, o que é vantajoso, pois além de tornar o modelo mais próximo da realidade e mais elucidativo com relação à estrutura relacional dos dados, pode levar a classificadores mais apurados, e com treinamento mais eficiente (i.e., computação de pesos para as fórmulas lógicas mais rápida).

Em vista disso, na Seção 3.2.3 são desenvolvidos alguns experimentos para um caso particular de aprendizado a partir da base de dados relacional *Tic-Tac-Toe*¹, para a qual um modelo de aprendizado de classificador supervisionado é especificado com incorporação de conhecimento. São testados classificadores com diversos níveis de conhecimento embutido em suas formulações, partindo-se desde uma modelagem inconsistente até implementações corretas representando exatamente as relações existentes na base de dados utilizada.

3.1.2 Classificadores Semi-supervisionados com Restrições

Esta seção faz uma investigação preliminar sobre a aplicação de RLMs no treinamento de classificadores semi-supervisionados com restrições determinísticas. Algumas questões de modelagem são discutidas e um modelo preliminar para a realização de alguns experimentos é proposto.

A representação de conhecimento de domínio com RLMs é delicada, dada a flexibilidade e a sintaxe compacta oferecidas por esta linguagem. Como um exemplo, suponha-se que se deseja aprender um classificador semi-supervisionado cuja codificação em RLM

¹Esta base contém o conjunto completo de 958 configurações possíveis para o final de um jogo chamado “Jogo da Velha”.

seja dada por

$$\forall \mathbf{x}, \mathbf{v}, \mathbf{r} \text{ Atributo}_i(\mathbf{x}, \mathbf{v}) \vee \text{Atributo}_j(\mathbf{x}, \mathbf{v}) \Rightarrow \text{ClasseOculta}(\mathbf{x}, \mathbf{r}).$$

Tal fórmula significa que se um destes dois atributos de um mesmo objeto $\mathbf{x}$ possui o valor $\mathbf{v}$, então tal objeto pertence à classe $\mathbf{r}$. Porém, nada impede que a verdadeira relação entre estes predicados de atributo e o predicado de classe, seja

$$\forall \mathbf{x}, \mathbf{v}, \mathbf{r} \text{ ClasseOculta}(\mathbf{x}, \mathbf{r}) \Rightarrow \text{Atributo}_i(\mathbf{x}, \mathbf{v}) \wedge \text{Atributo}_j(\mathbf{x}, \mathbf{v}).$$

Sintaticamente, a diferença entre as sentenças é notável. Uma disjunção é substituída por uma conjunção, e o atributos passam a ser uma implicação da classe. Em outras palavras, a classe passa a ser a causa dos valores que um conjunto de atributos pode assumir. Semanticamente, tal alteração pode influenciar consideravelmente a inferência de um rótulo de classe.

Em vista disso, para investigar a adequação de redes lógicas de Markov em aprendizado semi-supervisionado, diversas formulações foram testadas em alguns experimentos incipientes. Porém, apenas uma delas gerou resultados interessantes. Esta formulação, desenvolvida a partir do *Modelo 1* testado no aprendizado supervisionado, é apresentada a seguir.

Como em problemas de aprendizado semi-supervisionado boa parte dos exemplos de treino não estão rotulados, convém empregar uma formulação em RLM que privilegie o diagnóstico de classe com base nos atributos, idéia que é contemplada com o *Modelo 1* do contexto supervisionado. Porém, deve-se adaptá-lo para o contexto atual de aprendizado. Dado que a maioria dos exemplos não está rotulada neste tipo de aprendizado, substitui-se o predicado $\text{Classe}(\mathbf{x}, \mathbf{r})$ por $\text{ClaOc}(\mathbf{x}, \mathbf{r})$, que denota a *Classe Oculta* que deve ser inferida. Parte da formulação nova que atende ao problema semi-supervisionado é então

$$\forall \mathbf{x}, \mathbf{r} \text{ A}_1(\mathbf{x}, +\mathbf{v}_1) \wedge \dots \wedge \text{A}_n(\mathbf{x}, +\mathbf{v}_n) \Rightarrow \text{ClaOc}(\mathbf{x}, +\mathbf{r}),$$

no qual $\text{A}_i(\mathbf{x}, +\mathbf{v}_i)$, para simplificar a notação, representa o i -ésimo atributo. Além desta adaptação, novas fórmulas devem ser inseridas para modelar relações envolvendo restrições *must* e *cannot-links*, apresentadas na Seção 2.2.4. Em *must-links*, estabelece-se que dois objetos devem pertencer a uma mesma classe, já em *cannot-links*, o contrário, dois objetos não podem pertencer a mesma. A grande vantagem em oferecer alguma supervisão sob a forma deste tipo de restrição é que conhecer exatamente o rótulo de cada objeto usado

no treinamento pode se tornar menos importante. Logo, isto permite que uma pessoa com conhecimento de domínio reduzido fixe, apenas, pares de objetos os quais se sabe que (não) pertencem a uma mesma classe.

No contexto supervisionado, parece simples adicionar dados rotulados a um problema de classificação comum codificado com redes lógicas de Markov, bastando para isso aprender um classificador com dados de treino que possuam observações de rótulos de classe de objetos. Já inserir restrições do tipo *must* e *cannot-links* sugere maior complexidade.

Esta dissertação propõe uma representação dos *must* e *cannot-links* sobre rótulos de classe com a criação de dois predicados adicionais. Para tornar tais predicados mais intuitivos, um *must-link* terá daqui por diante o significado “classe igual” e, similarmente, um *cannot-link* significará “classe diferente”. Esta tradução não traz nenhum prejuízo ao sentido de *must* e *cannot-links*, sendo apenas uma forma de facilitar o entendimento de uma fórmula em lógica envolvendo tais restrições. Denotados por $\text{ClaIg}(\mathbf{x}, \mathbf{y})$ e $\text{ClaDif}(\mathbf{x}, \mathbf{y})$. O primeiro significa que dois objetos $\mathbf{x}$ e $\mathbf{y}$ pertencem a uma mesma classe, isto é, têm *Classes Iguais*. Já o segundo, significa o contrário, i.e., dois objetos $\mathbf{x}$ e $\mathbf{y}$ não pertencem a uma mesma classe, isto é, têm *Classes Diferentes*.

Ao estender o *Modelo 1*, algumas sentenças que correlacionam predicados de restrição, classe e atributos são adicionadas. Estas sentenças são

$$\begin{aligned} \forall \mathbf{x}, \mathbf{y}, \mathbf{v}_1, \dots, \mathbf{v}_n \text{ ClaIg}(\mathbf{x}, \mathbf{y}) &\Rightarrow \mathbf{A}_1(\mathbf{x}, +\mathbf{v}_1) \wedge \dots \wedge \mathbf{A}_n(\mathbf{x}, +\mathbf{v}_n) \wedge \mathbf{A}_1(\mathbf{y}, +\mathbf{v}_1) \wedge \dots \wedge \mathbf{A}_n(\mathbf{y}, +\mathbf{v}_n) \\ \forall \mathbf{x}, \mathbf{y}, \mathbf{v}_1, \dots, \mathbf{v}_n \text{ ClaDif}(\mathbf{x}, \mathbf{y}) &\Rightarrow \mathbf{A}_1(\mathbf{x}, +\mathbf{v}_1) \wedge \dots \wedge \mathbf{A}_n(\mathbf{x}, +\mathbf{v}_n) \wedge \neg \mathbf{A}_1(\mathbf{y}, +\mathbf{v}_1) \wedge \dots \wedge \neg \mathbf{A}_n(\mathbf{y}, +\mathbf{v}_n) \\ \forall \mathbf{x}, \mathbf{y}, l \text{ ClaIg}(\mathbf{x}, \mathbf{y}) &\Rightarrow (\text{ClaOc}(\mathbf{x}, +l) \Leftrightarrow \text{ClaOc}(\mathbf{y}, +l)) \\ \forall \mathbf{x}, \mathbf{y}, l \text{ ClaDif}(\mathbf{x}, \mathbf{y}) &\Rightarrow (\text{ClaOc}(\mathbf{x}, +l) \Leftrightarrow \neg \text{ClaOc}(\mathbf{y}, +l)). \end{aligned}$$

A primeira fórmula pressupõe que se dois objetos pertencem a uma mesma classe, então seus atributos devem possuir valores iguais. Já na segunda fórmula, objetos de classes distintas têm atributos com valores diferentes. Naturalmente, tais fórmulas não são universalmente sempre verdadeiras, mas capturam relações que provavelmente ocorrem em uma base de dados. Já as duas últimas fórmulas assumem relações que podem ser verdadeiras para a porção rotulada de dados. A terceira fórmula representa o fato de que se dois objetos distintos têm classes iguais, o rótulo de ambos tem o mesmo valor. A quarta fórmula representa situações em que objetos de classes diferentes têm rótulos diferentes.

Como exemplo de aplicação da terceira e quarta fórmulas, se em uma base de dados existissem os átomos $\text{ClaIg}(\text{Cachorro}, \text{Gato})$ e $\text{ClaOc}(\text{Cachorro}, \text{Mamifero})$, seria possí-

vel também classificar gatos como integrantes do grupo dos mamíferos na ausência do predicado proposicionalizado $\text{ClaOc}(\text{Gato}, \text{Mamifero})$.

Por fim, para propagar o efeito das restrições para todas as proposicionalizações, enriquecendo o modelo, propõe-se também um relação de transitividade, modelada por uma fórmula lógica, tal como

$$\forall x, y, z \text{ ClaIg}(x, y) \wedge \text{ClaIg}(y, z) \Rightarrow \text{ClaIg}(x, z).$$

A interpretação para a mesma é simples, e pode ser esclarecida com base no exemplo anterior. Se $\text{ClaIg}(\text{Cachorro}, \text{Gato})$ e $\text{ClaIg}(\text{Gato}, \text{Rato})$ são observados, torna-se implícita a restrição $\text{ClaIg}(\text{Cachorro}, \text{Rato})$. Convém alertar para o fato de que *cannot-links* não admitem, necessariamente, esta relação de transitividade, assim, uma sentença impondo essa relação não deve ser adicionada.

Assim, a formulação em RLM para um problema semi-supervisionado, proposta para teste é

$$\text{Modelo SSLC} \left\{ \begin{array}{l} \forall x, r \text{ A}_1(x, +v_1) \wedge \dots \wedge \text{A}_n(x, +v_n) \Rightarrow \text{ClaOc}(x, +r) \\ \forall x, y, v_1, \dots, v_n \text{ ClaIg}(x, y) \Rightarrow \text{A}_1(x, +v_1) \wedge \dots \wedge \text{A}_n(x, +v_n) \wedge \text{A}_1(y, +v_1) \wedge \dots \wedge \text{A}_n(y, +v_n) \\ \forall x, y, v_1, \dots, v_n \text{ ClaDif}(x, y) \Rightarrow \text{A}_1(x, +v_1) \wedge \dots \wedge \text{A}_n(x, +v_n) \wedge \neg \text{A}_1(y, +v_1) \wedge \dots \wedge \neg \text{A}_n(y, +v_n) \\ \forall x, y, l \text{ ClaIg}(x, y) \Rightarrow (\text{ClaOc}(x, +l) \Leftrightarrow \text{ClaOc}(y, +l)) \\ \forall x, y, l \text{ ClaDif}(x, y) \Rightarrow (\text{ClaOc}(x, +l) \Leftrightarrow \neg \text{ClaOc}(y, +l)) \\ \forall x, y, z (\text{ClaIg}(x, y) \wedge \text{ClaIg}(y, z)) \Rightarrow \text{ClaIg}(x, z). \end{array} \right.$$

Esta seção abordou aspectos relativos a modelagem de classificadores semi-supervisionados com RLMs. Como fruto do que foi discutido, propôs-se um modelo em RLM para testes de classificação neste paradigma de aprendizado.

3.2 Metodologia, Experimentos e Discussões de Resultados

Esta seção apresenta a metodologia para o desenvolvimento de testes de redes lógicas de Markov em classificação supervisionada e semi-supervisionada com restrições. Inicialmente, apresenta-se na Seção 3.2.1 um exemplo de aprendizado e classificação com redes lógicas de Markov e a ferramenta *Alchemy*. Em seguida, propõe-se na Seção 3.2.2 um esquema geral para construção de classificadores com esta linguagem, através de um fluxograma. Nas duas partes finais, Seções 3.2.3 e 3.2.4, constam os experimentos realizados

e a discussão dos resultados dos mesmos.

