REVISÃO DE TEORIAS RELACIONAIS PROBABILÍSTICAS ATRAVÉS DE EXEMPLOS COM INVENÇÃO DE PREDICADOS

Kate Cerqueira Revoredo

Tese de Doutorado apresentada ao Programa de Pos-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Gerson Zaverucha

Rio de Janeiro
Janeiro de 2009

REVISÃO DE TEORIAS RELACIONAIS PROBABILÍSTICAS ATRAVÉS DE EXEMPLOS COM INVENÇÃO DE PREDICADOS

Kate Cerqueira Revoredo

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof. Gerson Zaverucha, Ph.D.

Prof. Mario Roberto Folhadela Benevides, Ph.D

Prof. Vítor Manuel de Morais Santos Costa, Ph.D.

Prof. Fabio Gagliardi Cozman, Ph.D.

Prof. Bianca Zadrozny, Ph.D

RIO DE JANEIRO, RJ - BRASIL
JANEIRO DE 2009

Revoredo, Kate Cerqueira

Revisão de Teorias Relacionais Probabilísticas através de Exemplos com Invenção de Predicados/ Kate Cerqueira Revoredo. - Rio de Janeiro: UFRJ/COPPE, 2009

XXI, 154 p.: il.; 29,7 cm.

Orientador: Gerson Zaverucha

Tese (doutorado) - UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2009)

Referências Bibliográficas: p.139-150

1. Aprendizado de máquina. 2. Revisão de Teoria. 3. Rede Bayesiana. 4. Invenção de Predicados. I. Zaverucha, Gerson. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Dedicatória

Às minhas filhas Karen e Gabriela: os melhores presentes que Deus me deu.

Agradecimentos

À minha família em especial,
- a minha querida mãe, Luiza, pelo apoio e incentivo constante em todas as etapas da minha vida e, principalmente por ter sido, além de uma avó maravilhosa para as minhas filhas, uma excelente babá.
- ao meu marido, Leandro, pela enorme paciência e compreensão.
- as minhas filhas, Karen e Gabriela, por me fazerem sorrir nos momentos mais difíceis, me dando forças para continuar.
- ao meu pai, Sam, pelos conselhos e incentivo.
- a minha avó, Iza, por todo o carinho

Ao meu orientador, Professor Gerson Zaverucha, por toda a dedicação, orientação e oportunidades oferecidas durante esses anos de doutorado.

Ao meu co-orientador informal, Vitor Santos Costa pelas conversas, esclarecimentos técnicos e por estar sempre pronto a me ajudar.

A meu co-orientador informal, Luc de Raedt, pelas sugestões e por ter me recebido em seu laboratório para o doutorado sanduíche, me permitindo trabalhar em projetos bastante relevantes e com excelentes pesquisadores.

Aos meus amigos pelo apoio, carinho e grande incentivo, em especial
- a Fernanda Baião pela motivação, pelo suporte e grande exemplo.
- a Aline Paes pela parceria, pelas conversas, trocas de informações e pelo companherismo.
- a Juliana Bernardes, Jonice Oliveira, Carla Delgado, Vinicios Pereira, Matheus Wildemberg, Jairo Souza pelas conversas e grande torcida.
- aos membros do Laboratório de IA, Fábio Jimenez, Luis Rigo, André Nathan e

Ramon Diacovo, pelo suporte técnico e por manterem o bom funcionamento do laboratório.

Aos membros da banca pelos comentários.

Ao CNPq pelo suporte financeiro.

À Deus por minha vida.
"...e porque estreita é a porta e apertado o caminho que leva a vida, e poucos há que a encontrem"(Matheus,7,14)

Obrigada.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

REVISÃO DE TEORIAS RELACIONAIS PROBABILÍSTICAS ATRAVÉS DE EXEMPLOS COM INVENÇÃO DE PREDICADOS

Kate Cerqueira Revoredo

Janeiro/2009

Orientador: Gerson Zaverucha

Programa: Engenharia de Sistemas e Computação

Aprendizado de máquina indutivo tem por objetivo o desenvolvimento de ferramentas e técnicas para induzir modelos a partir de observações e sintetizar novos conhecimentos através de experiências. Diferentes mecanismos de representação de conhecimento, como redes Bayesianas (RB) e teorias probabilísticas de primeira-ordem (TPPO), originam diferentes algoritmos de aprendizado. A maioria deles usa apenas o vocabulário fornecido explicitamente nos dados para construir os modelos. No entanto, informações implícitas sobre objetos do domínio podem enriquecer o processo de aprendizado, muitas vezes tornando o modelo aprendido mais eficiente e/ou compacto. É interessante, então, a extensão automática do vocabulário inicial, com novas estruturas que representem essas informações. Em RB essas estruturas são conhecidas como *variáveis não-observadas* e em aprendizado estatístico relacional *predicados inventados probabilísticos*. Por outro lado, em alguns casos, já existe um modelo que é aproximadamente correto podendo a sua estrutura ser alterada o mínimo possível de forma a refletir corretamente o *dataset*. Temos, então, um caso particular de aprendizado denominado revisão de teoria, a qual utiliza os exemplos para identificar pontos na estrutura do modelo que ao serem modificados melhoram a performance deste, denominados pontos de revisão, propondo modificações apenas neles. Nesta tese, nós investigamos os benefícios da extensão do vocabulário quando revisando modelos probabilísticos, utilizando uma abordagem discriminativa: as variáveis de consulta (em RB)/predicados objetivo (em TPPO) e as suas *Coberturas de Markov* são pontos de revisão. Operadores de revisão, baseados em invenção de novas estruturas, propostos por nós, são aplicados a esses pontos de revisão. Nossa proposta foi aplicada com sucesso em *datasets* artificiais e reais.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

REVISION OF PROBABILISTIC RELATIONAL THEORIES FROM EXAMPLES WITH PREDICATE INVENTION

Kate Cerqueira Revoredo

January/2009

Advisor: Gerson Zaverucha

Program: Systems Engineering and Computer Science

Inductive Machine Learning aims to develop tools and techniques to induce models from observations and synthesize new knowledge through experiences. Different knowledge representation mechanisms, as Bayesian Networks (BN) and probabilistic first-order theories (PFOT), define different learning algorithms. The majority of them only uses the vocabulary provided explicitly in the dataset to construct the models. However, hidden information about domain objects, can enrich the learned models, sometimes making them more efficient and/or compact. Therefore, it is interesting to use techniques to automatically extend the initial vocabulary, considering new structures which represent those hidden informations. In BN, these structures are known as *hidden variables* and in statistical relational learning *probabilistic predicate invented*. From another aspect, an initial model approximately correct can be provided to the learning algorithms. In this case, one has a particular case of learning, called theory revision. Theory revision proposes modifications in potential points of the model indicated by the examples, called revision points, reducing the search space of possible models, using mechanisms, called revision operators, to propose modifications. In this thesis, we investigate the benefits of vocabulary extension when revising probabilistic models. To do so, we use a discriminative approach: the query variables (in BN)/ target predicates (in PFOT) and their Markov Blanket are the revision points. Revision operators, based on new structure introduction, proposed by us, are then applied to these revision points. Our proposals were successfully applied on artificial and real datasets.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ACC Acurácia

BK Conhecimento Preliminar

BLP *Bayesian Logic Program*

CLL Log-verossimilhança condicional

CLP(BN) *Constraint Logic Programming*

CPD Distribuição de Probabilidade Condicional

DAHVI *Discriminative Approach for Hidden Variable Introduction*

EM *Expectation Maximization*

FORTE *First-order Revision of Theories from Examples*

ILP Programação em Lógica Indutiva

KBMC *Knowledge-Based Model Construction*

LL Log-verossimilhança

PFORTE *Probabilistic First-order Revision of Theories from Examples*

PFORTE PI *Probabilistic First-order Revision of Theories from Example with Predicate Invention*

PRM *Probabilistic Relational Model*

RB Rede Bayesiana

SEM *Structural Expectation Maximization*

SLP *Stochastic Logic Program*

SLR *Statistical Relational Learning*

TPPO Teoria Probabilística de Primeira-ordem

Lista de Algoritmos

Capítulo 1

Introdução

Nesta tese, nós combinamos técnicas de revisão de teoria com a extensão do vocabulário inicial com novas estruturas, definindo dois sistemas de revisão.

Aprendizado de máquina indutivo (MITCHELL, 1997) tem por objetivo o desenvolvimento de ferramentas e técnicas para induzir modelos [1] a partir de observações (exemplos) e sintetizar novos conhecimentos através de experiências. Diferentes mecanismos de representação dos modelos, como (i) rede Bayesiana, (ii) teoria de primeira-ordem e (iii) teoria probabilística de primeira-ordem (teoria de primeira-ordem com distribuição de probabilidade associada a cada uma das cláusulas), originam diferentes algoritmos de aprendizado. A maioria deles usa apenas o vocabulário fornecido nos dados para construir os modelos. Por exemplo, se o *dataset* apenas contém informações sobre o gosto dos usuários por filmes, então a única relação que será representada no modelo final, quando considerando lógica de primeira-ordem, é a relação $gosta(Usuario, Filme)$. No entanto, informações sobre os indivíduos podem enriquecer o processo de aprendizado, muitas vezes tornando o modelo aprendido mais eficiente e/ou compacto. Por exemplo, definir se um usuário x gosta de um filme y, ou se o estudante s vai passar no curso c, indica a importância de conhecimento individual, tais quais a idade de x, a atitude de s com relação aos seus estudos, a dificuldade do curso c ou o gênero de y. Na prática, entretanto, não é sempre que características como essas estão disponíveis no *dataset*. É interessante então, a utilização de técnicas para estender de forma automática o vocabulário inicial com novas estruturas que representem essas informações "escondidas", sendo estas consideradas durante o aprendizado dos modelos. Em redes Bayesianas essas estruturas

[1] Nesta tese, modelo denomina hipótese em aprendizado de máquina indutivo.

são conhecidas como *variáveis não-observadas* (FRIEDMAN, 1997), em lógica de primeira-ordem por *predicados inventados* (KRAMER, 1995)(STAHL, 1996) e em lógica de primeira-ordem probabilística vamos chamar de *predicados probabilísticos inventados*.