Para testar o desempenho de classificadores aprendidos RLMs, foram realizados alguns experimentos com algumas bases de dados do repositório UCI², livremente disponível na internet, o qual oferece vasta coleção de bases de dados reais para tarefas gerais em aprendizado de máquina. As bases escolhidas para os testes foram : *Adult*, *Breast Cancer*, *Iris* e *Wine*, as quais estão entre as mais populares na literatura, e também *Car Evaluation*, *Ionosphere*, *Nursery* e *Tic-Tac-Toe*.

Na base *Adult* tem-se um problema de classificação binária, no qual deve-se determinar se uma pessoa recebe anualmente mais de R\$50.000,00 (ou menos), a partir de um conjunto de atributos tais como idade, classe de trabalho (e.g., privado, público), formação (e.g., graduação, mestrado), etnia ou sexo. A base possui 30162 instâncias de treino, sendo 22655 observações para renda anual inferior a R\$50.000,00, e 7509 para rendas superiores a esta.

Na base *Breast Cancer*, o problema também é de classificação binária, em que deve-se determinar se um tumor é maligno ou benigno a partir de um conjunto de atributos pré-computados com informações sobre uma célula doente, tais como raio, textura, simetria, concavidade, perímetro. Esta base possui 569 instâncias, sendo 357 casos benignos, e 212 malignos.

A base *Iris* é tradicionalmente empregada em diversas tarefas de classificação, sendo talvez a mais conhecida nessa área. A meta é classificar, a partir de quatro atributos reais, o gênero de plantas do tipo *Iris*, segundo três rótulos possíveis: *Iris Setosa*, *Iris Versicolour* e *Iris Virginica*. Esta base possui 150 instâncias, sendo 50 para cada rótulo.

A base *Wine* é um conjunto de resultados de análises químicas de vinhos de uma mesma região da Itália, mas originados de três produtores diferentes. A meta é determinar o tipo de álcool (3 classes possíveis: tipos 1, 2 e 3), a partir de um conjunto de atributos que envolve, por exemplo, variáveis como alcalinidade, teor de magnésio, intensidade de cor, quantidade de fenóis. A base possui 178 instâncias, sendo 59 observações para o tipo 1, 71 para o 2, e 48 para o 3.

Na base *Car Evaluation* o objetivo é avaliar o estado de um carro segundo quatro classes: inaceitável, aceitável, bom e muito bom. Levando-se em consideração atributos como preço de compra, custo de manutenção, número de portas, capacidade de pessoas, e tamanho do porta malas. Esta base foi derivada de um modelo hierárquico de decisão,

²Disponível em <http://archive.ics.uci.edu/ml/>

com outros 3 conceitos intermediários ainda. No entanto, nos dados de treino disponíveis no repositório, a informação estrutural está removida. Há 1728 instâncias, sendo 1210 para carros inaceitáveis, 384 aceitáveis, 69 bons, e 65 muito bons.

Na base *Ionosphere*, a meta é classificação binária de sinais de retorno de radares posicionados na ionosfera, a partir de atributos reais de medição destes sinais. Há duas classes: “bom”, para sinais de retorno que mostram alguma evidência de algum tipo de estrutura na ionosfera, e “ruim”, para aqueles sem qualquer evidência. Há 351 instâncias.

A base *Nursery* tem-se novamente uma base originada de um modelo de decisão hierárquico desenvolvido para ranquear candidatos a escolas de enfermagem. No repositório, a informação estrutural sobre outros quatro conceitos intermediários está removida. Há 12960 instâncias, e a meta é classificar um candidato como não-recomendável (4320 observações), recomendável (2 observações), muito-recomendável (328 observações), prioritário (4266 observações) e especialmente prioritário (4044 observações).

A base *Tic-Tac-Toe* apresenta o conjunto completo de 958 configurações possíveis para o final de um jogo conhecido como “Jogo da Velha”, no qual supõe-se que o jogador *x* fez o primeiro movimento. O objetivo é predizer qual jogador venceu a partida, sabendo-se que uma vitória acontece alcançando-se três marcações consecutivas formando uma fila, em qualquer direção. Há nove atributos de posição, cada um representando uma posição do jogo (e.g., quadrado-superior-direito, quadrado-superior-central, quadrado-meio-esquerda, etc) e a classe com duas categorias possíveis: *x* vence ou não. Cada atributo de posição tem três valores possíveis: *x*, que significa que o jogador *x* jogou, *o* quando o jogador *o* jogou e *b* para uma posição vazia).

Em alguns experimentos introdutórios com as bases *Adult* e *Nursery*, especialmente nos testes de classificação supervisionada, o tempo de processamento para aprendizado de pesos foi excessivamente longo (e.g., mais de 3 dias de execução em alguns casos) para algumas abordagens de modelagem. Portanto, optou-se por reduzir a quantidade de dados de treino destas duas bases. Assim pôde-se obter resultados de forma mais rápida. Essa redução do número de instâncias em cada base manteve as proporções de cada classe nos dados de treino. Somando-se a isso, instâncias com valores faltantes (*missing values*) foram removidas em todas as bases.

Cada base também sofreu uma redução na quantidade de atributos originalmente informada no repositório. O algoritmo de seleção de atributos empregado está disponível no *WEKA* e chama-se, simplificadamente, *CfsSubsetEval*. Este algoritmo avalia subconjuntos de atributos através da capacidade individual de predição de cada atributo, priorizando

atributos com baixa intercorrelação por conjunto. Para detalhes sobre o algoritmo, consultar Hall (1998). As únicas bases que não sofreram seleção de atributos foram a *Car Evaluation* e *Nursery*. Isso foi feito como uma tentativa de preservar a estrutura relacional hierárquica embutida nestas bases.

Outro fato relevante é a necessidade de discretização de atributos que assumam valores contínuos, antes de empregá-los no aprendizado com RLMs, as quais têm sua aplicação restrita a dados categóricos ou discretos. Esta discretização foi realizada para as bases *Adult* (uma parte dos atributos apenas), *Iris* (todos os atributos), *Wine* (uma parte) e *Ionosphere* (uma parte). Foi utilizado um algoritmo de discretização de Fayyad e Irani (1993), baseado no método MDL (*Minimum Description Length*), disponível no software *WEKA*.

A Tabela 3.1 sumariza as bases de dados usadas, suas características originais e também algumas alterações feitas (e.g., discretização, seleção de atributos). Na Figura 3.1 observam-se as quantidades de instâncias de treino e de teste empregadas nos experimentos.

Tabela 3.1: Bases de dados utilizadas nos testes de classificação.

Informação	<i>Adult</i>	<i>B. Cancer</i>	<i>Iris</i>	<i>Wine</i>	<i>Car Eval.</i>	<i>Ionosphere</i>	<i>Nursery</i>	<i>Tic-Tac-Toe</i>
Atributos	Categórico, Inteiro	Real	Real	Inteiro, Real	Categórico	Inteiro, Real	Categórico	Categórico
Instâncias originais	30162	569	150	178	1728	351	12960	958
Instâncias reduzidas	1500	—	—	—	—	—	3087	—
Atributos originais	14	32	4	13	6	34	8	9
Atributos selecion.	5	8	2	11	—	12	—	—
Atributos discret.	3	8	2	11	—	12	—	—
Domínio	Social	Saúde	Biologia	Químico	—	Físico	Social	Jogos

Como a maioria das bases escolhidas para os testes de classificação disponibiliza apenas dados de treino (poucas ofereciam um conjunto dedicado para teste), optou-se por avaliar todos os classificadores treinados em todos os experimentos com RLMs por meio da validação cruzada com 10-repetições. As medidas de desempenho adotadas foram a acurácia, precisão e revocação.

Nos testes de aprendizado supervisionado, as abordagens generativa e discriminativa do pacote *Alchemy* foram testadas. Quanto aos métodos de inferência, empregou-se apenas o MC-SAT e o MCMC. Já no treinamento semi-supervisionado com restrições, apenas a abordagem generativa foi testada. E para realização de inferência, foi usado o método MAP.

Figura 3.1: Quantidade de instâncias de treino e teste para cada base de dados utilizada nos experimentos.

3.2.1 Exemplo de Aprendizado de um Classificador com *Alchemy*

Esta seção tem por objetivo exemplificar como um classificador poderia ser especificado com RLMs no pacote *Alchemy*. Para construir um exemplo, emprega-se a base de dados *Iris*. Como modelo de especificação, adota-se aquele baseado em uma única regra de diagnóstico, na qual uma conjunção dos atributos preditores implica o atributo de classe. Qualquer outro dos três modelos propostos na Seção 3.1 poderia ser empregado neste exemplo, pois a meta nesta seção é apenas informar como se prepara um classificador com RLMs e o pacote *Alchemy*.

De acordo com a Tabela 3.1, a base *Iris*, após o pré-processamento (seleção e discretização), contém 2 atributos preditores: comprimento de pétala e largura de pétala. Há também o atributo de classe, isto é, o tipo de planta *Iris*, que pode assumir três valores (*Setosa*, *Virginica*, *Versicolour*). Uma forma de especificar um modelo de aprendizado de um classificador supervisionado com *Alchemy* seria criar um arquivo `iris.mln` contendo

```

/* 0 Achemy permite comentários com este formato,
que pode ser usado para múltiplas linhas de texto. */
// Ou este, para uma única linha apenas.
/***** Declaração de variáveis *****/
instance = {X1, X2, X3, X4, X5}
label = {Setosa, Virginica, Versicolour}
petalLength = {PL1, PL2, PL3}
petalWidth = {PW1, PW2, PW3}
/***** Declaração de predicados *****/
IrisClass(instance,label!)
PetalLength(instance,petalLength!)
PetalWidth(instance,petalWidth!)
/***** Fórmulas do modelo *****/
/* É permitido escrever as fórmulas com nomes
de variáveis diferentes daqueles declarados.*/
PetalLength(x,+pl) ^ PetalWidth(x,+pw) => IrisClass(x,+l)

```

As variáveis e suas contantes não precisam necessariamente ser declaradas, pois o *Alchemy* é capaz de deduzí-las a partir da base de treino utilizada. Comumente declara-se apenas as variáveis que assumem poucos valores, já que é cansativo e propenso a erros escrever longos conjuntos de constantes. Além disso, reusar o modelo extendendo-o para novos domínios implicaria atualizar a lista de constantes de cada variável. Por exemplo, uma variável como `instance` não costuma aparecer em declarações, já que ela é quem percorre todas as instâncias de treino da base, as quais podem ser milhares, ou até milhões.

Treinar este modelo significa aprender pesos para suas fórmulas lógicas a partir de uma base de dados de treino. No caso, como há apenas uma fórmula e foi empregado o operador `+`, serão aprendidos pesos específicos para cada uma das fórmulas (parcialmente) proposicionalizadas resultantes da combinação entre a variável `x` e cada constante possível assumida pelas variáveis `p1`, `pw` e `l`, sempre mantendo a variável `x` inalterada. Caso não houvesse esse operador, apenas um único peso global seria aprendido para a fórmula do modelo.