Geralmente, os sistemas de aprendizado indutivo aprendem os modelos partindo apenas do *dataset* e de um conhecimento preliminar invariante. Entretanto, em algumas situações, já existe um modelo inicial que é aproximadamente correto, ou seja apenas alguns pontos da sua estrutura o impede de refletir corretamente o *dataset*. Neste caso, é mais vantajoso considerar este modelo como ponto de partida para o aprendizado, encontrando pontos potenciais a sofrerem modificação, denominados *pontos de revisão*. Temos, então, um caso particular do aprendizado, que é a revisão do modelo a partir dos exemplos. A técnica de revisão utiliza exemplos classificados incorretamente para guiar a busca por *pontos de revisão*. *Operadores de revisão* são então utilizados para propor alterações a esses pontos, como inclusão ou exclusão de dependência entre as variáveis do modelo. A melhor proposta é então escolhida, de acordo com uma função de avaliação, para ser implementada. A revisão prossegue enquanto for possível melhorar o modelo corrente. Como técnicas de revisão definem *pontos de revisão*, o espaço de busca de modelos é menor, tornando o aprendizado mais rápido e o modelo final tão eficaz quanto o modelo aprendida a partir do zero.

Motivados, então, pela possibilidade de (i) aprender modelos melhores a partir da extensão do voabulário com novas estruturas e (ii) reduzir o espaço de busca a partir da utilização de técnicas de revisão de teoria, nós propomos nesta tese dois algoritmos de revisão de modelo baseados em invenção e introdução de novas estruturas ao vocabulário, tornando o modelo final mais eficaz e/ou compacto. O primeiro algoritmo revisa redes Bayesianas introduzindo *variáveis não-observadas* e o segundo revisa teorias probabilísticas de primeira-ordem introduzindo *predicados probabilísticos inventados*.

1.1 Revisando Redes Bayesianas através da Introdução de Variáveis Não-observadas

Redes Bayesianas (RB) são grafos direcionados, onde cada nó é uma variável aleatória. De forma a construir uma rede Bayesiana, é preciso definir que variáveis são de interesse, como elas são condicionalmente independentes, isto é, a estrutura da rede, e os parâmetros das distribuições de probabilidade para cada variável aleatória da rede. Vários algoritmos de aprendizado de redes Bayesianas a partir de *dataset*, já foram propostos na literatura (HECKERMAN, 1995). Esses algoritmos vão desde o caso mais simples, onde a estrutura da rede é conhecida e os dados observados, apenas precisando aprender os parâmetros da rede, até casos mais complexos, onde os dados são somente parcialmente observados, a estrutura pode ou não ser conhecida e variáveis aleatórias de interesse podem nunca ter sido observadas. Tais variáveis são conhecidas como *variáveis não-observadas*.

Variáveis não-observadas podem ser muito importantes tanto para redes Bayesianas dinâmicas quanto estáticas. No último caso, geralmente funcionam como mecanismos de agrupamento, que capturam informações a partir de um conjunto de variáveis observadas passando essas informações para outra parte da rede. Dessa forma, a introdução de *variáveis não-observadas* pode simplificar a estrutura da rede. Uma estrutura concisa permite inferência e aprendizado melhores, já que, reduz o número de parâmetros e ainda diminui a chance de *overfitting*. A Figura 1.1 exibe um exemplo motivador, originalmente descrito em (BINDER, KOLLER, et al., 1997), onde H é uma *variável não-observada*. Neste exemplo, introduzir H reduz o número de arestas na rede de 12 para 6, e, assumindo um domínio binário para cada variável aleatória, reduz o número de parâmetros probabilísticos em 61%. E o mais importante, retorna uma rede mais clara.

Infelizmente, aprender novas *variáveis não-observadas* em uma rede Bayesiana estática não é uma tarefa simples. Cada *variável não-observada* candidata precisa de um conjunto inicial de dependências e a definição de um domínio. Dessa forma, é preciso aprender uma nova estrutura de rede, que por sua vez precisa ter os seus parâmetros probabilísticos aprendidos. Este processo, pode ser repetido

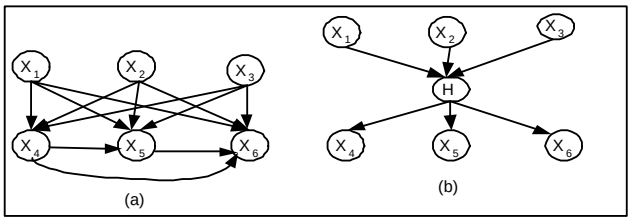


Figura 1.1: (b) Uma estrutura de rede Bayesiana com uma *variável não-observada* (H). (a) Uma estrutura de rede Bayesiana com a mesma distribuição sem a *variável não-observada*.

várias vezes, dependendo de quantas *variáveis não-observadas* deseja-se introduzir. Existem pesquisas para minimizar esse problema. O algoritmo *Structural Expectation Maximization* (SEM) (FRIEDMAN, 1998) combina aprendizado de estrutura com o algoritmo EM (MCLACHLAN, KRISHNAN, 1997) para redes Bayesianas, introduzindo *variáveis não-observadas* e aprendendo os parâmetros probabilísticos da estrutura para elas, mas exige que um conhecimento preliminar sobre a localização e número dessas *variáveis não-observadas* seja fornecido, para que ele consiga ter uma boa performance. Se não forem fornecidas tais informações, outras abordagens existem. O algoritmo *Semi-clique* (ELIDAN, LOTNER, et al., 2001), por exemplo, baseia-se na observação de que geralmente *variáveis não-observadas* podem simplificar segmentos da estrutura da rede que estão altamente conectados. Alternativamente, pode-se tirar vantagem da topologia da rede: BANNER (RAMACHANDRAN, MOONEY, 1998) introduz *variáveis não-observadas* em uma rede Bayesiana *noisy-or* e *noisy-and*.

Nesta tese, introduzimos uma nova abordagem para aprender *variáveis não-observadas* baseada em *revisão de teoria* (WROBEL, 1996). Nosso algoritmo de revisão, denominado **DAHVI** (**D***iscriminative* **A***pproach for* **H***idden* **V***ariable* **I***ntroduction*) (REVOREDO, PAES, et al., 2009), utiliza uma abordagem discriminativa: as *variáveis de consulta* e as suas *Coberturas de Markov* [2] correspondentes definem os *pontos de revisão*. Nos baseamos na observação de que geralmente queremos melhorar a performance para um conjunto de *variáveis de consulta*. Para

[2]Lembramos que a cobertura de Markov de um nó em uma rede Bayesiana consiste dos seus nós pais, dos seus nós filho e dos nós pai dos seus nós filho. A Cobertura de Markov considerando teorias probabilísticas de primeira-ordem será definida nesta tese.

isto, focamos nas suas *Coberturas de Markov*, já que as variáveis que influenciam o valor inferido para as *variáveis de consulta* pertencem a ela. Um *operador de revisão*, para especificamente introduzir *variáveis não-observadas*, é proposto nesta tese. Dessa forma, nosso algoritmo prossegue olhando para os exemplos classificados incorretamente, computando as *Coberturas de Markov* para as *variáveis de consulta*, propondo *variáveis não-observadas* e avaliando se elas melhoram a performance da rede ou não. Aplicamos o algoritmo DAHVI a 13 datasets, sendo 1 artificial e 12 reais, considerando 3 funções de avaliação. Os resultados mostraram os benefícios do nosso algoritmo, inclusive, demonstrando que (i) a nossa abordagem para selecionar *pontos de revisão* reduz o espaço de busca, e (ii) o nosso *operador de revisão* introduz *variáveis não-observadas* em lugares importantes da rede Bayesiana, indicando ser uma boa heurística para definição da localização das *variáveis não-observadas*.

1.2 Revisando Teorias Probabilísticas de Primeira-ordem através da Introdução de Predicados Probabilísticos Inventados

A possibilidade de representação de indivíduos, suas propriedades e as relações entre eles, fazem da lógica de primeira ordem um sistema de representação de conhecimento muito vantajoso, mas limitado por não representar incerteza. Vários formalismos têm sido propostos com a intenção de integrar estes sistemas com mecanismos de raciocínio probabilístico, definindo uma *teoria probabilística de primeira-ordem*. Alguns exemplos são: *Probabilistic Relational Model (PRM)* (KOLLER, 1999) (FRIEDMAN, GETOOR, et al., 1999), *Bayesian Logic Program (BLP)* (KERSTING, DE RAEDT, 2001c), *Constraint Logic Programming (CLP(BN))* (COSTA, PAGE, et al., 2003), *Stochastic Logic Program (SLP)* (MUGGLETON, 2002), *Markov Logic Network (MLN)* (RICHARDSON, DOMINGOS, 2006) e *Probabilistic Prolog (ProbLog)* (DE RAEDT, KIMMIG, et al., 2007) entre outros. A tarefa de aprender tais teorias é denominada com freqüência, Aprendizado Estatístico Relacional (*Statistical Relational Learning* (*SRL*)).

A maioria dos algoritmos de aprendizado propostos, com pequenas exceções como em (DE RAEDT, KERSTING, et al., 2008), onde um sistema de revisão para ex-

cluir cláusulas é definido na abordagem ProbLog, assumem que queremos aprender uma teoria do zero ou efetuam alterações em toda a estrutura sem beneficiar-se de uma heurística para redução do espaço de busca. Em (REVOREDO, ZAVERUCHA, 2002; PAES, REVOREDO, et al., 2005b; PAES, REVOREDO, et al., 2005a; PAES, REVOREDO, et al., 2006) nós definimos um sistema de revisão de teorias probabilísticas de primeira-ordem, denominado (**P***robabilistic* **F***irst* **O***rder* **R***evision of* **T***heories from* **E***xamples*). PFORTE revisa teorias probabilísticas de primeira-ordem, que como BLPs, são *cláusulas definidas* sem funções com algum *predicado probabilístico*, isto é, um predicado com um domínio associado, como antecedente e uma distribuição de probabilidade associada. PFORTE utiliza os exemplos classificados incorretamente, ou seja, exemplos que tiveram o valor do domínio inferido diferente do valor definido no exemplo, para determinar os *pontos de revisão*. Esses *pontos de revisão* são cláusulas cuja a cabeça ou é o *predicado objetivo* ou é um predicado pertencente a *Cobertura de Markov* do *predicado objetivo*. Dessa forma, a seleção dos *pontos de revisão* é feita através de uma abordagem discriminativa. Em seguida, *operadores de revisão*, como inclusão/exclusão de cláusulas e/ou antecedentes são aplicados, com o melhor, de acordo com uma função de avaliação probabilística, sendo escolhido e implementado.