O operador `+` é conveniente quando para se deseja capturar a chance de cada fórmula proposicionalizada, gerada a partir das combinações de `x` com as constantes assumidas por `p1`, `pw` e `l`, ser verdadeira nos mundos gerados pela fórmula que define o modelo. Usar este operador torna o treinamento computacionalmente mais custoso, e requer cuidado, já que este implica calcular um peso para cada fórmula proposicionalizada. Algumas destas fórmulas geradas pelo emprego deste operador são exemplificadas a seguir


```

PetalLength(x,PL1) ^ PetalWidth(x,PW1) => IrisClass(x,Setosa)
PetalLength(x,PL1) ^ PetalWidth(x,PW1) => IrisClass(x,Virginica)
PetalLength(x,PL1) ^ PetalWidth(x,PW1) => IrisClass(x,Versicolour)
PetalLength(x,PL1) ^ PetalWidth(x,PW2) => IrisClass(x,Setosa)
PetalLength(x,PL1) ^ PetalWidth(x,PW2) => IrisClass(x,Virginica)
PetalLength(x,PL1) ^ PetalWidth(x,PW2) => IrisClass(x,Versicolour)
      ⋮
PetalLength(x,PL3) ^ PetalWidth(x,PW3) => IrisClass(x,Versicolour)

```

Este treinamento requer uma base de dados rotulados, e.g., `iris-train.db`, que deve ser fornecida como argumento de entrada no comando de treinamento

```

learnwts -g -i iris.mln -o iris-classifier.mln -t iris-train.db -ne
IrisClass

```

Um exemplo de base rotulada, em que os rótulos são os predicados de classe proposicionalizados, seria

```

PetalLength(X1,PL2)
PetalWidth(X1,PW1)
IrisClass(X1,Setosa)
PetalLength(X2,PL1)
PetalWidth(X2,PW1)
IrisClass(X2,Versicolour)
PetalLength(X3,PL1)
PetalWidth(X3,PW2)
IrisClass(X3,Virginica)
PetalLength(X4,PL2)
PetalWidth(X4,PW2)
IrisClass(X4,Setosa)

```

O resultado do treinamento com esta base é um arquivo de saída chamado, por exemplo, `iris-classifier.mln`, que nada mais é do que o classificador automático de planta do tipo *Iris* aprendido. Como explicado na introdução desta dissertação, um classificador em RLMs é um conjunto de fórmulas em LPO ponderadas. Como o conteúdo do arquivo `iris-classifier.mln` é longo, apenas uma parte dele é listada a seguir

```

0.814429 PetalLength(x,PL2) ^ PetalWidth(x,PW1) => IrisClass(x,Setosa)
2.21642 PetalLength(x,PL2) ^ PetalWidth(x,PW1) => IrisClass(x,Virginica)
-3.81809 PetalLength(x,PL2) ^ PetalWidth(x,PW1) => IrisClass(x,Versicolour)
2.90651 PetalLength(x,PL2) ^ PetalWidth(x,PW2) => IrisClass(x,Setosa)
-0.85706 PetalLength(x,PL2) ^ PetalWidth(x,PW2) => IrisClass(x,Virginica)
      ⋮
0.977288 PetalLength(x,PL1) ^ PetalWidth(x,PW2) => IrisClass(x,Versicolour)

```

Com este classificador, pode-se realizar inferências para determinar o tipo de *Iris* para uma instância não rotulada. Supondo uma base de dados de evidência, chamada `iris-evidence.db`, com alguns predicados de atributos observados para as instâncias X5, X6 e X7, e sem a observação de predicados de classe, tal como

```

PetalLength(X5,PL2)
PetalWidth(X5,PW2)
PetalLength(X6,PL1)
PetalWidth(X6,PW2)
PetalLength(X7,PL2)
PetalWidth(X7,PW1)

```

Com base nestas evidências, constróem-se redes de Markov proposicionalizadas para classificar as instâncias. O comando

```

infer -ms -i iris-classifier.mln -r iris-classified.result -e
iris-evidence.db -q IrisClass,

```

permite inferir a classe de *Iris* a partir da rede e das evidências, posicionando o predicado `IrisClass` como *query* (`-q`), e empregando o algoritmo MC-SAT (`-ms`). O resultado é exportado para um arquivo `iris-classified.result`, que contém as probabilidades marginais de cada instância, como exemplificado a seguir

```

IrisClass(X5,Setosa) 0.936956
IrisClass(X5,Virginica) 0.0220478
IrisClass(X5,Versicolour) 0.0410459
IrisClass(X6,Setosa) 0.227027
IrisClass(X6,Virginica) 0.557994
IrisClass(X6,Versicolour) 0.215028
IrisClass(X7,Setosa) 0.192031
IrisClass(X7,Virginica) 0.807969
IrisClass(X7,Versicolour) 4.9995e-05

```

Mediante estas probabilidades, pode-se concluir que **X5** pertence à classe **Setosa**, já **X6** e **X7** foram classificadas como **Virginica**. Para medir o desempenho deste classificador, pode-se obter valores de precisão, revocação e acurácia, confrontando-se estas classes inferidas com classes reais conhecidas previamente.

Uma vantagem das RLMs é a facilidade de reuso e extensão de modelos de aprendizado de classificadores. A partir do exemplo de classificador supervisionado apresentado, pode-se estendê-lo para os paradigmas não-supervisionado e semi-supervisionado com restrições, bastando para isso alterá-lo adequadamente. Para obter um modelo não-supervisionado, basta substituir o predicado de classe **IrisClass** por um predicado de classe oculta denominado, por exemplo, **IrisHiddenClass**, mantendo as mesmas variáveis. Outra modificação necessária deve ser feita na base de treino, a qual não deve conter observações do predicado de classe. Feito, isso, as demais etapas de treinamento e inferência são análogas, bastando apenas usar nomes de arquivos coerentes.

Caso se queira estender o modelo para o contexto semi-supervisionado com restrições, além de substituir **IrisClass** por **IrisHiddenClass**, torna-se necessário declarar dois predicados adicionais: **SameHiddenClass(x1,x2)** para instâncias que pertençam a uma mesma classe oculta e **DiffHiddenClass(x1,x2)** para instâncias que não pertençam a uma mesma classe, sendo que **x1** e **x2** percorrem também as instâncias **X1**, ..., **X4** da base de treino. Com estes predicados, três fórmulas devem ser incluídas no modelo. São elas

```

SameHiddenClass(x1,x2) => IrisHiddenClass(x1,+1) <=> IrisHiddenClass(x2,+1)
DiffHiddenClass(x1,x2) => IrisHiddenClass(x1,+1) <=> !IrisHiddenClass(x2,+1)
SameHiddenClass(x1,x2) ^ SameHiddenClass(x2,x3) => SameHiddenClass(x1,x3)

```

A base de treino deste classificador passa a conter não só observações de predicados de atributos preditores (e.g., **PetalLength** e **PetalWidth**), mas também observações de predicados de restrição, e poucas observações de rótulos de classe. Os predicados pro-

posicionalizados a seguir exemplificam uma base de treino semi-supervisionada, chamada `iris-semi-train.db`, derivada da base de treino supervisionada apresentada anteriormente.

```
PetalLength(X1,PL2)
PetalWidth(X1,PW1)
PetalLength(X2,PL1)
PetalWidth(X2,PW1)
IrisClass(X2,Versicolour)
PetalLength(X3,PL1)
PetalWidth(X3,PW2)
PetalLength(X4,PL2)
PetalWidth(X4,PW2)
SameHiddenClass(X1,X4)
DiffHiddenClass(X3,X4)
```

O comando para aprendizado de pesos é análogo àquele do contexto supervisionado porém, como não se conhece previamente as classes das instâncias durante o treinamento, o *Alchemy* oferece uma funcionalidade interessante para o caso não-supervisionado. Pode-se informar ao algoritmo de aprendizado que as proposicionalizações do predicado de classe oculta `IrisHiddenClass`, as quais não constam na base de treino, são “abertas”, isto é, seu valor verdade não está fixado. Caso isso não seja explicitado ao algoritmo, este considera tais proposicionalizações do predicados de classe oculta (e todos os demais predicados também) como “fechadas”, ou seja, se não estão presentes na base de treino, seu valor verdade é falso. Isso pode ser informado ao algoritmo com a opção `-ow IrisHiddenClass`, na qual `-ow` denota “mundo aberto” para o predicado informado. O comando para aprender o classificador teria uma estrutura como

```
learnwts -g -withEM -i iris-semi.mln -o iris-semi-classifier.mln -t
iris-semi-train.db -ow IrisHiddenClass
```

Analogamente, de posse de uma base de evidências, este modelo semi-supervisionado pode ser usado para inferir probabilidades para uma classe oculta de plantas do tipo Iris, empregando-se um comando como

```
infer -ms -i iris-semi-classifier.mln -r iris-semi-classified.result -e
iris-semi-evidence.db -q IrisHiddenClass
```

Concluído este exemplo sobre como se pode especificar um classificador com auxílio do pacote *Alchemy*, apresenta-se na próxima seção um esquema com etapas para construção de um classificador com RLM.

3.2.2 Esquema Geral Proposto para Aprendizado de Classificadores com RLMs

Esta seção expõe um fluxograma, representado pela Figura 3.2 e adaptado do trabalho de Kotsiantis (2007), para o processo de obtenção de um classificador empregando RLMs. Nesta figura, as etapas de construção de um modelo de aprendizado em RLM, aprendizado de pesos para este modelo e ajuste de fórmulas em LPO foram incluídas para direcionar o aprendizado de classificadores com RLMs. As demais etapas são aplicáveis em qualquer outra técnica de classificação supervisionada.

A primeira etapa do processo é a identificação dos dados representativos do problema. Se um especialista estiver disponível, ele pode sugerir as informações relevantes que devem ser levantadas, do contrário, uma saída custosa tanto financeiramente como computacionalmente é a “força-bruta”, segundo Kotsiantis (2007), a qual significa medir tudo o que for possível acerca do domínio estudado na esperança de que atributos e características importantes sejam capturados. No entanto, uma base de dados obtida por métodos de “força-bruta” não é adequada para treinamento de classificadores, já que esta pode conter ruídos e valores faltantes.

A segunda etapa envolve o pré-processamento dos dados coletados. Nesta etapa ruídos e valores faltantes são adequadamente tratados. Além disso, algoritmos de seleção de atributos ou mesmo de construção de novos atributos a partir dos originais³ devem ser aplicados para redução da dimensionalidade do problema. Selecionar atributos não é algo obrigatório, porém, é conveniente, já que estes em número reduzido, além de simplificar a implementação do modelo em RLMs, podem diminuir consideravelmente o custo computacional na estimação de pesos, tornando o algoritmo de aprendizado empregado mais rápido e eficiente.

A etapa de definição de um conjunto de treino tem por objetivo estabelecer que parte dos dados será usada para aprender o classificador. Neste momento, políticas como validação cruzada podem ser adotadas. Esta etapa é extremamente importante, e deve ser realizada com cuidado, pois o classificador deve ser treinado com dados que encerrem

³Caso alguns dos atributos originais sejam interdependentes, uma simples seleção poderia descartar informações de relacionamentos preciosas.

Figura 3.2: Fluxograma de um processo de aprendizado de um classificador com RLMs.

a maior gama possível de fenômenos do domínio estudado. Por exemplo, aleatoriedades, estruturas relacionais se for o caso e número mínimo de observações para certa classe.

A etapa de construção de um modelo de aprendizado com RLMs envolve duas atividades principais. A primeira consiste em definir predicados em LPO, representativos do domínio estudado, e também variáveis que serão argumentos destes predicados. Deve-se construir predicados para cada variável de atributo selecionado, para a variável de classe e para outros relacionamentos que se queira modelar também. Por exemplo, em um universo acadêmico, predicados como $\text{DisciplinaPosGrad}(d, p)$, o qual significa que a disciplina d de um curso de pós-graduação é ministrada por um professor p ; ou $\text{Aluno}(p)$, que significa que p é um aluno; ou ainda $\text{Orientador}(p, a)$, que especifica um professor p como orientador de um aluno a . Nesta etapa de definição de predicados, pode-se iniciar o uso do pacote *Alchemy* (KOK *et al.*, 2005) para implementar um modelo de aprendizado.

A segunda é a atividade núcleo na especificação de um modelo: a construção de fórmulas em LPO que modelem, de acordo com o conhecimento disponível, o domínio tratado. Nesta etapa as sugestões de um especialista ou mesmo a coleta de informações com pessoas relativamente familiarizadas com o problema tratado é extremamente importante. Por exemplo, frases como “Se duas amostras de petróleo têm valores de próximos

de viscosidade, então elas podem ser classificadas como um mesmo grau API⁴, podem ser convertidas em sentenças em LPO que modelam certo relacionamento. Contudo, se não houver qualquer conhecimento prévio, postular regras de diagnóstico ou de causa em LPO pode ser uma abordagem razoável.

Na etapa de aprendizado, deve-se estimar pesos para as fórmulas especificadas no modelo. Deve-se avaliar que tipo de aprendizado, generativo ou discriminativo, é relevante ao problema. Pode-se empregar um algoritmo do pacote *Alchemy* para estimar os pesos de cada fórmula.

A etapa final consiste em avaliar o classificador. As medidas de desempenho comumente empregadas são acurácia, precisão e revocação. Em geral, usa-se um conjunto de dados de teste especialmente projetado para verificar o desempenho de classificação. No entanto, na ausência deste conjunto, pode-se separar uma parte dos dados de treino apenas para testar o classificador. Se o desempenho for aceitável, o classificador pode ser validado, do contrário, inicia-se uma nova iteração com ponto de partida sendo uma das etapas anteriores, de acordo com o refinamento que se queira realizar (e.g., ajuste de fórmulas, construção de novo modelo, definição de um novo conjunto de treino, etc).