Como a maioria dos algoritmos de aprendizado, o nosso sistema PFORTE apenas usa o vocabulário fornecido nos dados para revisar as teorias. Dessa forma, ele também pode ser beneficiado com a utilização de técnicas para estender de forma automática o vocabulário inicial, inventando novos predicados, seus domínios e distribuições de probabilidade associadas, definindo assim, *predicados probabilísticos inventados*.

Os mecanismos de invenção de predicados (KRAMER, 1995)(STAHL, 1996) em *Inductive Logic Programming* (ILP), (MUGGLETON, 1992) percorrem toda a estrutura da teoria para identificar aonde um *predicado inventado* poderia ser introduzido. Ao combinarmos invenção de predicados com técnicas de revisão de teoria, permitimos que a introdução do *predicado inventado* seja feita em pontos específicos da estrutura da teoria, ou seja, nos *pontos de revisão*, reduzindo assim o espaço de busca. Os sistemas KRT (WROBEL, 1994) e CWS (BAIN, MUGGLETON, 1992), por

exemplo, durante a especialização de uma cláusula, invés de adicionar antecedentes nesta, adicionam antecedentes na cláusula que define o *predicado inventado* e acrescentam esse ao corpo da cláusula sendo especializada, ou seja especializam através do *predicado inventado*. A diferença entre esses sistemas encontra-se na aridade do *predicado inventado*: KRT considera *predicados inventado* de aridade 1 e 2 enquanto que CWS considera *predicados inventado* com aridade igual ao número de variáveis da cláusula *ponto de revisão*.

Motivados pelos benefícios (i) da utilização de técnicas de revisão nos mecanismos de invenção de predicados e (ii) da associação de distribuições de probabilidade condicionais as cláusulas permitindo a representação de incerteza, propomos nesta tese combinar invenção de predicados com revisão de teorias de primeira-ordem probabilísticas, através da extensão do nosso sistema PFORTE, criando, por nós denominado, PFORTE_PI (**P**robabilistic **F**irst **O**rder **R**evision of **T**heories from **E**xamples with **P**redicate **I**nvention) (REVOREDO, PAES, et al., 2006; REVOREDO, PAES, et al., 2007). O sistema PFORTE_PI incorpora dois novos *operadores de revisão*, que utilizam técnicas de invenção de predicados para (i) propor *predicados probabilísticos inventados*, (ii) definir o seu domínio, (iii) aprender sua definição e (iv) distribuição de probabilidade condicional associada. Os resultados experimentais obtidos com a aplicação do PFORTE_PI em 2 *datasets* reais comprovam os benefícios da proposta.

Invenção de predicados probabilísticos no PFORTE_PI é uma generalização da invenção de *variáveis não-observadas* no sistema DAHVI.

Alguns trabalhos da literatura, como o sistema SAYU-VISTA (DAVIS, ONG, et al., 2007) e o sistema MRC (KOK, DOMINGOS, 2007) consideram invenção de predicados em SRL. A principal diferença desses sistemas para o sistema PFORTE_PI, é o fato deles não considerarem uma heurística para reduzir o espaço de busca e além disso, apesar de serem algoritmos de SRL, aprenderem a estrutura da teoria a partir de um algoritmo lógico: Aleph (SRINIVASAN, 2001) no caso do SAYU-VISTA e CLAUDIEN (DE RAEDT, DEHASPE, 1997) no caso do MRC.

1.3 Objetivos e Organização

As contribuições desta tese são: (i) definição de um algoritmo de revisão de redes Bayesianas que introduz *variáveis não-observadas*, denominado DAHVI; (ii) definição de uma algoritmo de revisão de teorias probabilísticas de primeira-ordem, que introduz *predicados probabilísticos inventados*, denominado PFORTE_PI; (iii) definição de um procedimento discriminativo para seleção dos *pontos de revisão*: as *variáveis de consulta* (quaisquer variáveis aleatórias em uma rede Bayesiana ou *predicados objetivos* em uma teoria probabilística de primeira-ordem) e suas *Coberturas de Markov* definem pontos potenciais à modificações; (iv) definição de um *operador de revisão*, que introduz *variáveis não-observadas*; (v) definição de dois *operadores de revisão*, que introduzem *predicados probabilísticos inventados*. Além dessas contribuições relacionadas a extensão do vocabulário inicial com novas estruturas, também é contribuição desta tese a utilização de técnicas de revisão de teoria na linguagem *Probabilistic Prolog (ProbLog)* (DE RAEDT, KERSTING, et al., 2008).

Esta documentação foi organizada da seguinte forma. No Capítulo 1, introduzimos o nosso trabalho. Nos Capítulos 2, 3 e 4 fazemos uma revisão bibliográfica de conceitos importantes para o entendimento das nossas propostas, como conceitos: (i) probabilísticos (Capítulo 2): aprendizado de redes Bayesianas e funções de avaliação probabilísticas; (ii) lógicos (Capítulo 3): revisão de teorias de primeira-ordem e invenção de predicados; (iii) lógico probabilístico (Capítulo 4): BLPs e o nosso sistema PFORTE. Em seguida apresentamos os nossos sistemas. O sistema DAHVI no Capítulo 5, com os resultados experimentais obtidos com a aplicação dele em diferentes *datasets* no Capítulo 6, e o sistema PFORTE_PI no Capítulo 7, com os resultados correspondentes no Capítulo 8. No Capítulo 7 também é apresentado o sistema de revisão de ProbLogs. Finalizando, no Capítulo 9, concluímos o nosso trabalho e apresentamos alguns possíveis caminhos para novas pesquisa.

Capítulo 2

Revisão de Modelos Probabilísticos

Neste capítulo, discutimos alguns conceitos importantes para revisão de modelos probabilísticos como: conceitos de probabilidade (seção 2.1); redes Bayesianas (seção 2.2) e algumas funções de avaliação probabilísticas (seção 2.3). O leitor familiarizado com a Teoria das probabilidades, pode seguir para a seção 2.2.

2.1 Probabilidade

Um problema com a lógica de primeira ordem, e conseqüentemente com as abordagens dos agentes lógicos, é que os agentes quase sempre não têm acesso a toda a verdade sobre os assuntos que eles estão discutindo. Algumas sentenças podem ser diretamente verificadas através da percepção do agente, outras podem ser inferidas de uma percepção anterior, ou atual junto com um conhecimento sobre as propriedades do ambiente. Na maioria dos casos, entretanto, mesmo nos mais simples, existem importantes questões para as quais os agentes não encontram uma resposta categórica, logo terão de agir com um grau de incerteza. Por exemplo, suponha que um determinado agente gostaria de conduzir uma pessoa que deseja viajar até o aeroporto. Mesmo que o agente se programe para sair de casa noventa minutos antes da partida do avião e leve em consideração que o aeroporto localiza-se a 30km da sua casa, ele não poderá garantir que chegará ao aeroporto em tempo hábil, pois um dos pneus do carro poderá furar ou o próprio carro poderá ter algum tipo de pane, o trânsito poderá estar extremamente lento, em decorrência a gasolina poderá acabar, enfim, vários imprevistos poderão ocorrer em somatório ou não.

Logo, em um caso como este, o que o agente poderia concluir, é que ele pode chegar ao aeroporto no horário previsto com uma chance de, por exemplo, 0.9 considerando que tudo sairá como desejado.

Uma outra questão é o fato de muitas regras sobre o domínio estarem incompletas, ou porque existem muitas situações a serem enumeradas explicitamente ou porque algumas das condições são desconhecidas. Com isso, não pode-se ter total certeza do que é inferido a partir delas. Nesses casos o que o agente pode fornecer é um grau de crença na situação em questão. A principal ferramenta para lidar com incertezas é a teoria de probabilidades, que associa a cada sentença um grau de crença numérico entre 0 e 1.

Um elemento básico pertencente à linguagem que compõe a teoria de probabilidades é a variável aleatória, que representa a parte da teoria onde o valor assumido não é conhecido. No exemplo anterior, a variável aleatória estaria representando a chegada do agente ao aeroporto a tempo do vôo ou não. Cada variável aleatória tem um domínio de valores possíveis que ela pode assumir. No exemplo acima o domínio seria composto pelos valores verdadeiro, para o caso do agente chegar ao aeroporto a tempo do vôo e falso, caso contrário. As variáveis aleatórias não assumem somente valores binários, elas também podem ser discretas, assumindo valores de um domínio finito, ou contínuas, onde o seu domínio é formado por números reais.

Uma noção que será útil futuramente é a de eventos atômicos. Evento atômico (interpretação) é uma completa especificação de um estado do mundo sobre o qual o agente está em dúvida, ou seja é a associação de valores para todas as variáveis aleatórias do problema que está sendo abordado. Por exemplo, se considerássemos que o nosso mundo é composto apenas pelas variáveis Chegar_ao_Aeroporto e Tráfego, então a proposição Chegar_ao_Aeroporto = verdadeiro $\wedge$ Tráfego = bom é um evento atômico. Os eventos atômicos são exclusivos, no máximo um é o realmente procurado.

A probabilidade a priori de uma proposição a é o grau de crença nesta na ausência de qualquer outra informação. Por exemplo, se a probabilidade a priori do tráfego estar bom é de 0.5, escreveríamos:

P(Tráfego = bom) = 0.5

Uma distribuição de probabilidade seria a definição de probabilidades para todos os possíveis valores de uma determinada variável. Por exemplo, a distribuição de probabilidade da variável Tráfego, P(Tráfego), pode ser definida como:

P(Tráfego= muito_bom) = 0.2

P(Tráfego = bom) = 0.5

P(Tráfego= ruim) = 0.3

ou simplesmente P(Tráfego) = $< 0.2, 0.5, 0.3 >$

Como notação, consideraremos:

P(X = x) ou P(x) para denotar a probabilidade da variável X assumir o valor x e P(X) para denotar a distribuição de probabilidade sobre X.

Para variáveis contínuas, não é possível escrever toda a distribuição como uma tabela, porque existem infinitos valores. Utiliza-se então a função de densidade de probabilidade, a qual indica a probabilidade da variável aleatória pertencer a um determinado intervalo de números reais. Como exemplo temos a função de distribuição Normal, (DEGROOT, 1987), $N(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(\frac{(x-\mu)^2}{\sigma})}$, onde μ é a média e σ a variância. Nesta tese apenas utilizaremos variáveis aleatórias discretas, logo a notação P(X) apenas denotará distribuições de probabilidade, não sendo utilizada para densidades de probabilidade.