Na seção a seguir, são apresentados os principais resultados desta dissertação, obtidos a partir da aplicação de redes lógicas de Markov em problemas de aprendizado supervisionado.

3.2.3 Resultados e Discussão de Testes de RLMs em Aprendizado de Classificadores Supervisionados

Esta seção apresenta os resultados principais deste trabalho, que são os testes de redes lógicas de Markov em aprendizado supervisionado e a discussão destes. A seguir, há quatro subseções. Nas duas primeiras encontram-se, respectivamente, os resultados e a discussão do desempenho de RLMs quando classificadores são treinados na ausência de conhecimento de domínio. E nas duas últimas, os resultados e discussão de RLMs aplicadas ao aprendizado na presença de conhecimento.

⁴O grau API de um petróleo é uma medida que o classifica quanto a sua viscosidade em relação a água, em outras palavras, o grau API determina quais petróleos podem ser classificados como leves, médios, pesados ou ultrapesados quando comparados com a água.

Resultados de Classificadores Especificados sem Conhecimento Prévio

Para avaliar redes lógicas de Markov no aprendizado de classificadores supervisionados, foram testados quatro modelos distintos, anteriormente propostos na Seção 3.1.1. Nos experimentos com cada um destes modelos, foram empregadas três bases de dados: *Adult*, *Breast Cancer* e *Iris*. A formulação base de cada um destes modelos está descrita na Seção 3.1; contudo, os itens a seguir relembram sucintamente cada uma das abordagens propostas para teste nesta seção, considerando apenas dois atributos, para tornar a notação mais simples. Para cada base de dados, foram aprendidos quatro classificadores, um com cada modelo proposto. O treinamento e teste destes classificadores seguiu o esquema geral apresentado na Seção 3.2.2.

- *Modelo 1* – Regra de diagnóstico. Avalia-se a influência de uma conjunção de atributos na inferência de uma classe.

$$\forall \mathbf{x}, v_1, v_2, r \text{ Atributo}_1(\mathbf{x}, +v_1) \wedge \text{Atributo}_2(\mathbf{x}, +v_2) \Rightarrow \text{Classe}(\mathbf{x}, +r)$$

- *Modelo 2* – Regra de causa. Avalia-se a influência de uma classe sobre os atributos considerados conjuntamente.

$$\forall \mathbf{x}, v_1, v_2, r \text{ Classe}(\mathbf{x}, +r) \Rightarrow \text{Atributo}_1(\mathbf{x}, +v_1) \wedge \text{Atributo}_2(\mathbf{x}, +v_2)$$

- *Modelo 3* – Regra de diagnóstico. Avalia-se a qualidade de predição de cada atributo.

$$\forall \mathbf{x}, v_1, r \text{ Atributo}_1(\mathbf{x}, +v_1) \Rightarrow \text{Classe}(\mathbf{x}, +r)$$

$$\forall \mathbf{x}, v_2, r \text{ Atributo}_2(\mathbf{x}, +v_2) \Rightarrow \text{Classe}(\mathbf{x}, +r)$$

- *Modelo 4* – Regra de causa. Avalia-se a influência da classe sobre cada atributo.

$$\forall \mathbf{x}, v_1, r \text{ Classe}(\mathbf{x}, +r) \Rightarrow \text{Atributo}_1(\mathbf{x}, +v_1)$$

$$\forall \mathbf{x}, v_2, r \text{ Classe}(\mathbf{x}, +r) \Rightarrow \text{Atributo}_2(\mathbf{x}, +v_2)$$

As Figuras 3.3 a 3.14 apresentam alguns resultados obtidos, segundo cada um dos quatro modelos e cada uma das três bases, para classificadores chamados SMLC (*Supervised Markov Logic Classifier*). Na Figura 3.3, por exemplo, analisa-se o desempenho do classificador SMLC aprendido para a base *Adult* considerando o *Modelo 1*. Nesta figura, além de métricas como acurácia, precisão e revocação, apresenta-se também uma medida chamada *probabilidade de acerto*, que é um valor médio das probabilidades conferidas aos objetos corretamente classificados pelo método de inferência testado.

Esta medida auxiliar tem por objetivo capturar o grau de certeza com que um método de inferência prediz uma classe corretamente. No eixo das ordenadas estão os valores

Figura 3.3: Desempenho do SMLC especificado segundo o *Modelo 1* de aprendizado para a base *Adult*.

encontrados para as métricas, e nas abscissas observam-se pares tipo de aprendizado (generativo ou discriminativo)/método de inferência empregado (MCMC ou MC-SAT).

Figura 3.4: Desempenho do SMLC especificado segundo o *Modelo 2* de aprendizado para a base *Adult*.

As Figuras 3.3 a 3.14 focam informações de desempenho dos classificadores aprendidos segundo diversos modelos e bases de dados. Contudo, é interessante comparar estes classificadores, em termos de desempenho, com outros métodos de classificação supervisionada disponíveis na literatura. Esta comparação pode ser observada nas Figuras 3.15 a 3.20. Nelas, pode-se examinar simultaneamente o desempenho do SMLC e também de outros métodos comumente utilizados em aplicações de classificação. Como os resultados anteriores para os quatro modelos são bastante similares, escolheu-se adotar para comparação o SLMC baseado no *Modelo 3*, com aprendizado generativo e método de inferência MC-SAT.

Ao total foram escolhidos oito tipos de classificadores, que diferem entre si basicamente pelo arcabouço teórico empregado. Três destes métodos, cujos resultados estão ilustrados nas Figuras 3.15 e 3.16, classificam objetos com base em regras de associação obtidas a partir dos dados. Outros três são centrados em árvores de decisão, e seus resultados são apresentados nas Figuras 3.17 e 3.18. Os dois últimos fundamentam-se principalmente em

Figura 3.5: Desempenho do SMLC especificado segundo o *Modelo 3* de aprendizado para a base *Adult*.

Figura 3.6: Desempenho do SMLC especificado segundo o *Modelo 4* de aprendizado para a base *Adult*.

Figura 3.7: Desempenho do SMLC especificado segundo o *Modelo 1* de aprendizado para a base *Breast Cancer*.

Figura 3.8: Desempenho do SMLC especificado segundo o *Modelo 2* de aprendizado para a base *Breast Cancer*.

Figura 3.9: Desempenho do SMLC especificado segundo o *Modelo 3* de aprendizado para a base *Breast Cancer*.

Figura 3.10: Desempenho do SMLC especificado segundo o *Modelo 4* de aprendizado para a base *Breast Cancer*.

Figura 3.11: Desempenho do SMLC especificado segundo o *Modelo 1* de aprendizado para a base *Iris*.

Figura 3.12: Desempenho do SMLC especificado segundo o *Modelo 2* de aprendizado para a base *Iris*.

Figura 3.13: Desempenho do SMLC especificado segundo o *Modelo 3* de aprendizado para a base *Iris*.

Figura 3.14: Desempenho do SMLC especificado segundo o *Modelo 4* de aprendizado para a base *Iris*.

redes Bayesianas. As Figuras 3.19 e 3.20 revelam o desempenho destes. Atenção especial é dada aos métodos bayesianos.

Figura 3.15: Comparação (parte 1) de desempenho do SMLC (*Modelo 3*) com métodos centrados em regras de associação.

O *Naive Bayes* é um classificador puramente probabilístico, baseado em um tipo de grafo direcionado chamado rede Bayesiana (JENSEN, 2001). Este classificador recebe o nome “ingênuo” por assumir uma forte hipótese de independência condicional entre atributos preditores, dado o atributo de classe. Esta hipótese é dita “forte” porque em problemas reais, não se pode afirmar que os atributos de predição em um conjunto de dados não agregam qualquer informação uns aos outros, caso o rótulo de classe seja observado. Apesar desta hipótese ingênua porém forte, o Naive Bayes demonstra excelente desempenho, em termos de acurácia e eficiência computacional, em diversas tarefas de classificação (COHEN *et al.*, 2004; IDE *et al.*, 2006).

O *Tree-Augmented Naive Bayes* (MADDEN, 2009), em geral conhecido com TAN, refina a ingenuidade do método Naive Bayes pelo introdução de relações de dependência expressas por meio de arestas entre pares de atributos preditores, característica inexistente no Naive Bayes, cuja estrutura estabelece apenas arestas entre atributos e classe. Tais relações impõem uma estrutura de árvore ao TAN, tornando-o ainda mais eficiente e

Figura 3.16: Comparação (parte 2) de desempenho do SMLC (*Modelo 3*) com métodos centrados em regras de associação.

robusto que o próprio Naive Bayes, em diversos domínios (FRIEDMAN *et al.*, 1997).

Figura 3.17: Comparação (parte 1) de desempenho do SMLC (*Modelo 3*) com métodos centrados em árvores de decisão.

Figura 3.18: Comparação (parte 2) de desempenho do SMLC (*Modelo 3*) com métodos centrados em árvores de decisão.

Esta seção apresentou uma série de resultados de experimentos em que as RLMs foram testadas como linguagem para especificação de classificadores supervisionados cujo formulação está desprovida de conhecimento de domínio. Foram testados quatro modelos distintas, duas delas (*Modelo 1* e *Modelo 3*) com base em regras de diagnóstico, e outras duas (*Modelo 2* e *Modelo 4*) com base em regras de causa.

Figura 3.19: Comparação (parte 1) de desempenho do SMLC (*Modelo 3*) com métodos centrados principalmente em redes Bayesianas.

Figura 3.20: Comparação (parte 2) de desempenho do SMLC (*Modelo 3*) com métodos centrados principalmente em redes Bayesianas.

Para comparar o desempenho de RLMs, aplicadas em classificação supervisionada, com outras metodologias de classificação, outros seis gráficos com valores de acurácia foram plotados, considerando-se três técnicas de classificação usualmente empregadas em problemas de aprendizado: regras, árvores e redes bayesianas. Nestes experimentos, foram usadas combinações de dois tipos de aprendizado (generativo e discriminativo) e dois métodos de inferência (MCMC e MC-SAT) do pacote *Alchemy*.

Discussão dos Resultados em Aprendizado sem Conhecimento Prévio

Discute-se agora os resultados obtidos nos experimentos de classificação supervisionada e a relevância de RLMs e do pacote *Alchemy* nesta vertente de classificação.

Para resumir os resultados do treinamento supervisionado, duas figuras são apresentadas nesta seção. Na primeira, Figura 3.21, o foco é realçar o desempenho do SMLC com relação às bases de dados empregadas nos experimentos. Nesta figura, a probabilidade de acerto e as demais métricas são valores médios dos quatro modelos de aprendizado testados (*Modelo 1*, *Modelo 2*, *Modelo 3* e *Modelo 4*).

Já na Figura 3.22, o objetivo é destacar o desempenho do SMLC com relação aos

modelos de aprendizado testados. Nesta figura mostra-se, desta vez, os valores médios de probabilidade de acerto e das demais métricas encontrados para as três bases de dados empregadas (*Adult*, *Breast Cancer* e *Iris*).

Em uma visão geral, os valores apresentados nestas figuras indicam um ótimo desempenho para os classificadores aprendidos a partir das três bases de dados e também dos quatro modelos em RLMs testados, independentemente do tipo de aprendizado/método de inferência utilizado (GEN/MCMC, DISC/MCT-SAT, GEN/MC-SAT, etc). Isso indica que os algoritmos do pacote *Alchemy*, no cenário supervisionado, foram adequados para os problemas estudados.

Dentre todos os valores de acurácia, precisão e revocação ilustrados na Figura 3.21, a base *Breast Cancer* apresentou o SMLC com os melhores resultados. Todas as medidas de desempenho foram iguais ou superiores a 93%. Em segundo lugar está o SMLC para a base *Iris*, com medidas variando de 91% (precisão) a 94% (acurácia). Por fim, o SMLC para a base *Adult*, cujas medidas de desempenho variaram de 76% (precisão) a 85% (probabilidade de acerto e acurácia).

Figura 3.21: Desempenho global do classificador SMLC aprendido para as bases *Adult*, *Breast Cancer* e *Iris*.

Com relação aos modelos testados, todos proporcionaram classificadores com desempenho elevado, conforme valores indicados na Figura 3.22. Logo, pode-se concluir que tanto regras de diagnóstico como regras de causa foram capazes de proporcionar classificadores apurados. Porém, mesmo gerando bons resultados de classificação, convém analisar algumas questões delicadas com relação ao tipo de sentença lógica empregada em cada modelo.

A fórmula do *Modelo 1* é uma conjunção de predicados de atributos que implica o predicado de classe. Tal fórmula, ao ser processada pelo *Alchemy*, é convertida para forma

clausal, assumindo a estrutura a seguir

$$\textit{Modelo 1: } \neg\text{Atributo}_1(\mathbf{x}, +\mathbf{v}_1) \vee \dots \vee \neg\text{Atributo}_n(\mathbf{x}, +\mathbf{v}_n) \vee \text{Classe}(\mathbf{x}, +\mathbf{r}),$$

que é uma disjunção de predicados, com os predicados de atributos negados.

Este tipo de construção pode exigir elevado custo computacional dependendo do número de atributos da base de dados, representado por n . Em alguns experimentos iniciais, à medida que n crescia, o aprendizado de classificadores com o *Modelo 1* mostrou-se computacionalmente intratável, dado que este requer a estimação de um número exponencial de pesos quando se emprega o operador $+$. Por outro lado, não utilizar este operador diminuiu a potencialidade do modelo capturar, no dados de treino, a variabilidade da classe de um objeto. Isto consequentemente produziu pesos menores e classificadores pobres.

Figura 3.22: Desempenho global do classificador SMLC, aprendido segundo o *Modelo 1*, *Modelo 2*, *Modelo 3* e *Modelo 4*.

Por exemplo, se cada atributo possuir m valores diferentes, e o problema de aprendizado supervisionado for multinomial com relação ao rótulo de classe, isto é, há r rótulos possíveis, um algoritmo de aprendizado do *Alchemy* precisaria estimar uma quantidade exponencial de pesos, dada por $r.m^n$, para treinar um classificador. Considerando-se, por exemplo, $n = 5$, $m = 4$ e $r = 2$, seria necessário aprender 2048 pesos.

Em situações como esta, alguns testes introdutórios revelaram que o tempo de computação necessário para efetuar os cálculos de pesos foi extremamente longo. Para exemplificar este fato, o tempo de aprendizado do SMLC para as bases *Breast Cancer*, *Adult* e *Iris* foi medido para o *Modelo 1*. A Figura 3.23 apresenta estes valores.

Consequentemente, usar uma abordagem como *Modelo 1* pode não ser viável em aplicações cuja base de treino seja de grande dimensionalidade. Como uma solução intermediária para este entrave, o *Modelo 2* reduziu substancialmente o tempo de aprendizado

Figura 3.23: Tempos de aprendizado do SMLC para (a) *Modelo 1*, (b) *Modelo 2*, (c) *Modelo 3* e (d) *Modelo 4* para as bases *Breast Cancer*, *Adult* e *Iris*.

do SMLC. A Figura 3.23 apresenta os tempos para este modelo. A explicação para esta redução é análoga a anterior. A fórmula do *Modelo 2* é um predicado de classe implicando uma conjunção de predicados de atributos, a qual escrita sob a forma clausal, se torna um conjunto de cláusulas tal como representado a seguir

$$Modelo\ 2 \left\{ \begin{array}{l} \neg Classe(x, +r) \vee Atributo_1(x, +v_1) \\ \neg Classe(x, +r) \vee Atributo_2(x, +v_2) \\ \vdots \\ \neg Classe(x, +r) \vee Atributo_n(x, +v_n). \end{array} \right.$$

Este modelo mostrou-se computacionalmente mais eficiente. Considerando os mesmos valores n , m e r anteriores, necessário aprender apenas $(r.m)n$ pesos. A redução no número de parâmetros que devem se estimados é notável. Por exemplo, para $n = 5$, $m = 4$ e $r = 2$, seria necessário aprender apenas 40 pesos. Para o *Modelo 3* e *Modelo 4*, foram aprendidos também $(r.m)n$ pesos. Os tempos de aprendizado para estes modelos constam também na Figura 3.23. A forma clausal do *Modelo 4* é igual àquela do *Modelo 2*, e para o *Modelo*

3, a forma clausal é

$$\text{Modelo 3} \left\{ \begin{array}{l} \neg \text{Atributo}_1(\mathbf{x}, +v_1) \vee \text{Classe}(\mathbf{x}, +r) \\ \neg \text{Atributo}_2(\mathbf{x}, +v_2) \vee \text{Classe}(\mathbf{x}, +r) \\ \vdots \\ \neg \text{Atributo}_n(\mathbf{x}, +v_n) \vee \text{Classe}(\mathbf{x}, +r). \end{array} \right.$$

Nota-se pela Figura 3.23 que, para uma base com número reduzido de rótulos, atributos e seus valores, o treinamento de um classificador pode ser extremamente rápido mesmo quando o *Modelo 1* é empregado. A base *Iris* é um exemplo disso, para a qual o SMLC levou apenas 6 segundos para ser treinado.

Quanto à influência, sobre o *Modelo 3* e *Modelo 4*, da abordagem generativa ou discriminativa em combinação com inferência por MC-SAT ou MCMC, novamente as diferenças não são relevantes para os valores das métricas tradicionais. Como exemplo, observando-se a Figura 3.13, que retrata o desempenho do classificador com *Modelo 3* para a base *Iris*, quando se fixa o aprendizado em generativo, os valores de acurácia, precisão e revocação são iguais para ambos os tipos de inferência e, na Figura 3.14, tais valores praticamente não mudam.

Ainda considerando esta figura, quando se escolhe a forma generativa de aprendizado, a probabilidade de acerto não sofre variação significativa ao usar o MCMC (97%) ou o MC-SAT (99%). Porém, quando se adota a tipo discriminativo, esta diferença é notável. Usar o classificador com inferência do tipo MCMC leva ao valor médio de 77% para os objetos corretamente classificados com este método. Já o MC-SAT aumenta esse valor em vinte pontos percentuais, passando a 97%.

Por fim, as Figuras 3.15 a 3.20 proporcionam uma comparação entre o SMLC treinado com base no *Modelo 3*, e outros classificadores. Talvez esta comparação pareça injusta, mas não é. A intenção não é confrontar em detalhes cada classificador, tampouco eleger o melhor deles, o que seria ingênuo, dado que cada um baseia-se em técnicas distintas para classificar um dado. O objetivo ao mostrar estes resultados é apenas posicionar um classificador aprendido com uma técnica nova, no caso as redes lógicas de Markov, em um cenário atual de desempenho de alguns métodos notáveis centrados em linguagens distintas (e.g., modelos gráficos probabilísticos, árvores, regras) e usualmente empregados em problemas reais.

Em comparação com os classificadores baseados em regras, o SMLC apresentou acu-

rácia claramente superior para todas as bases quando comparado com o *ZeroR*, um classificador primitivo e utilizado como padrão de *benchmark*. Apresentar resultados piores que aqueles obtidos com o *ZeroR* pode significar um sério problema de *overfitting*. Com relação ao *Prism* e ao *Decision Table*, o SMLC foi competitivo. Diante dos métodos fundamentados em árvores, o SMLC teve valores de acurácia superiores em alguns casos e próximos em outros. Frente aos classificadores baseados em redes Bayesianas, cujos pontos fortes são a eficiência computacional e a acurácia, o SMLC apresentou resultados também expressivos.

Esta seção apresentou uma discussão do desempenho de RLMs e do pacote *Alchemy* quando aplicados em problemas de aprendizado supervisionado de classificadores. As Figuras 3.21 e 3.22 revelaram que RLMs proporcionam classificadores apurados em diversas bases de dados. Sobre o pacote *Alchemy*, tanto abordagens generativas como discriminativas tiveram êxito na construção de classificadores. Quanto aos métodos de inferência empregados, o MC-SAT e o MCMC tiveram desempenho elevado. Caso se queira eleger o melhor modelo, tanto o *Modelo 3* como o *Modelo 4* seria adequado, pois ambos apresentaram bom desempenho.

A discussão sobre o número de pesos que devem ser estimados para certa formulação em RLMs e a Figura 3.23 trouxeram informações relevantes sobre tempos necessários para aprender um classificador, considerando diversas abordagens (*Modelo 1*, *Modelo 2*, etc). Por fim, uma comparação com outros métodos usuais de classificação disponíveis na literatura demonstra que o SMLC também pode ser uma opção interessante para problemas de classificação supervisionada.

Resultados de Classificadores Especificados com Conhecimento Prévio

Nesta seção são apresentados os resultados alcançados com algumas especificações de modelos as quais permitem explorar a expressividade de RLMs. Para realçar a adequação de RLMs para abordagem de problemas com estrutura relacional complexa, experimentos com a base de dados *Tic-Tac-Toe* foram desenvolvidos. Ao todo, sete tipos de modelos foram testados.

Os dois primeiros foram especificados sem conhecimento, apenas regras de diagnóstico e causa foram usadas. Nos cinco modelos seguintes, o conhecimento é gradualmente introduzido. Pretende-se com isso demonstrar o contraste de desempenho existente entre especificações sem e com conhecimento prévio.

As Figuras 3.24 e 3.25 apresentam o desempenho dos classificadores aprendidos. Nestes testes, foram usados apenas a forma generativa de aprendizado e o método MC-SAT para inferência. Na primeira figura, constam resultados em termos de acurácia, precisão e revocação. Na segunda, algumas medidas de tempo necessário para treinamento são reportadas.

Para representar o problema, emprega-se a denominação a seguir para os predicados de atributo da base *Tic-Tac-Toe*: $\text{TopLeft}(\mathbf{r}, \mathbf{v})$, $\text{TopMiddle}(\mathbf{r}, \mathbf{v})$ e $\text{TopRight}(\mathbf{r}, \mathbf{v})$ para as três posições da linha superior do tabuleiro, sendo $\mathbf{r}$ e $\mathbf{v}$ variáveis que representam a instância observada e o valor do atributo nessa instância, respectivamente. $\text{MiddleLeft}(\mathbf{r}, \mathbf{v})$, $\text{MiddleMiddle}(\mathbf{r}, \mathbf{v})$ e $\text{MiddleRight}(\mathbf{r}, \mathbf{v})$ para as três posições da linha intermediária do tabuleiro, e $\text{BottomLeft}(\mathbf{r}, \mathbf{v})$, $\text{BottomMiddle}(\mathbf{r}, \mathbf{v})$, $\text{BottomRight}(\mathbf{r}, \mathbf{v})$ para as três posições da última linha. $\text{Winner}(\mathbf{r}, \mathbf{w})$ representa o predicado de classe, em que $\mathbf{w}$ é o jogador.

Nos primeiros dois experimentos, as formulações são desprovidas de qualquer informação sobre o funcionamento do jogo. Uma delas baseia-se em regras de causa e a outra em regras de diagnóstico. Os modelos testados, chamados *Modelo de Causa* e *Modelo de Diagnóstico*, apresentam as seguintes formulações

$$\text{Modelo de Causa} - \text{Winner}(\mathbf{r}, +\mathbf{w}) \Rightarrow \text{TopLeft}(\mathbf{r}, +\mathbf{v}) \wedge \dots \wedge \text{BottomRight}(\mathbf{r}, +\mathbf{v}).$$

$$\text{Modelo de Diagnóstico} - \text{TopLeft}(\mathbf{r}, +\mathbf{v}) \wedge \dots \wedge \text{BottomRight}(\mathbf{r}, +\mathbf{v}) \Rightarrow \text{Winner}(\mathbf{r}, +\mathbf{w}).$$

O desempenho dos classificadores gerados com estes modelos pode ser visto na Figura 3.24, para os cenários *Causa* e *Diagnóstico*. Na Figura 3.25 tem-se o tempo necessário para aprender cada classificador.

Em uma terceira abordagem, assumiu-se que uma parte reduzida do conhecimento correto para modelar o comportamento do jogo tornou-se disponível a partir de informações obtidas previamente. Pôde-se então, especificar o novo modelo, chamado de *Modelo CP1* (Modelo com Conhecimento Parcial 1), ainda simples e facilmente adaptado a partir do experimento anterior, porém com uma única fórmula semanticamente adequada ao jogo, dada por

$$\text{Modelo CP1} - \text{TopRight}(\mathbf{r}, +\mathbf{v}) \wedge \text{MiddleRight}(\mathbf{r}, +\mathbf{v}) \wedge \text{BottomRight}(\mathbf{r}, +\mathbf{v}) \Rightarrow \text{Winner}(\mathbf{r}, +\mathbf{w}),$$

o qual captura as chances de um jogador marcar as três posições da coluna à direita do tabuleiro e, conseqüentemente, sagrar-se vencedor. Os resultados destes testes estão representados na Figura 3.24, para um nível de conhecimento denominado *Parcial 1*.