Os axiomas 1, 2, 3a ou 3b definidos abaixo são suficientes como fundamentos da teoria de probabilidade.

1. Para qualquer proposição $\underline{a}$, $0 \leq P(a) \leq 1$;

2. Necessariamente proposições verdadeiras têm probabilidade 1, enquanto que proposições falsas têm probabilidade 0;

3. a) $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$

   b) $P(a) = \sum_{e_i \in e(a)} P(e_i)$ onde e(a) são todos os eventos atômicos onde $\underline{a}$ acontece.

A partir dos axiomas 1,2 e 3a é possível mostrar que para qualquer variável aleatória X,

$$\sum_{i=1}^{n} P(X = x_i) = 1$$

onde o domínio de X é dom(X) = $< x_1, ..., x_n >$. Ou seja, qualquer distribuição de probabilidades de uma variável (discreta) deve somar 1.

Algumas vezes estamos interessados na probabilidade de todas as combinações de valores de um conjunto de variáveis aleatórias. Considere a expressão P(Tráfego,Chegar_ao_Aeroporto). Ela significa que estamos interessados em todas as combinações de possíveis valores para as variáveis Trafego e Chegar_ao_Aeroporto. Como a variável Tráfego tem um domínio composto por 3 valores e a variável Chegar_ao_Aeroporto um domínio binário, especificamos 6 valores combinados e a representação desses valores pode ser feita através de uma tabela 3 por 2. Esta distribuição é conhecida como distribuição de probabilidade conjunta das variáveis em questão. Quando consideramos todas as variáveis do problema em questão estamos olhando para a distribuição de probabilidade conjunta total, esta especifica a probabilidade de todos os eventos atômicos.

A partir do momento em que o agente obtiver alguma evidência sobre as variáveis aleatórias, a probabilidade a priori não pode mais ser aplicada, utiliza-se então a probabilidade condicional ou posterior. A notação utilizada é P(a|b) que significa a probabilidade de a dado que tudo que sabemos é b. Por exemplo, P(Chegar_ao_Aeroporto = verdadeiro| Tráfego = muito_bom) = 0.8 indica que se o tráfego estiver muito bom e nenhuma outra informação for fornecida então a probabilidade de uma pessoa chegar ao aeroporto a tempo do vôo é de 0.8.

Probabilidades condicionais podem ser definidas em termos de probabilidades a priori:

$$P(a|b) = \frac{P(a \wedge b)}{P(b)}$$

Escrevendo de outra maneira

$$P(a \wedge b) = P(a|b)P(b)$$

que é chamada de regra do produto.

Técnicas como a consideração de exemplos virtuais (MITCHELL, 1997) podem ser utilizadas para evitar que $P(b) = 0$.

Generalizando, suponha que tenhamos as seguintes variáveis aleatórias $\{X_1, ..., X_n\}$. Pela regra do produto

$$
\begin{aligned}
P(x_1, ..., x_n) &= P(x_n|x_{n-1}, ..., x_1)P(x_{n-1}, ..., x_1) \\
&= P(x_n|x_{n-1}, ..., x_1)P(x_{n-1}|x_{n-2}, ..., x_1)...P(x_2|x_1)P(x_1) \\
&= \prod_{i=1}^{n} P(x_i|x_{i-1}, ..., x_1) \qquad (1)
\end{aligned}
$$

Esta identidade é válida para qualquer conjunto de variáveis aleatórias e é chamada de regra da cadeia.

Um teorema importante na teoria de probabilidades é o teorema de Bayes definido como:

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

O teorema de Bayes é útil na prática, porque existem muitos casos onde temos boas estimativas de probabilidade para os três termos da direita e precisamos computar o da esquerda.

Para o caso mais geral, o teorema de Bayes pode ser escrito utilizando a notação de distribuição de probabilidades.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

2.2 Redes Bayesianas

Uma rede Bayesiana é uma maneira compacta de representar conhecimento probabilístico.

A idéia é representar um domínio em termos de variáveis aleatórias e explicitamente modelar a interdependência das variáveis aleatórias em termos de um grafo. Dizemos então que uma rede Bayesiana é um grafo direcionado no qual as variáveis aleatórias são os nós da rede e os arcos da rede representam dependência direta entre estas variáveis aleatórias.

Associado a cada nó está uma tabela de probabilidades condicionais, que fornece a probabilidade de cada valor da variável dado cada combinação possível de valores para cada um dos seus predecessores imediatos[1] (Pais), em outras palavras é a distribuição de probabilidade condicional (CPD) de uma determinada variável aleatória

[1]Dizemos que Y é predecessor de X se existe um caminho direto de Y até X. E dizemos que Y é predecessor imediato de X se Y for predecessor de X e não existir nenhuma outra variável entre os dois.

dados os seus pais. Podemos dizer que, a CPD define como os pais interagem entre si produzindo uma influência conjunta sobre o seu nó filho.

Uma definição importante em rede Bayesiana é a de independência condicional. Dizemos que X é condicionalmente independente de Y dado Z se a distribuição de probabilidade governando X é independente do valor de Y dado o valor de Z ou seja se

$$(\forall x_i, y_j, z_k)P(X = x_i|Y = y_j, Z = z_k) = P(X = x_i|Z = z_k) \tag{2.1}$$

onde x_i , y_j , z_k pertencem ao domínio das variáveis X, Y, Z respectivamente. Geralmente a fórmula acima é escrita na forma abreviada P(X|Y,Z) = P(X|Z). A definição de condicionalmente independente pode ser estendida para conjuntos de variáveis da mesma maneira.

$$P(X_1, ..., X_l|Y_1, ..., Y_m, Z_1, ..., Z_n) = P(X_1, ..., X_l|Z_1, ..., Z_n) \tag{2.2}$$

Por construção supõe-se que esta propriedade acontece em redes Bayesianas, com as suas variáveis sendo condicionalmente independentes dos seus não-descendentes na rede, dado seus predecessores imediatos. Com isso a probabilidade conjunta para qualquer associação de valores para as variáveis da rede, isto é, $P(x_1, ..., x_n)$, pode ser calculada utilizando a seguinte fórmula:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|Pais(X_i)) \tag{2.3}$$

onde $P(x_i|Pais(X_i))$ é obtida da CPD associada a variável X_i, o que reduz a computação necessária para a definição da probabilidade conjunta das variáveis aleatórias da rede.

Comparando as equações 2.2 e 2.3, verifica-se que para toda variável X_i, na rede

$$P(X_i|X_{i-1}, ..., X_1) = P(X_i|Pais(X_i)) \tag{2.4}$$

onde $Pais(X_i) \subseteq \{X_{i-1}, ..., X_1\}$. Esta condição pode ser satisfeita se ordenarmos os nós de uma maneira consistente com a ordenação implícita da estrutura da rede Bayesiana.

A Figura 2.1 mostra um exemplo de rede Bayesiana, (RUSSELL, NORVIG, 2002). A situação considerada é a de um sistema de alarme que pode ser ativado

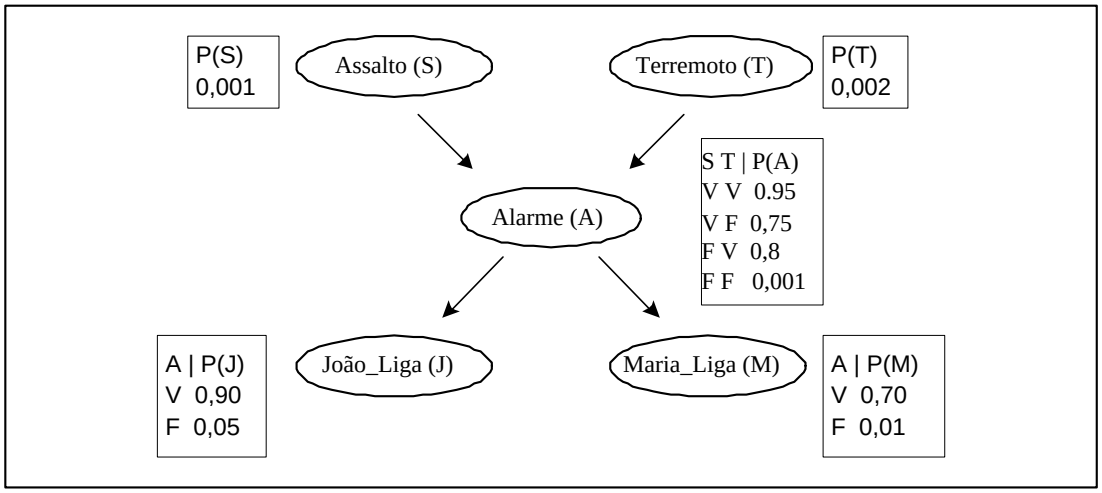


Figura 2.1: Rede Bayesiana para o sistema de alarme

tanto por um assaltante quanto por um terremoto, este último em casos mais extremos. E dado que o alarme foi tocado espera-se que Maria e/ou João liguem para a polícia. Só que às vezes, estas duas pessoas confundem o barulho do alarme com outros como, por exemplo, o toque do telefone, com isto existe uma incerteza quanto as ligações.

A rede indica que a decisão de Maria ligar para polícia independe da de João. Já o disparo do alarme está diretamente ligada ao fato de ter ocorrido um assalto ou um terremoto.

Dadas as CPDs, a distribuição de probabilidade conjunta total da rede pode ser computada como mostrado abaixo:

$$P(S, T, A, J, M) = P(S)P(T)P(A|S, T)P(J|A)P(M|A)$$

Como exemplo vamos supor que gostaríamos de calcular a probabilidade do evento: o alarme tocou, mas nem um assalto nem um terremoto ocorreram, e ambos João e Maria ligaram. Considere que não existe ruído ou falha no equipamento. Utilizaremos letras minúsculas para indicar o valor que cada uma das variáveis aleatórias esta assumindo e $\neg$ para indicar a negação, logo:

$$\begin{aligned} P(\neg s \wedge \neg t \wedge a \wedge j \wedge m) &= P(\neg s)P(\neg t)P(a|\neg s, \neg t)P(j|a)P(m|a) \\ &= 0.999 * 0.998 * 0.001 * 0.90 * 0.70 \\ &= 0.00062 \end{aligned}$$

Relacionamentos com incerteza podem freqüentemente ser caracterizados por

relações *noisy*. O exemplo padrão é a relação *noisy-or*, uma generalização do *ou* lógico. Voltando à rede Bayesiana definida na Figura 2.1. Podemos dizer que o alarme pode ser disparado se e somente se ocorrer um terremoto ou um assalto. O modelo *noisy-or* permite associar uma incerteza sobre a habilidade que cada pai tem de influenciar o filho a ser verdadeiro- a relação causal entre os pais e o filho pode ser inibida. O modelo faz duas suposições. Primeiro, assume que todas as possíveis causas são listadas. Segundo, assume que a inibição de cada pai é independente da inibição de qualquer outro pai - por exemplo , qualquer que seja o motivo pelo qual um assalto não acionou o alarme independe do motivo pelo qual um terremoto também não acionou o alarme. A partir destas suposições, o alarme não disparará se e somente se todos os seus pais que são verdadeiros forem inibidos, e a probabilidade disto é o produtório das probabilidade de inibição de cada pai. Suponha as probabilidades inibidoras individuais mostradas abaixo.

$P(Alarme = F|Assalto = V, Terremoto = F) = 0.25$

$P(Alarme = F|Assalto = F, Terremoto = V) = 0.2$

A partir destas probabilidades a CPD do predicado alarme pode ser computada como mostrado na tabela 2.1.

Assalto	Terremoto	P(Alarme=V)	P(Alarme=F)
F	F	0	1
F	V	0.8	0.2
V	F	0.75	0.25
V	V	0.95	$0.2 * 0.25 = 0.05$

Tabela 2.1: CPD do predicado Alarme utilizando *noisy-or*.

Em geral, um relacionamento lógico *noisy* na qual a variável depende de k pais pode descrever sua distribuição de probabilidades condicionais utilizando O(k) parâmetros invés de $O(2^k)$. Isto torna o acesso e o aprendizado muito mais fácil.

2.2.1 Inferência

Considere a rede Bayesiana da Figura 2.1. Suponha que tenha ocorrido um assalto, ou seja que o valor da variável assalto é verdadeiro, e que queremos saber a probabilidade de Maria ligar, ou seja $P(Maria\_Liga = V|Assalto = V)$. Como nesse caso, o valor das variáveis Terremoto e Alarme é desconhecido, inferência é

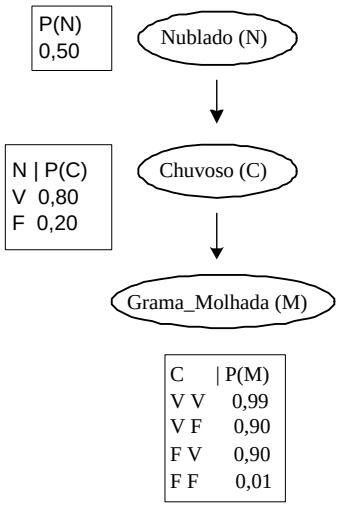


Figura 2.2: Rede *PolyTree*

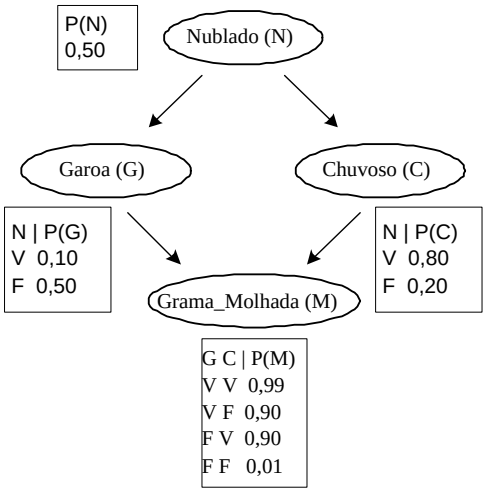


Figura 2.3: Rede com Ciclo

necessária. A variável Maria_Liga é chamada de *variável de consulta* enquanto a variável Assalto é uma *evidência*.

Generalizando, uma rede Bayesiana pode ser utilizada para inferir o valor da variável de consulta dados valores observados para o conjunto de evidências. Já que estamos lidando com variáveis aleatórias não seria muito correto associar à variável de consulta um determinado valor, o que realmente estamos querendo inferir é a distribuição de probabilidade para esta variável. De um modo geral, redes Bayesianas são utilizadas para inferir a distribuição de probabilidade de um conjunto de variáveis de consulta, dado um conjunto de evidências. Redes Bayesianas são flexíveis o suficiente para permitir que qualquer conjunto de nós possa servir tanto como um conjunto de variáveis de consulta como de evidência.

A estrutura da rede Bayesiana tem influência significativa na complexidade da inferência. Baseadas na sua estrutura, redes Bayesianas podem ser divididas em duas classes: *polytrees*, que têm uma estrutura simples onde cada par de variáveis está conectada por até um caminho (Figura 2.2), e redes com ciclos, que tem ciclos não-direcionados na estrutura (Figura 2.3).

Podemos dividir os métodos de inferência em dois grandes grupos: os métodos de inferência exata e os de inferência aproximada.

Inferência exata de probabilidades, em geral para uma rede Bayesiana arbitrária é NP-difícil (COOPER, 1990). Os métodos de inferência aproximada, sacrificam pre-

cisão para ganhar em eficiência. Por exemplo, o método de Monte Carlo que fornece uma solução aproximada para aleatoriamente achar amostras das distribuições das variáveis que não são observadas, ou seja que não são evidência (PRADHAM, DAGUM, 1996). Na teoria, até estes podem ser NP-difícil (DAGUM, LUBY, 1993). Na prática métodos de inferência aproximada mostram-se úteis em muitos casos.

Como utilizamos um método de inferência exata nos nossos algoritmos é este grupo que abordaremos a seguir.

Inferência Exata

Um processo simples de inferência exata é a inferência por enumeração, onde todas as variáveis do problema em questão, que não estão sendo observadas são somadas.

$$P(Q|\mathbf{e}) = \frac{P(Q,\mathbf{e})}{P(\mathbf{e})} = \alpha P(Q,\mathbf{e}) = \alpha \sum_{y} P(Q,\mathbf{e},y)$$

onde Q representa a variável de consulta, $\mathbf{e}$ o conjunto de evidências e Y as variáveis que não tiveram o seu valor observado. Como P($\mathbf{e}$) é invariante para cada valor de Q considerado, podemos substitui-lo por uma constante, α, que chamaremos de constante de normalização. Ela assegura que a soma das probabilidades para cada um dos valores de Q dará 1.

Supondo a rede Bayesiana da Figura 2.1, vamos considerar que estamos em busca da P(Assalto | Joao_Liga = verdadeiro , Maria_Liga = verdadeiro). Neste caso, as variáveis que não tiveram o seu valor observado são Terremoto e Alarme, temos então:

$$P(S|j,m) = \alpha P(S,j,m) = \alpha \sum_{t} \sum_{a} P(S,t,a,j,m).$$

Supondo Assalto = verdadeiro a expressão fica:

$$P(s|j,m) = \alpha P(s,j,m) = \alpha \sum_{t} \sum_{a} P(s)P(t)P(a|s,t)P(j|a)P(m|a).$$

Para computar esta expressão, temos que somar quatro termos (um para cada combinação de valores para as variáveis Terremoto e Alarme: VV,VF,FV,FF), cada um computando o produto de cinco probabilidades. No pior caso, onde temos que somar quase todas as variáveis da rede, a complexidade do algoritmo para uma rede com n variáveis binárias é de $O(n2^n)$. Fazendo uma reorganização da expressão

acima, percebendo quais das variáveis não são influenciadas pelo somatório, como mostra a fórmula abaixo, podemos reduzir a complexidade para $O(2^n)$.

$$P(s|j,m) = \alpha P(s,j,m) = \alpha P(s) \sum_t P(t) \sum_a P(a|s,t) P(j|a) P(m|a).$$

No exemplo acima o produto entre $P(j|a)$ e $P(m|a)$ é feito duas vezes, uma para t e outra para $\neg t$, o que tratando-se de um domínio com muitas variáveis pode tornar-se computacionalmente inviável. Um outro método de inferência exata é a eliminação de variável, que se mostra mais eficiente do que o por enumeração, por exatamente evitar estas computações repetidas.

A idéia é fazer a computação apenas uma vez e guardar o resultado para usos futuros. O algoritmo analisa a equação utilizada no algoritmo de enumeração da direita para esquerda, os resultados intermediários são armazenados e os somatórios sobre variáveis são feitos somente para as partes da expressão que dependem desta variável.

Voltando ao exemplo do assalto:

$$P(s|j,m) = \alpha P(s,j,m) = \alpha P(s) \sum_t P(t) \sum_a P(a|s,t) P(j|a) P(m|a)$$

para cada parte da expressão associamos um nome, que na maioria dos casos é a inicial do nome da variável em questão, e são denominados fatores.

O primeiro fator então, seria o fator M que estaria representando a P(m|a). Os passos então são os seguintes.

- O fator para M não necessita de um somatório sobre este, pois seu valor é fixo. Simplesmente guardamos a probabilidade dada para cada valor de $\underline{a}$ em um vetor de dois elementos, que chamaremos de $f_M(A)$:

$$f_M(A) = (P(m|a), P(m|\neg a)).$$

- Similarmente o fator para J, $P(j|a)$, é guardado como um vetor de dois elementos, $f_J(A) = (P(j|a), P(j|\neg a))$.

- O fator para A é P(a|s,t), que será uma matriz 2 X 2 X 2, $f_A(A,S,T)$.

- Agora soma-se para A o produto desses três fatores.

$$f_{AJM}(S,T) = \sum_a f_A(A,S,T) * f_J(A) * f_M(A)$$
$$= f_A(a,S,T) * f_J(a) * f_M(a) + f_A(\neg a,S,T) * f_J(\neg a) * f_M(\neg a).$$

- Agora processa-se T da mesma maneira: somatório de T para o produto de $f_T(T)$ e $f_{AJM}(S,T)$,

$$f_{TAJM}(S) = f_T(t) * f_{AJM}(S,t) + f_T(\neg t) * f_{AJM}(S,\neg t).$$

- Podemos computar o resultado simplesmente multiplicando o fator para S, isto é $f_S(S) = P(S)$ por $f_{TAJM}(S)$:

$$P(S|j,m) = \alpha f_S(S) * f_{TAJM}(S).$$

Note que o problema anteriormente mencionado da computação duplicada do produtório entre $P(j|a)$ e $P(m|a)$, foi resolvido, pois o fator $f_{AJM}(S,T)$ armazenou este cálculo e quando deseja-se saber o valor para t e $\neg t$ apenas busca-se o valor correspondente.