Imaginando agora que o conhecimento disponibilizado aumente gradual e correta-

Figura 3.24: Desempenho do SMLC considerando diversos níveis de conhecimento prévio introduzido na modelagem do classificador com RLMs.

mente, são estabelecidos mais três modelos para três novos níveis de conhecimento chamados *Parcial 2*, *Parcial 3* e *Parcial 4*. A formulação para o *Parcial 2*, chamada de *Modelo CP2*, é

$$\text{Modelo CP2} \begin{cases} \text{BottomLeft}(r, +v) \wedge \text{BottomMiddle}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopRight}(r, +v) \wedge \text{MiddleRight}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w). \end{cases}$$

O *Modelo CP2* enriquece o anterior, *Modelo CP1*, através da adição de uma fórmula que captura as chances de um jogador marcar as três posições inferiores do tabuleiro e sagrar-se vencedor. O nível *Parcial 3* estende o *Parcial 2*, porém, introduz-se desta vez informação prévia modelando-se relações que capturam as chances de vitória quando um jogador consegue marcar todas as casas da segunda linha do tabuleiro, ou a segunda coluna deste. As sentenças lógicas que definem o *Modelo CP3* são

$$\text{Modelo CP3} \begin{cases} \text{MiddleLeft}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{MiddleRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopMiddle}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{BottomMiddle}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{BottomLeft}(r, +v) \wedge \text{BottomMiddle}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopRight}(r, +v) \wedge \text{MiddleRight}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w). \end{cases}$$

Os resultados para estes níveis de conhecimento estão representados na Figura 3.24. Por fim, o último nível parcial de conhecimento se aproxima de um modelo ideal, isto é, implementado com conhecimento total e correto do comportamento do jogo. O modelo

com nível *Parcial 4*, chamado de *Modelo CP4*, foi implementado também como uma extensão do nível *Parcial 3*, capturando as chances de vitória quando um jogador marca as posições da linha superior ou as posições da coluna da esquerda. A formulação testada foi

$$\text{Modelo CP4} \left\{ \begin{array}{l} \text{TopLeft}(r, +v) \wedge \text{TopMiddle}(r, +v) \wedge \text{TopRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopLeft}(r, +v) \wedge \text{MiddleLeft}(r, +v) \wedge \text{BottomLeft}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{MiddleLeft}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{MiddleRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopMiddle}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{BottomMiddle}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{BottomLeft}(r, +v) \wedge \text{BottomMiddle}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopRight}(r, +v) \wedge \text{MiddleRight}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w). \end{array} \right.$$

O conhecimento utilizado nestas modelagens ainda não é completo. Para ilustrar o desempenho de um classificador modelado em uma situação ideal em que o conhecimento é pleno e correto, uma última formulação foi testada, a qual estende as anteriores, mas captura integralmente o comportamento do jogo. Nesta formulação, especifica-se um modelo com conhecimento correto, chamado *Modelo Correto*, com oito sentenças lógicas representativas das oitos configurações possíveis geradoras de vitória para um jogador. A formulação testada foi

$$\text{Modelo Correto} \left\{ \begin{array}{l} \text{TopLeft}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopRight}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{BottomLeft}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopLeft}(r, +v) \wedge \text{TopMiddle}(r, +v) \wedge \text{TopRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopLeft}(r, +v) \wedge \text{MiddleLeft}(r, +v) \wedge \text{BottomLeft}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{MiddleLeft}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{MiddleRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopMiddle}(r, +v) \wedge \text{MiddleMiddle}(r, +v) \wedge \text{BottomMiddle}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{BottomLeft}(r, +v) \wedge \text{BottomMiddle}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w) \\ \text{TopRight}(r, +v) \wedge \text{MiddleRight}(r, +v) \wedge \text{BottomRight}(r, +v) \Rightarrow \text{Winner}(r, +w). \end{array} \right.$$

A diferença deste nível de conhecimento para o imediatamente anterior é que agora sabe-se que um jogador também pode vencer uma partida ao marcar a diagonal primária ou a secundária⁵ do tabuleiro. Os resultados para este modelo ideal de aprendizado de classificador para a base *Tic-Tac-Toe* são também apresentados na Figura 3.24.

Esta seção apresentou alguns resultados alcançados através de experimentos em que

⁵Imaginando-se o tabuleiro como uma matriz quadrada de dimensão 3x3.

Figura 3.25: Tempo de treinamento do SMLC para diversos níveis de conhecimento.

RLMs foram utilizadas como linguagem para especificação e treinamento de classificadores supervisionados, quando se dispõe de conhecimento prévio possível de ser inserido na modelagem em RLM. A seguir, os resultados destes experimentos são discutidos.

Discussão dos Resultados em Aprendizado com Conhecimento Prévio

Discute-se agora a relevância das RLMs quando o problema de aprendizado de um classificador envolve dados com estrutura intrinsecamente relacional, isto é, quando as relações nestes dados seguem um comportamento bem definido, o qual nem sempre pode ser capturado com sentenças lógicas de aplicação geral, tais como os modelos em RLMs propostos na Seção 3.1.1 e discutidos anteriormente.

As Figuras 3.24 e 3.25 expressam alguns resultados para a base de dados *Tic-Tac-Toe*. Na primeira, para o classificador modelado com regra de diagnóstico, nota-se que seu desempenho foi extremamente inferior aos demais classificadores aprendidos para outras bases de dados (e.g., *Adult*, *Breast Cancer* e *Iris*). Isso ratifica a intuição de que nem sempre uma sentença lógica que foca o diagnóstico de uma classe é adequada para modelar um problema de aprendizado em RLM. Além disso, o tempo de treinamento, dado pela Figura 3.25, revela a ineficiência que esta abordagem implica, pois foram necessárias cerca de 4,8 horas (286,33 minutos) para treinar o classificador.

Por outro lado, especificando-se um modelo de causa, as medidas de probabilidade de acerto e acurácia melhoraram substancialmente, mantendo-se todas acima de 90%.

Somando-se a isto, o tempo de aprendizado sofreu queda considerável, passando a cerca de 1,24 horas (74,33 minutos). Este resultado já seria suficiente para este problema de aprendizado, porém, o *Modelo de Causa* também pode se tornar extremamente ineficiente, dependendo das características da base de dados para a qual se queira treinar um classificador (e.g. número de atributos, valores possíveis para cada atributo e para a classe).

Em vista disso, o *Modelo CP1*, para um nível de conhecimento chamado *Parcial 1*, foi extremamente rápido, levando apenas 3,73 minutos para ter um classificador treinado. Contudo, um modelo semanticamente incompleto como este, isto é, que atinge parcialmente o comportamento dos dados da base *Tic-Tac-Toe*, ainda não é capaz de produzir um classificador apurado. As métricas na Figura 3.24 demonstram esse fato.

Nos demais testes, isto é, *Modelo CP2*, *Modelo CP3* e *Modelo CP4*, o conhecimento foi incrementado gradualmente (e.g., *Parcial 2*, *Parcial 3* e *Parcial 4*) de forma a elevar o desempenho do SMLC. Nestes casos, todas as quatro medidas de avaliação sofreram melhoras graduais e sensíveis, e o tempo de treinamento sofreu pequena alteração frente ao *Modelo CP1*, com cenário de conhecimento *Parcial 1*. Por fim, um modelo ideal (*Modelo Correto*) com conhecimento pleno chamado *Correto* foi testado, e seus resultados alcançaram um limite no qual o classificador praticamente não comete erros.

Essa sequência de testes demonstra a influência do conhecimento prévio na especificação de um classificador com RLMs. Contudo, o aspecto mais importante é a flexibilidade por RLM para que se possa transitar entre um modelo e outro, sem a necessidade de drástica reformulação matemática na especificação do problema de aprendizado, como é comum com outras técnicas de classificação. Com RLMs, o formalismo matemático e algorítmico para aprendizado e inferência fica por conta dos métodos implementados no pacote *Alchemy*. Cabe ao desenvolvedor da aplicação de classificação apenas modelar o problema com base em algum conhecimento prévio, e ajustá-lo se necessário, à medida que se obtém informação sobre os dados, para gerar o melhor classificador possível. Consequentemente, reusar um modelo torna-se também uma opção natural e ágil ao sondar-se metodologias de classificação com RLMs.

3.2.4 Resultados e Discussão de Testes de RLMs em Aprendizado de Classificadores Semi-supervisionados com Restrições

Esta seção tem dois objetivos. O primeiro é apresentar resultados preliminares obtidos a partir de alguns experimentos realizados para testar redes lógicas de Markov em apren-

dizado de classificadores semi-supervisionados com restrições. O segundo é desenvolver uma discussão do desempenho de redes lógicas de Markov nestes experimentos. Inicialmente, os resultados de classificação para o contexto semi-supervisionado são apresentados e, posteriormente, estes são discutidos.

Resultados

Quer no treinamento supervisionado, quer no semi-supervisionado, os predicados de atributo são sempre conhecidos (i.e., são condicionantes). No entanto, o predicado de classe está oculto em parcela substancial dos dados no contexto semi-supervisionado. Isto justifica o emprego de uma forma de aprendizado que priorize estimar probabilidades condicionais de um predicado de classe oculta, dados os atributos do problema. Desta forma, optou-se inicialmente por testar o aprendizado discriminativo, que aprende pesos a partir de probabilidades condicionais como $P(\text{Classe} = c | \text{Atributo}_1 = a_1, \dots, \text{Atributo}_n = a_n)$. No entanto, esta abordagem foi extremamente ineficiente, tomando tempo em excesso para treinar um classificador.

Escolheu-se então a forma generativa de aprendizado, que estima pesos a partir de probabilidades conjuntas como $P(\text{Classe} = c, \text{Atributo}_1 = a_1, \dots, \text{Atributo}_n = a_n)$. Os tempos de computação se tornaram aceitáveis, e os classificadores aprendidos tiveram seu desempenho melhorado. Quanto ao método de inferência testado, empregou-se apenas o MAP, que estima a classe mais provável de um objeto. Na realização dos experimentos, foram aprendidos classificadores para a base *Iris*.

As Figuras 3.26 e 3.27 apresentam o desempenho do classificador aprendido, chamado *Semi-supervised Markov Logic Classifier* (SSMLC), segundo dois cenários. Na primeira figura, o classificador foi aprendido considerando-se apenas 10% de dados rotulados, e na segunda figura, 20%. Em ambos os experimentos, foi investigado se o desempenho de um classificador treinado com relações de restrição, tal como o *Modelo SSMLC*, seria superior a um modelo que desconsidera a supervisão alternativa que as restrições oferecem. As restrições foram inseridas para todos os pares de dados que não estavam rotulados. Para elevar os pesos das fórmulas que envolvem predicados de restrição e predicados de classe oculta, também foram inseridas nos dados de treino observações de predicados de restrição para pares de objetos rotulados.

A Figura 3.28 traz uma comparação do SSMLC com outros métodos não-supervisionados cujo sucesso em problemas reais é reconhecido na literatura: o algoritmo de maximização de verossimilhança EM (*Expectation-Maximization*) e o algoritmo *K-means*, que foram

Figura 3.26: Desempenho do SSMLC aprendido para a base *Iris* para um cenário semi-supervisionado com 10% dos dados rotulados.

utilizados a partir do pacote *WEKA*. Diversos trabalhos têm utilizado estes métodos, dentre eles, pode-se citar Basu *et al.* (2006), Frade e Kanade (2006), Law *et al.* (2005a), Shental *et al.* (2003), Cozman (2000) e Wells III (1993).

Figura 3.27: Desempenho do SSMLC aprendido para a base *Iris* para um cenário semi-supervisionado com 20% dos dados rotulados.

Esta seção apresentou resultados de alguns experimentos preliminares de RLMs no aprendizado de classificadores semi-supervisionados. A discussão dos resultados destes experimentos pode ser acompanhada na seção a seguir.

Discussão

As métricas de avaliação para o classificador SSMLC revelaram um desempenho insatisfatório. A razão básica deste desempenho inferior está na concepção da forma como é feita a estimativa de um peso para certa fórmula de uma RLM. Simplesmente, um peso é aprendido através de uma contagem do número de mundos em que tal fórmula é logicamente válida. Feito isso, converte-se o valor deste peso para uma medida de probabilidade

Figura 3.28: Comparação de valores de acurácia entre o SSMLC e outros métodos não-supervisionados, empregando-se a base *Iris*.

a priori associada a esta fórmula.