O algoritmo apresentado acima é simples e eficiente para responder consultas individuais. Se quisermos computar a probabilidade posterior para todas as variáveis na rede, entretanto, pode ser menos eficiente. Por exemplo, em uma rede *polytree* seriam necessárias O(n) consultas com um custo de O(n) para cada uma, com um total de tempo $O(n^2)$.

Algoritmos de agrupamento (também conhecidos como algoritmos *join tree* (*junction tree*)(JENSEN, 1996), (COWELL, DAWID, et al., 1999)), podem reduzir o custo para O(n) em algumas situações fazendo com que sejam muito utilizados em ferramentas comerciais de redes Bayesianas.

A idéia básica do algoritmo de agrupamento é agrupar nós individuais da rede para formar um agrupamento de nós de tal maneira que a rede resultante seja uma *polytree*. Por exemplo, a rede mostrada na Figura 2.3 pode ser convertida em uma *polytree* através da combinação dos nós Garoa e Chuvoso em um nó agrupado chamado Garoa+Chuvoso, como mostrado na Figura 2.4. Os dois nós binários são substituídos por um mega-nó que assume os quatro possíveis valores: VV, VF, FV e

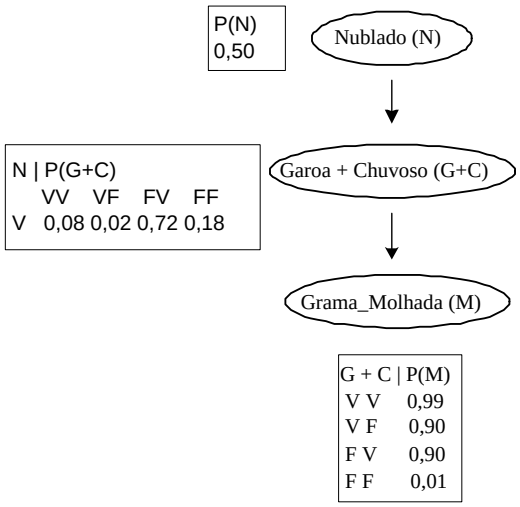


Figura 2.4: Rede *PolyTree* encontrada pelo algoritmo de agrupamento

FF. O mega-nó tem apenas um pai, a variável binária Nublado, então existem dois casos condicionais.

A partir do momento que a rede tem a estrutura de uma *polytree*, um algoritmo de inferência especial é aplicado. Essencialmente, o algoritmo é uma forma de propagação restrita onde as restrições garantem que os grupos vizinhos concordem na probabilidade posterior de qualquer uma das variáveis que eles tem em comum. Pode-se dizer com uma certa cautela, que este algoritmo é capaz de computar probabilidades posteriores para todas as variáveis não evidência da rede em um tempo O(n), onde n agora é o tamanho da rede modificada. Entretanto, se a rede precisar de tempo exponencial para a inferência com o algoritmo de eliminação de variável, então será preciso tempo exponencial para construir as CPDs da rede agrupada.

Maiores detalhes sobre métodos de inferência para redes Bayesianas são fornecidos em (RUSSELL, NORVIG, 2002) e (JENSEN, 1996).

2.2.2 Aprendizado das CPDs

Uma importante questão quando trabalhando com redes Bayesianas é a determinação das distribuições de probabilidade condicionais (CPDs) associadas a cada uma das variáveis aleatórias da rede. Nesta seção nós mostramos como aprender as CPDs a partir de um conjunto de exemplos.

Quando estamos lidando com um domínio sendo representado por uma rede

Bayesiana, onde esta é fixa, e o conjunto de exemplos de treinamento é completo, ou seja, o valor de todas as variáveis aleatórias do domínio são especificados em todos os exemplos tornando todas as variáveis em evidência em todos os exemplos, nós podemos simplesmente utilizar o método da contagem para determinar a distribuição de probabilidade destas variáveis. Suponha o interesse na probabilidade P(X=x | Pa=pa). Para determinar esta probabilidade, basta contar o número de exemplos onde as variáveis X e Pa assumem os valores x e pa respectivamente e dividir pelo número de exemplos onde a variável Pa assume o valor pa:

$$P(X = x|Pa = pa) = \frac{N(X = x, Pa = pa)}{N(Pa = pa)}$$

Agora, se o *dataset* não for completo, esta contagem não pode mais ser efetuada. Existem algoritmos para lidar com esta questão, como o gradiente, (MITCHELL, 1997), (BINDER, KOLLER, et al., 1997) e o *Expectation Maximization* (EM) (DEMPSTER, LAIRD, et al., 1977),(MCLACHLAN, KRISHNAN, 1997), (LAURITZEN, 1995). Como o segundo foi o por nós utilizado em trabalhos anteriores e é o que pretendemos utilizar na nossa proposta de trabalho detalharemos como ele funciona nos nossos algoritmos na próxima seção.

Expectation Maximization **(EM)**

O EM é um método proposto para o aprendizado das CPDs (θ) quando o conjunto de exemplos de treinamento ($\mathbf{C}$) é incompleto. Ele começa com CPDs iniciais e calcula a verossimilhança ($L(\theta : \mathbf{C})$, será definida na seção 2.3) dos exemplos utilizando estas CPDs. A cada iteração tenta encontrar novas CPDs (θ^*) que reflitam melhor o conjunto de exemplos de treinamento, ou seja tem por objetivo maximizar a verossimilhança.

$$\theta^* = \max_{\theta \in \mathbf{\Theta}} L(\theta : \mathbf{C}) = \max_{\theta \in \mathbf{\Theta}} P(\mathbf{C}|\theta, H) \tag{2.5}$$

onde H representa a rede Bayesiana para a qual estamos aprendendo as CPDs. Intuitivamente, ele fornece uma maneira de completar a contagem usando as CPDs correntes.

Considere por exemplo a rede Bayesiana exibida na Figura 2.5. Nela podemos ver as contagens que precisam ser calculadas para a definição das CPDs. Quando o

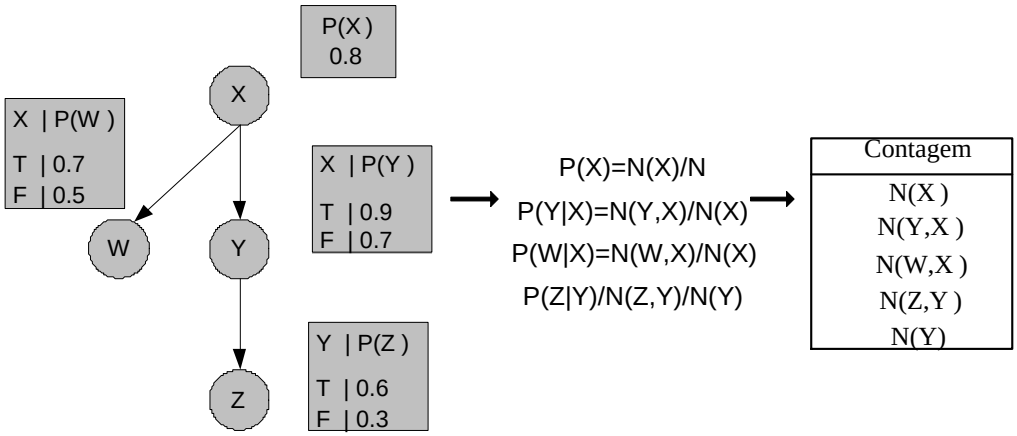


Figura 2.5: Indica as contagens necessárias para determinar as distribuições de probabilidade para a rede Bayesiana exibida nesta Figura.

dataset não contem informação a respeito de alguma variável, a contagem respectiva precisa ser estimada. O primeiro passo do algoritmo EM, que é o da Estimação (*Expectation*) (passo-E), tem por objetivo calcular essas contagens estimadas.

Suponha que no exemplo da Figura 2.5, as quatro variáveis {X, Y, Z, W}, tenham domínio binário, dom(X) = dom(Y) = dom(Z) = dom(W) = {F,T} e que desejamos estimar a contagem $N(Y = T, X = T)$ referente à probabilidade $P(Y = T|X = T)$ da CPD do nó Y. Para isto vamos considerar o conjunto de dados exibidos na Figura 2.6.

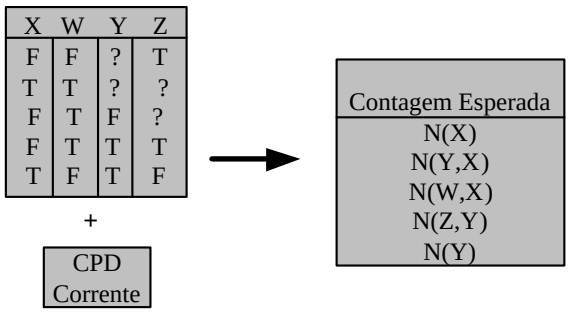


Figura 2.6: Dado o *dataset* parcialmente observado, utiliza-se o passo-E do algoritmo EM para determinar as contagens esperadas.

$$N(Y = T, X = T) = \sum_C P(Y = T, X = T|e) =$$
$$P(Y = T, X = T|X = F, W = F, Z = T)+$$
$$P(Y = T, X = T|X = T, W = T)+$$
$$P(Y = T, X = T|X = F, W = T, Y = F)+$$

$$P(Y = T, X = T|X = F, W = T, Y = T, Z = T)+$$
$$P(Y = T, X = T|X = T, W = F, Y = T, Z = F)$$
$$= 0+? + 0 + 0 + 1 =?$$

Como pode ser observado, o segundo exemplo não tem informação a respeito de Y, logo precisamos inferir $P(Y = T, X = T|X = T, W = T)$ considerando as CPDs correntes, exibidas na Figura 2.5, e as evidências definidas neste exemplo. Supondo a inferência por enumeração o calculo da probabilidade ficará:

$$P(Y = T, X = T|X = T, W = T) = P(Y = T|X = T, W = T) =$$
$$\alpha P(Y = T, X = T, W = T) =$$
$$\alpha \sum_Z P(Y = T, X = T, W = T, Z) =$$
$$\alpha P(X = T)P(W = T|X = T)P(Y = T|X = T) \sum_Z P(Z|Y = T) =$$
$$\alpha[0.8 * 0.7 * 0.9(0.6 + 0.4)] = \alpha 0.504$$

onde α é a constante de normalização. Normalizando temos que a probabilidade desejada é 0.9. Retornando à contagem esperada temos: $N(Y = T, X = T) = 0+? + 0 + 0 + 1 = 0 + 0.9 + 0 + 0 + 1 = 1.9$

Após calcular todas as contagens esperadas relevantes para o problema em consideração, o passo de Maximização, (*Maximization*) (passo-M) será efetuado. Este passo vai utilizar estas contagens, para encontrar os novas CPDs e calcular a nova verossimilhança.

O EM garante que a verossimilhança da iteração anterior é necessariamente menor do que a do passo corrente. Em outras palavras:

$$L(\theta_\mathbf{1}|\mathbf{C}) \geq L(\theta_\mathbf{0}|\mathbf{C})$$

onde θ_1 são as CPDs corrente e θ_0 as CPDs da iteração anterior.

Quando $L(\theta_\mathbf{1}|\mathbf{C}) - L(\theta_\mathbf{0}|\mathbf{C}) \leq \xi$, onde ξ é um erro considerado, podemos parar a computação.

A Figura 2.7 ilustra melhor o que acabamos de discutir, onde $L_{atual} = L(\theta_\mathbf{1}|\mathbf{C})$, $L_{anterior} = L(\theta_\mathbf{0}|\mathbf{C})$ e $erro = \xi$.

Este algoritmo efetua uma busca gulosa na superfície da verossimilhança convergindo para um ponto estacionário local (geralmente um máximo local). Infeliz-

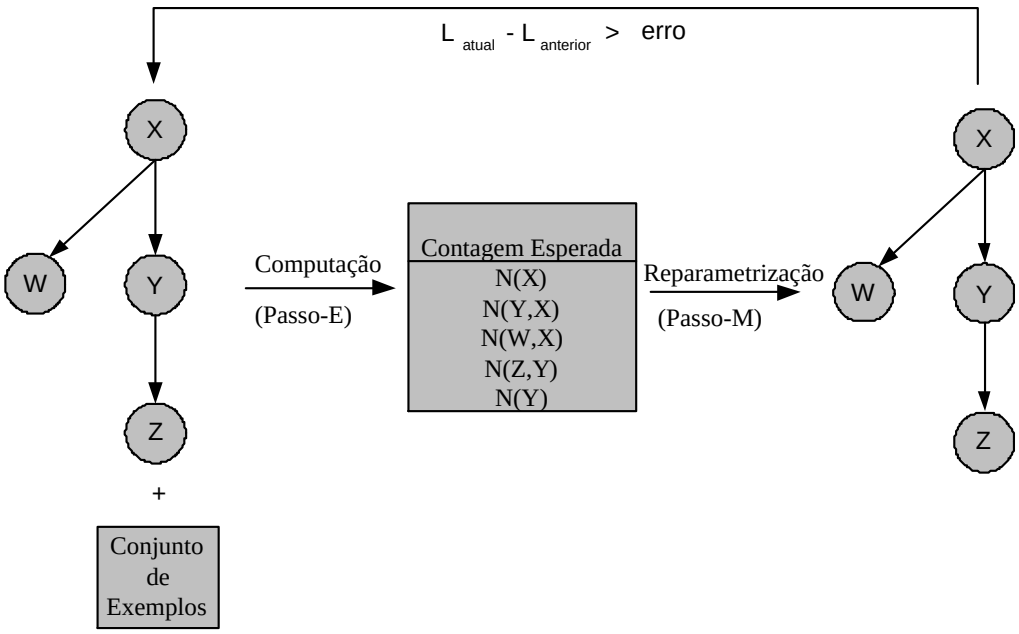


Figura 2.7: Esquema do algoritmo EM

mente, em problemas de aprendizado no mundo real, existem muitos máximos locais que podem não levar a uma boa solução. Alguns trabalhos já foram feitos propondo estratégias para solucionar esta questão (GLOVER, LAGUNA, 1993),(ELIDAN, NINIO, et al., 2002).

2.2.3 Aprendizado de Estrutura Utilizando EM

Uma abordagem comum para aprender a estrutura de redes Bayesianas é avaliar cada possível estrutura de rede de acordo com uma função de avaliação (definida na seção 2.3), escolhendo a de melhor avaliação para ser implementada. A cada proposta de estrutura é preciso re-aprender a CPD das variáveis aleatórias. Como vimos anteriormente, no caso de um *dataset* completo o aprendizado das CPDs é feito através da contagem do número de ocorrências para cada possível associação de valores para a variável em questão e seus pais. Dessa forma, ao efetuar uma alteração na estrutura de rede corrente (adição de uma aresta de X_1 para X_2) apenas a CPD das variáveis envolvidas na alteração precisam ser re-aprendidas. Da mesma forma, se tivermos utilizando uma função de avaliação, como a verossimilhança, que pode ser decomposta em um produto de termos (veja fórmula 2.3), apenas o termo correspondente a alteração efetuada na rede precisa ser re-calculado, ficando

todos os outros invariantes. Esta propriedade permite procedimentos eficientes de aprendizado.

Já quando o *dataset* é parcialmente observado, uma alteração na estrutura da rede corrente implica em re-aprender todas as CPDs e re-calcular todos os termos da função verossimilhança já que é preciso efetuar inferência em toda a rede para poder avaliá-la. Além disso, lembramos que para re-aprender as CPDs de todas as variáveis, deverá ser utilizado, por exemplo, o algoritmo EM. Com isso, são feitas várias chamadas a esse algoritmo, antes de se escolher qual alteração será implementada a estrutura de rede corrente. Se o *dataset* for muito grande, e/ou a quantidade de *variáveis não-observada*s for grande, a eficiência dos algoritmos de aprendizado de estrutura fica comprometida.

Em (FRIEDMAN, 1997; FRIEDMAN, 1998) foi apresentado uma abordagem para buscar a melhor estrutura de rede como um passo dentro do algoritmo EM. O procedimento proposto, chamado *Structural EM*(SEM) (originalmente chamado seleção de modelo utilizando EM (MS-EM)), em cada iteração do algoritmo EM avalia estruturas candidatas computando as contagens esperadas necessárias utilizando a estrutura candidata corrente. Já que esta busca é feita nos dados "completos"é possível utilizar as propriedades de decomposição da função verossimilhança e assim ter um procedimento similar ao caso da base completa.

O algoritmo SEM é implementado usando a arquitetura descrita na Figura 2.8, onde o módulo *Ferramenta de Busca* é responsável por escolher a estrutura candidata que será avaliada. Para cada candidato escolhido, este módulo chama o

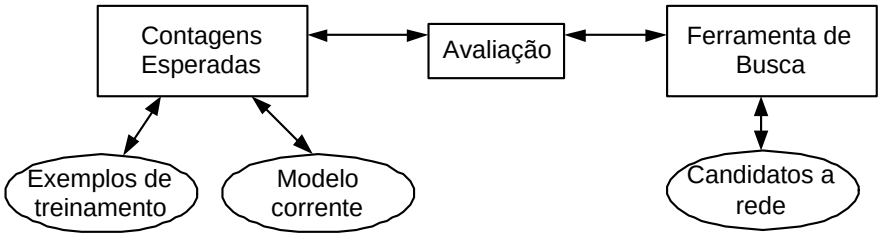


Figura 2.8: Arquitetura da implementação do algoritmo SEM.

módulo *Avaliação*, o qual implementa a função de avaliação. Este módulo solicita as contagens esperadas necessárias para a avaliação ao módulo *Contagens Esperadas*. No caso de um *dataset* completo, o módulo *Contagens Esperadas* responde

as solicitações através da contagens das instâncias nos exemplos de treinamento. No SEM, este módulo computa as contagens esperadas, usando a rede Bayesiana encontrada na iteração anterior.

Considere, por exemplo, a rede Bayesiana exibida na Figura 2.5, como sendo a rede Bayesiana candidata corrente, ou seja a rede Bayesiana encontrada na iteração anterior. Supondo que o módulo *Ferramenta de Busca* escolha as duas estruturas candidatas exibidas na Figura 2.9. Ao ser solicitado o módulo *Contagens Esperadas* retorna as contagens esperadas exibidas na Figura 2.9. Estas contagens foram encontradas como descrito na seção 2.2.2, utilizando a rede Bayesiana corrente.

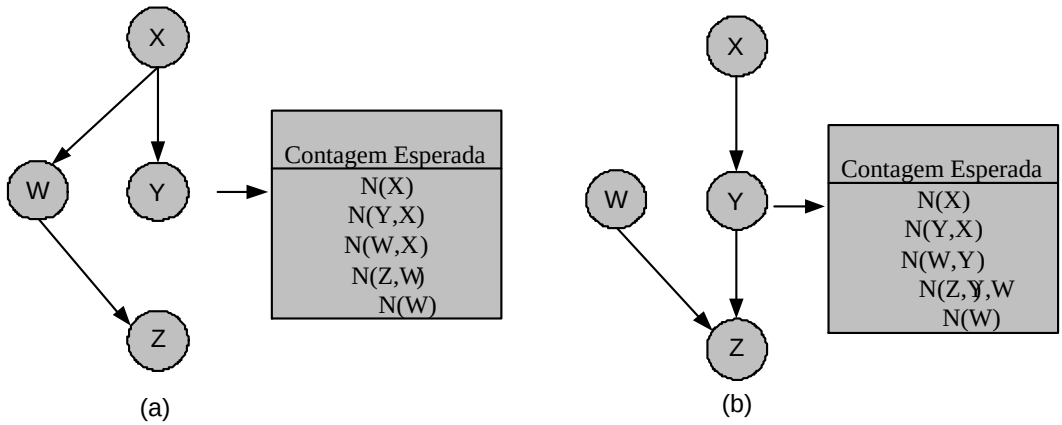


Figura 2.9: Dois exemplos de estruturas candidatas com suas respectivas contagens esperada.

O algoritmo SEM é capaz de aprender estruturas considerando *variáveis não-observadas*, mas ele não atinge bons resultados, a não ser que algum conhecimento a priori da localização das *variáveis não-observadas* e da cardinalidade delas seja conhecida (pelo menos aproximada).

2.3 Funções de Avaliação Probabilística

Nesta seção, descrevemos as duas funções de avaliação probabilística que utilizamos nos nossos algoritmos para avaliar os modelos propostos.