Como em RLMs pesos altos significam probabilidades elevadas, estimá-los com valores baixos gera fórmulas pouco prováveis e, conseqüentemente, classificadores pouco acurados. Este fato permite concluir que a observação de um predicado *query* (i.e., predicado que se quer classificar) nos dados de treino de um classificador modelado com RLM é extremamente importante para que se obtenha boas estimativas de pesos. Como no aprendizado semi-supervisionado a parcela de dados rotulados é extremamente inferior àquela não-rotulada, os pesos estimados foram baixos, o que se traduziu em um desempenho fraco.

Mesmo a inserção de relações de restrição não possibilitou o treinamento de classificadores de desempenho elevado, em comparação com cenários em que a única forma de supervisão foi o rótulo de classe. As Figuras 3.26 e 3.27 comprovam esta conclusão para o SSMLC da base *Iris*. Ao treiná-lo com o *Modelo SSMLC*, que incorpora quatro fórmulas envolvendo restrições, o desempenho não foi superior frente à formulação sem restrições.

Em comparação com outros métodos tradicionais para classificação não-supervisionada, conforme ilustrado na Figura 3.28, o SSMLC apresentou acurácia média inferior. O algoritmo EM e o método *K-means*, métodos que não usam dados rotulados para treinar um classificador, foram claramente superiores. No entanto, se as RLMs ainda não proporcionam o aprendizado de classificadores semi-supervisionados com restrições com desempenho à altura de outros métodos de sucesso na literatura, um aspecto vantajoso da linguagem é a facilidade e simplicidade com que se pode especificar um modelo de classificador com restrições e, caso seja necessário, estendê-lo para testes em novas aplicações.

Nos experimentos conduzidos, adaptar o *Modelo 2* (proposto no aprendizado supervisionado) para o contexto com restrições, criando o modelo para o SSMLC, não exigiu grande esforço. Foi necessário apenas inserir o predicado de classe oculta, representado por $\text{ClaOc}(\mathbf{x}, \mathbf{r})$, outros dois predicados de restrição, denotados por $\text{ClaIg}(\mathbf{x}, \mathbf{y})$ e $\text{ClaDif}(\mathbf{x}, \mathbf{y})$,

e quatro novas fórmulas lógicas (vide Seção 3.1.2) envolvendo predicados de restrição, atributo e classe.

Comenta-se agora algumas limitações de sintaxe de RLMs para implementação de classificadores semi-supervisionados com restrições. A primeira reside na dificuldade de manipulação de expressões matemáticas mais complexas, tais como fórmulas para cálculo de distâncias e/ou somatórios, ambos muito empregados em tarefas semi-supervisionadas (e.g., *clustering* semi-supervisionado com restrições, abordado nos trabalhos de Shental *et al.* (2003) e Basu *et al.* (2004)) que se baseiam em medidas de similaridade dadas por distâncias. O algoritmo *K-means* por exemplo, usa distâncias médias entre objetos para decidir a que classe estes pertencem.

Com mais liberdade na expressividade matemática de linguagem, seria possível por exemplo, modelar predicados envolvendo distâncias e utilizá-los como conhecimento prévio. Isso propiciaria uma forma alternativa de fornecer alguma supervisão ao treinamento, sem necessariamente determinar rótulos. O ganho com uma abordagem como esta residiria no fato de que pesos baixos aprendidos para fórmulas envolvendo predicados de classe oculta, não observados durante o treinamento, não seriam tão determinantes para o desempenho pobre de um classificador, dado que predicados de similaridade baseado em distâncias entre instâncias poderiam ser calculados durante o aprendizado.

Como decorrência da inabilidade de lidar com números reais, RLMs em sua implementação no pacote *Alchemy* não permite um manuseio adequado de atributos cujos argumentos sejam variáveis contínuas. Tal fato gera a necessidade de discretização de dados, i.e., criação de categorias. Em problemas de *clustering*, cujos dados são naturalmente contínuos em inúmeras aplicações (como é o caso de algumas das bases de dados testadas, que tiveram seus atributos discretizados), implementar um modelo em RLM pode comprometer os resultados. Pequenas variações no valor de certo atributo, o que poderia influenciar a inferência de um rótulo de classe, podem ser encobertas por uma discretização inadequada. Torna-se difícil então, descobrir como categorizar adequadamente uma base de dados antes de conduzir um experimento com RLMs.

Sobre o algoritmo de inferência MAP, empregado nos testes dos classificadores SSMLC, este gerou resultados coerentes com a intuição. Como a principal forma de supervisão, i.e., predicados de classe, estava suprimida durante o treinamento, era esperada uma queda sensível nos valores de acurácia, precisão e revocação, em comparação com os valores elevados destas mesmas métricas, obtidos nos experimentos em aprendizado supervisionado.

Quanto a técnica de aprendizado utilizada, inicialmente empregou-se a discriminativa

que, segundo Richardson e Domingos (2006), é mais eficiente do que a abordagem generativa. Intuitivamente ele era uma opção natural, pois no aprendizado de pesos para fórmulas, o método precisa estimar também uma distribuição condicional do predicado de classe dados seus predicados de atributos. Esse é de fato o cenário de aprendizado dos experimentos conduzidos (predicados de atributos são sempre condicionantes, e os de classe são *query*). Entretanto, o método discriminativo consumiu tempo demasiado para a computação de pesos, mostrando-se ineficiente. Em alguns casos, a execução do algoritmo se desenvolveu por dias, gerando classificadores muito pobres. Restou apenas testar o algoritmo generativo, que foi capaz de estimar tais pesos em tempo hábil.

Sintetizando em poucas palavras a discussão da aplicação de RLMs em aprendizado semi-supervisionado, a principal vantagem consiste, novamente, na facilidade que a sintaxe da linguagem oferece para a modelagem de classificadores. Por outro lado, RLMs ainda carecem de funcionalidades melhores para tratar problemas de aprendizado que empregam porção substancial de dados não-rotulados.

4 *Conclusão e Trabalhos Futuros*

A meta principal desta dissertação foi testar redes lógicas de Markov como linguagem lógico-probabilística para o aprendizado de classificadores automáticos de dados, considerando dois grandes paradigmas de aprendizado: supervisionado e semi-supervisionado com restrições. Secundariamente, um pacote de algoritmos próprios para aprendizado e inferência em RLMs chamado *Alchemy* foi também testado e avaliado nos experimentos conduzidos.

As contribuições deste trabalho foram duas. A primeira delas foi a discussão da aplicabilidade da linguagem à especificação de classificadores nos paradigmas citados no parágrafo anterior, com posterior constatação, fundamentada em diversos experimentos de classificação, de que RLMs oferecem um formalismo promissor principalmente para o treinamento supervisionado de classificadores.

Nos testes de desempenho com modelos (*Modelo 1*, *Modelo 2*, *Modelo 3* e *Modelo 4*) baseados em regras lógicas gerais (regras de diagnóstico, causa) e sem conhecimento prévio embutido, a linguagem mostrou-se versátil, permitindo o treinamento de classificadores supervisionados apurados. As Figuras 3.3 a 3.14 demonstram tais resultados para cada modelo.

Em comparação com outras técnicas de classificação (e.g., com base em regras, árvores e redes bayesianas) supervisionada disponíveis na literatura, o SMLC apresentou acurácia superior aos demais em diversos casos, e revelou-se à altura de métodos consagrados como o *Naive Bayes* e TAN. As Figuras 3.15 a 3.20 confirmam o desempenho elevado do classificador aprendido.

A aplicação de RLMs em domínios intrinsecamente relacionais também teve êxito nos resultados de classificação. Para a base de dados *Tic-Tac-Toe*, sete modelos foram investigados. Os dois primeiros (*Modelo de Causa* e *Modelo de Diagnóstico*) exploraram o potencial de formulações em RLM desprovidas de conhecimento, quando aplicadas no treinamento de classificadores para uma base de dados relacional.

O *Modelo de Dianóstico* teve desempenho consideravelmente inferior. As métricas de avaliação geradas foram baixas, e o tempo de aprendizado foi extremamente alto. Já o *Modelo de Causa* proporcionou bons resultados de classificação, porém, dependendo da dimensionalidade do problema (e.g., número de atributos, valores possíveis, etc), o tempo de treinamento (que foi aceitável para a base empregada) pode tornar esta abordagem proibitiva.

Já os cinco demais modelos (e.g., *Modelo CP1*, *Modelo CP2*, *Modelo CP3*, *Modelo CP4* e *Modelo Correto*), focados em realçar a expressividade de redes lógicas de Markov, demonstraram o ganho (em acurácia e tempo de treinamento) de desempenho que se obtém, frente a especificações mais gerais (*Modelo de Causa* e *Modelo de Diagnóstico*), quando conhecimento está embutido no modelo.

Como conclusões principais sobre a linguagem testada tem-se: a flexibilidade na modelagem de problemas, o poder de representatividade de diversos domínios de conhecimento, a simplicidade e facilidade com que se pode fornecer conhecimento prévio à modelagem de um classificador e, também, a possibilidade de extensão e reuso de um modelo de aprendizado já implementado. Estes aspectos fazem de RLMs uma técnica útil e merecedora de novos estudos de aplicações no contexto de classificação supervisionada especialmente.

Quanto ao paradigma semi-supervisionado com restrições, experimentos preliminares revelaram que a linguagem ainda não está adequada para a construção de classificadores com desempenho elevado. As métricas de avaliação demonstraram classificadores pobres aprendidos neste paradigma. Mesmo com a inserção de relações envolvendo restrições, da forma como isso foi proposto nesta dissertação, os resultados alcançados não foram satisfatórios.

Ainda no campo semi-supervisionado, RLMs carecem de mecanismos mais desenvolvidos para o tratamento numérico (e.g., somatórios, cálculos com números reais, mecanismos iterativos) necessário em muitos problemas semi-supervisionados, e mesmo supervisionados e não-supervisionados. Talvez, novas implementações que forneçam subsídios próximos àqueles de uma linguagem de programação tais como *loops* e condições pudessem estender os domínios de aplicação desta linguagem.

A segunda contribuição deste trabalho consistiu nos testes de alguns dos algoritmos de aprendizado e inferência do pacote *Alchemy*, em especial, as abordagens generativa e discriminativa de aprendizado de pesos, e os métodos de inferência probabilísticas MCMC e MC-SAT. Em uma visão geral, treinar um classificador supervisionado de forma generativa ou discriminativa não causou impacto substancial nas métricas tradicionais de

avaliação de desempenho: acurácia, precisão e revocação. Nos experimentos de aprendizado supervisionado, todas elas alcançaram valores elevados, acima de 90% para as bases *Iris* e *Breast Cancer*, e entre 80% e 90% para a base *Adult*.

Apenas com relação à medida de avaliação proposta chamada *probabilidade de acerto*, notou-se a influência do tipo de aprendizado adotado. Quando se treinou um classificador de forma discriminativa, empregando o método MC-SAT para inferência da classe, observou-se sensível melhora na certeza com que um rótulo de classe foi inferido, em diversos cenários de testes. Em outras palavras, com o MC-SAT, as classes inferidas apresentaram probabilidades maiores em comparação àquelas obtidas com o MCMC, do que se pode concluir que, para os experimentos deste trabalho, um algoritmo lógico-probabilístico como o MC-SAT é adequado para inferência em redes de Markov proposicionalizadas a partir de um modelo em RLMs.

Esta dissertação propôs também um esquema, representado por um fluxograma, que auxilia na especificação de um classificador com RLMs. Neste fluxograma, redesenhado do trabalho de Kotsiantis (2007), as etapas principais são a construção de um modelo de aprendizado em RLMs, o aprendizado de pesos para obtenção de um classificador, e uma etapa de ajuste deste modelo, na qual as sentenças lógicas representativas do problema podem ser refinadas ou mesmo reprojetadas para treinamento de um classificador mais acurado, preciso e com melhor revocação.

Acerca de oportunidades de trabalhos futuros, uma opção natural que daria continuidade a esta dissertação seria a extensão de RLMs para tornar possível o treinamento com sucesso de classificadores semi-supervisionados e, também, não-supervisionados, já que linguagem oferece o formalismo e expressividade da LPO para modelagem de diversos domínios. Nesta linha, a literatura tem demonstrado a existência de metodologias envolvendo LPO e medidas de similaridade aplicadas em *clustering* relacional.

Pode-se citar o trabalho de Bisson (1992), que desenvolveu um sistema para tarefas de *clustering* empregando representações de primeira ordem com extensões para manipular eficientemente números e conhecimento procedural. Na publicação de Ramon e Bruynooghe (1998), tem-se uma metodologia para a definição de distâncias entre átomos em LPO, para posterior emprego em classificação não-supervisionada. Há também propostas de extensão do próprio algoritmo *K-means* para aplicações com LPO, como é o caso do trabalho de Kirsten e Wrobel (2000).

Outra possibilidade futura seria a combinação de RLMs com sistemas de aprendizado de regras da área de mineração de dados. Na ausência de qualquer conhecimento prévio

para modelar um classificador com RLMs, um sistema como esse poderia ser útil na descoberta de relações interessantes entre atributos em grande bases de dados, sendo tais relações traduzidas para RLMs, a fim de proporcionar um classificador cujas estruturas relacionais em sua especificação possam capturar corretamente o “comportamento” de dados usados para treino.

Referências Bibliográficas

- ADAMS, I. D.; CHAN, M.; CLIFFORD, P. C.; COOKE, W. M.; DALLOS, V.; DOMBAL, F. T. de; EDWARDS, M. H.; HANCOCK, D. M.; HEWETT, D. J.; MCINTYRE, N. Human computer-aided diagnosis of abdominal pain: further report with emphasis on performance of clinicians. *British Medical Journal*, v. 1, p. 376–380, 1974.
- BACCHUS, F. *Representing and Reasoning with Probabilistic Knowledge: A Logical Approach*. Cambridge: MIT Press, 1990.
- BASU, S.; BILENKO, M.; BANERJEE, A.; MOONEY, R. Probabilistic semi-supervised clustering with constraints. In: CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. (Ed.). *Semi-Supervised Learning*. Cambridge, MA, USA: MIT Press, 2006. p. 73–102.
- BASU, S.; BILENKO, M.; MOONEY, R. J. A probabilistic framework for semi-supervised clustering. In: *KDD '04: Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. New York, NY, USA: ACM Press, 2004.
- BESAG, J. Statistical analysis of non-lattice data. *The Statistician*, Blackwell Publishing for the Royal Statistical Society, v. 24, n. 3, p. 179–195, 1975. ISSN 00390526. Disponível em: <<http://www.jstor.org/stable/2987782>>.
- BILENKO, M.; BASU, S.; MOONEY, R. J. Integrating constraints and metric learning in semi-supervised clustering. In: *ICML '04: Proceedings of the twenty-first international conference on Machine learning*. New York, NY, USA: ACM, 2004. p. 11+. ISBN 1-58113-828-5. Disponível em: <<http://dx.doi.org/10.1145/1015330.1015360>>.
- BISSON, G. Conceptual clustering in a first order logic representation. In: *ECAI '92: Proceedings of the 10th European Conference on Artificial Intelligence*. New York, NY, USA: John Wiley & Sons, Inc., 1992. p. 458–462. ISBN 0-471-93608-1.
- CASTELLI, V. *The Relative Value of Labeled and Unlabeled Samples in Pattern Recognition*. Tese (Doutorado) — Stanford University, December 1994.
- CASTELLI, V.; COVER, T. M. On the exponential value of labeled samples. *Pattern Recognition Letters*, v. 16, p. 105–111, 1995.
- CHAPELLE, O.; SCHÖLKOPF, B.; ZIEN, A. *Semi-Supervised Learning*. Cambridge, Massachusetts, USA: MIT Press, 2006.
- COHEN, I.; COZMAN, F.; SEBE, N.; CIRELO, M. C.; HUANG, T. Semisupervised learning of classifiers: Theory, algorithms, and their application to human-computer interaction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 26, n. 12, p. 1553–1568, 2004.

- COWELL, R. G.; DAWID, A. P.; LAURITZEN, S. L.; SPIEGELHALTER, D. J. *Probabilistic Networks and Expert Systems*. New York: Springer-Verlag, 1999. 256 p.
- COZMAN, F. G. Removing redundancy in the expectation-maximization algorithm for Bayesian network learning. In: BARROS, L. N. de; MARCONDES, R.; COZMAN, F. G.; COSTA, A. H. R. (Ed.). *Workshop on Probabilistic Reasoning in Artificial Intelligence*. São Paulo: Editora Tec Art, 2000. p. 21–26.
- COZMAN, F. G.; COHEN, I. Unlabeled data can degrade classification performance of generative classifiers. In: *Proceedings of the Fifteenth International Florida Artificial Intelligence Research Society Conference*. Pensacola, Florida. 2002. p. 327–331.
- DEMPSTER, A. P.; LAIRD, N. M.; RUBIN, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Statistical Society B*, v. 44, p. 1–38, 1977.
- DEVROYE, L. *A course in density estimation*. Cambridge, MA, USA: Birkhauser Boston Inc., 1987.
- DEVROYE, L.; GYORFI, L.; LUGOSI, G. *A Probabilistic Theory of Pattern Recognition*. New York: Springer Verlag, 1996.
- FAYYAD, U. M.; IRANI, K. B. Multi-interval discretization of continuousvalued attributes for classification learning. In: *Thirteenth International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers, 1993. v. 2, p. 1022–1027.
- FRADE, F. D. la T.; KANADE, T. Discriminative cluster analysis. In: *International Conference on Machine Learning*. New York, NY, USA: ACM Press, 2006. v. 148, p. 241 – 248.
- FRIEDMAN, N.; GEIGER, D.; GOLDSZMIDT, M. Bayesian network classifiers. *Machine Learning*, v. 29, n. 2-3, p. 131–163, 1997.
- FRIEDMAN, N.; KOLLER, L. G. ad D.; PFEFFER, A. Learning probabilistic relational models. In: *International Joint Conference on Artificial Intelligence*. 1999. p. 1300–1309.
- GENESERETH, M. R.; NILSSON, N. J. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann Publishers, 1987.
- GETOOR, L.; TASKAR, B. *Introduction to Statistical Relational Learning*. MIT Press, 2007. ISBN 0262072882.
- GILKS, W. R.; RICHARDSON, S.; SPIEGELHALTER, D. J. *Markov Chain Monte Carlo in Practice*. London, England: Chapman and Hall, 1996.
- HALL, M. A. *Correlation-based Feature Subset Selection for Machine Learning*. Tese (Doutorado) — University of Waikato, Hamilton, New Zealand, 1998.
- HALPERN, J. Y. An analysis of first-order logics of probability. *Artificial Intelligence*, v. 46, p. 311–350, 1990.
- HASTIE, T.; TIBSHIRANI, R.; FRIEDMAN, J. H. *The Elements of Statistical Learning*. Springer, 2001.

- HERBRICH, R. *Learning kernel classifiers: theory and algorithms*. MIT Press, 2002.
- IDE, J.; COLLA, E.; COZMAN, F. Bayesian network classifiers for short period country risk forecasting. In: *Proceedings of the Workshop em Algoritmos e Aplicações de Mineração de Dados (WAAMD)*. 2006. p. 109–112.
- JENSEN, F. V. *Bayesian Networks and Decision Graphs*. New York: Springer Verlag, 2001.
- KAMVAR, S. D.; KLEIN, D.; MANNING, C. D. Spectral Learning. In: *IJCAI*. 2003.
- KERSTING, K.; RAEDT, L. D.; KRAMER, S. Interpreting Bayesian logic programs. In: *AAAI-2000 Workshop on Learning Statistical Models from Relational Data*. 2000.
- KIRSTEN, M.; WROBEL, S. Extending k-means clustering to first-order representations. In: *ILP '00: Proceedings of the 10th International Conference on Inductive Logic Programming*. London, UK: Springer-Verlag, 2000. p. 112–129. ISBN 3-540-67795-X.
- KOK, S.; SINGLA, P.; RICHARDSON, M.; DOMINGOS, P. *The Alchemy system for statistical relational AI*. 2005.
- KOLMOGOROV, A. N. *Foundations of the theory of probability*. New York: Chelsea Publishing Company (translation edited by Nathan Morrison, original from 1933), 1950.
- KOTSIANTIS, S. B. Supervised machine learning: A review of classification techniques. *Informatica*, v. 31, p. 249–268, 2007.
- LAW, M. H. C.; TOPCHY, A.; JAIN, A. K. Model-based clustering with probabilistic constraints. In: *SIAM Data Mining*. 2005.
- LIPPI, M.; FRASCONI, P. Prediction of protein beta-residue contacts by markov logic networks with grounding-specific weights. *Bioinformatics*, 2009.
- LIU, D. C.; NOCEDAL, J.; C, D. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, v. 45, p. 503–528, 1989.
- MADDEN, M. G. On the classification performance of tan and general bayesian networks. *Know.-Based Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 22, n. 7, p. 489–495, 2009. ISSN 0950-7051.
- MANNING, C. D.; RAGHAVAN, P.; SCHÜTZE, H. *Introduction to Information Retrieval*. 1. ed. Cambridge University Press, 2008. Hardcover. ISBN 0521865719.
- MARINARI, E.; PARISI, G. Simulated tempering: A new monte carlo scheme. *Europhysics Letters*, p. 19:451–458, 1992.
- MCLACHLAN, G.; PEEL, D. *Finite Mixture Models (Wiley Series in Probability and Statistics)*. Wiley-Interscience, 2000.
- MUGGLETON, S. *Advances in Inductive Logic Programming*. IOPress, 1996. 254-264 p.
- NEAPOLITAN, R. E. *Learning Bayesian Networks*. Upper Saddle River, NJ, USA: Prentice Hall, 2003.

- NEVILLE, J.; JENSEN, D. Dependency networks for relational data. In: *ICDM*. 2004. p. 170–177.
- NIGAM, K.; MCCALLUM, A. K.; THRUN, S.; MITCHELL, T. Text classification from labeled and unlabeled documents using EM. *Machine Learning*, v. 39, p. 103–144, 2000.
- NILSSON, N. J. Probabilistic logic. *Artificial Intelligence*, v. 28, p. 71–87, 1986.
- PEARL, J. On probability intervals. *International Journal of Approximate Reasoning*, v. 2, p. 211–216, 1988.
- POON, H.; DOMINGOS, P. Sound and efficient inference with probabilistic and deterministic dependencies. In: *AAAI*. AAAI Press, 2006.
- RAMON, J.; BRUYNOOGHE, M. A framework for defining distances between first-order logic objects. In: *Proceedings of 8th International Conference on Inductive Logic Programming (ILP'98), Madison, Wisconsin, USA*. Springer, 1998. (Lecture Notes in Computer Science, v. 1446), p. 271–280.
- RICHARDSON, M.; DOMINGOS, P. Markov logic networks. *Machine Learning*, v. 62, n. 1-2, p. 107–136, 2006.
- RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach (2nd ed.)*. New Jersey: Upper Saddle River, Prentice Hall, 1995.
- SAVAGE, L. J. *The Foundations of Statistics*. New York: Dover Publications, Inc, 1972.
- SHENTAL, N.; BAR-HILLEL, A.; HERTZ, T.; WEINSHALL, D. Computing Gaussian mixture models with EM using equivalence constraints. In: *NIPS*. 2003.
- SINGLA, P.; DOMINGOS, P. Discriminative training of Markov logic networks. In: *AAAI*. AAAI Press / The MIT Press, 2005.
- SINGLA, P.; DOMINGOS, P. Markov logic in infinite domains. In: *Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2007. p. 368–375.
- WAGSTAFF, K.; CARDIE, C.; ROGERS, S.; SCHROEDL, S. Constrained k-means clustering with background knowledge. In: *In ICML*. Morgan Kaufmann, 2001. p. 577–584.
- WEI, W.; ERENDRICH, J.; SELMAN, B. Towards efficient sampling: Exploiting random walk strategies. In: *In AAAI-04*. 2004.
- WELLMAN, M. P.; BREESE, J. S.; GOLDMAN, R. P. From knowledge bases to decision models. *Knowledge Engineering Review*, v. 7, n. 1, p. 35–53, 1992.
- Wells III, W. M. Statistical object recognition with the expectation-maximization algorithm in range-derived features. In: *Computer Vision and Pattern Recognition*. IEEE, 1993. p. 836–850.
- YU, S. X.; SHI, J. Segmentation given partial grouping constraints. *IEEE Trans. Pattern Anal. Mach. Intell.*, IEEE Computer Society, Washington, DC, USA, v. 26, n. 2, p. 173–183, 2004. ISSN 0162-8828.