1. Verossimilhança - a função de verossimilhança é definida como:

$$L(H : \mathbf{C}) = \mathbf{P}(\mathbf{C}|H, \mathbf{\Theta}) \tag{2.6}$$

onde H é o modelo sendo avaliado, $\mathbf{C}$ o conjunto de exemplos de treinamento e $\mathbf{\Theta}$ são as distribuições de probabilidade. Considerando que os exemplos são independentes, e utilizando a propriedade de decomposição da probabilidade conjunta (veja fórmula 2.3), temos:

$$P(\mathbf{C}|\mathbf{\Theta}, H) = \prod_{i=1}^{m} P(C_i|\mathbf{\Theta}, H) = \prod_{i=1}^{m}\prod_{j=1}^{v} P(x_{i,j}|\mathbf{\Theta}, H) \tag{2.7}$$

onde m é o número de exemplos no conjunto de treinamento e v o número de variáveis aleatórias no nosso modelo. Por ser de mais fácil manipulação, é usual o uso do *log* da função verossimilhança ($LL(H : \mathbf{C})$):

$$LL(H : \mathbf{C}) = \sum_{i=1}^{m} logP(C_i|\mathbf{\Theta}, H) = \sum_{i=1}^{m}\sum_{j=1}^{v} logP(x_{i,j}|\mathbf{\Theta}, H) \tag{2.8}$$

Olhando do ponto de vista estatístico, quanto maior a log-verossimilhança, mais próximo H está de modelar a distribuição sobre os dados C (FRIEDMAN, GEIGER, et al., 1997).

A log-verossimilhança negativa é uma medida padrão de erro de treinamento e é definida como: $NLL(H|C) = -LL(H|C)$. Neste caso, deseja-se minimizar a log-verossimilhança negativa.

Na seção 2.2.2, utilizamos $L(\theta : \mathbf{C})$ invés de $L(H : \mathbf{C})$, ou seja θ invés de H, para enfatizar que o objetivo era o aprendizado das CPDs, enquanto que no segundo era o modelo, ou seja a estrutura e as CPDs.

2. Verossimilhança condicional - quando o problema que está sendo modelado é de classificação, o objetivo geralmente é descobrir um modelo que forneça corretamente o valor de um grupo de variáveis $\{x_k, ..., x_v\}$ dados os demais atributos $\{x_1, ..., x_{k-1}\}$, ou seja os valores que maximizem $P(x_k, ..., x_v|x_1, ..., x_{k-1})$. Se a medida de performance é a fração de classificações corretas, o ideal em uma tarefa de classificação é encontrar o modelo que tenha o menor erro de classificação possível. Em (FRIEDMAN, GEIGER, et al., 1997) foi mostrado que maximizar a verossimilhança condicional (vide fórmula 2.9) da classe é equivalente a minimizar o erro de classificação. Isto ocorre porque na tarefa de

classificação, somente a parcela relativa à verossimilhança condicional da classe dados os atributos é relevante. Portanto esta função é preferível à verossimilhança em problemas de classificação.

$$CLL(H|C) = \sum_{i=1}^{m} logP(x_{i,k}, ..., x_{i,v}|x_{i,1}, ..., x_{i,k-1}) \tag{2.9}$$

onde $C_i = \{x_{i,k}, ..., x_{i,v}, x_{i,1}, ..., x_{i,k-1}\}$ e $x_{i,k}, ..., x_{i,v}$ representam as classes no exemplo i.

O problema com a log-verossimilhança condicional é que, diferente da log-verossimilhança, não é possível decompô-la em termos separados para cada variável, o que torna a sua computação mais complexa (FRIEDMAN, GEIGER, et al., 1997). Devido a esse problema, em (GROSSMAN, DOMINGOS, 2004) a log-verossimilhança condicional foi usada para guiar o aprendizado apenas da estrutura, semelhante ao que fazemos nesse trabalho. Usar a log-verossimilhança condicional para aprender ambas a estrutura e as CPDs apresenta custos proibitivos.

De forma semelhante, a log verossimilhança condicional negativa pode ser definida como:

$$NCLL(H|C) = -CLL(H|C) \tag{2.10}$$

Capítulo 3

Revisão de Modelos Lógicos

Neste capítulo, revisamos alguns conceitos lógicos importantes para o entendimento das nossas propostas, como refinamento de teoria (WROBEL, 1996) e invenção de predicados (KRAMER, 1995). Este capítulo está dividido da seguinte forma: na seção 3.1 revisamos *cláusulas definidas*. Na seção 3.2 apresentamos os conceitos de revisão de teoria e na seção 3.3 os conceitos de invenção de predicados.

3.1 Cláusulas Definidas

Redes Bayesianas são essencialmente uma representação proposicional do conhecimento (ver figura 3.1): o conjunto de variáveis aleatórias é fixo e finito.

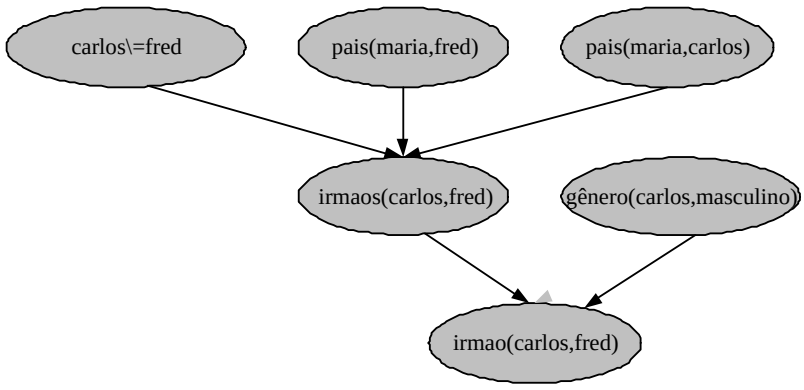


Figura 3.1: Conhecimento de irmandade representado através de uma rede Bayesiana.

Representações mais concisas podem ser conseguidas usando representações de primeira-ordem do conhecimento, tais como *cláusulas definidas*, que consideram a existência de objetos e relações entre eles e podem expressar fatos sobre alguns ou

todos os seus objetos.

Um alfabeto de primeira-ordem é um conjunto de símbolos de predicados e um conjunto de símbolos de função. Constantes são símbolos de função de aridade 0. Uma *cláusula definida* é uma fórmula da forma

$$H : -B_1, ..., B_n$$

onde H e $B_i, \forall i \in [1, n]$ são átomos lógicos. Um *átomo lógico* $p(t_i, ..., t_m)$ é um símbolo de predicado p seguido por uma n-tupla de termos t_i. Um *termo* é uma variável ou um símbolo de função. Quando todos os termos de um átomo são constantes dizemos que este átomo é um *átomo básico*. Uma cláusula definida pode ser lida como H se B_1 e ... e B_n. H é chamado de *cabeça* e $B_1, ..., B_n$ *corpo* da cláusula. Um *programa de cláusulas definidas* é um conjunto de cláusulas definidas.

O *programa de cláusulas definidas* exibido na Tabela 3.1 é a representação de primeira-ordem para o conhecimento representado pela rede Bayesiana da figura 3.1.

Tabela 3.1: *Programa de cláusulas definidas* representando o mesmo conhecimento representado pela rede Bayesiana em 3.1

1.	$irmao(carlos, fred) : -genero(carlos), irmaos(carlos, fred).$
2.	$irmaos(carlos, fred) : -pais(mary, carlos), pais(mary, fred), carlos \backslash = fred.$

Seu alfabeto consiste dos símbolos de predicado $irmao, genero, irmaos$, $pais$ e das constantes $carlos$ e $fred$.

3.2 Refinamento de Teoria

A aquisição de conhecimento é uma tarefa difícil, demorada e com chances de erro. O processo de automaticamente melhorar uma teoria lógica de primeira-ordem, utilizando métodos de aprendizado, é executado pelos sistemas de refinamento de teoria (WROBEL, 1996).

Experiências com sistemas especialistas e sistemas baseados em regras, rapidamente mostraram-se não muito eficientes. O fato do conhecimento ser fornecido por especialistas torna a tarefa demorada e conseqüentemente custosa. Fora isto, muitas das informações poderiam estar incompletas ou até mesmo incorretas.

Uma maneira encontrada para resolver este problema foi o aprendizado indutivo (MITCHELL, 1997), onde a idéia era minimizar o conhecimento fornecido por um especialista fazendo com que o sistema extraísse o conhecimento através de exemplos aprendendo uma teoria. Tomando classificação como exemplo, o sistema receberia instâncias de classes e aprenderia a generalização destas, utilizando-se dos exemplos. Com a generalização seria possível resolver problemas futuros de classificação. Apesar de muitas técnicas desenvolvidas para este fim terem sido bem sucedidas elas requeriam uma base de exemplos de treinamento muito grande.

Uma outra abordagem seria o refinamento de teoria.

O processo de refinamento de uma teoria pode ser dividido em dois tipos: revisão e reestruturação de uma teoria. Ambos têm por objetivo aumentar a qualidade da teoria.

A tarefa de revisar significa mudar o conjunto de respostas de uma dada teoria, isto é melhorar sua capacidade de inferência proporcionando assim a adição de respostas que haviam sido perdidas ou a remoção de respostas incorretas. No primeiro caso falamos em generalização da teoria e no segundo de especialização. Já a tarefa de reestruturação não muda o conjunto de respostas da teoria fornecida, o objetivo é a melhora de como as respostas são encontradas . Pode-se distinguir dois grupos de reestruturação, um com objetivo de melhorar o desempenho e o outro em facilitar o entendimento do usuário. O gráfico na figura 3.2 ilustra a idéia descrita.

Como neste trabalho o que utilizamos é revisão de teoria, o enfoque será nesta questão, para mais detalhes veja (WROBEL, 1996).

A figura 3.3 apresenta um esquema para revisão de teoria. Um sistema de revisão de teoria recebe uma teoria inicial e um conjunto de exemplos. Esta teoria inicial pode ser dividida em duas partes: uma assumidamente correta (conhecimento preliminar (BK)) e outra que pode ser modificada (H) pela revisão. Quando revisando teorias de primeira-ordem o conjunto de exemplos é dividido em exemplos positivos (E^+) e exemplos negativos (E^-). Neste caso, o processo de revisão deve gerar uma teoria final tal que irá provar todos os exemplos positivos em E^+ e nenhum dos exemplos negativos em E^-, de forma que a teoria final será consistente com o *dataset*. Também é desejável que a teoria final considere um critério adicional

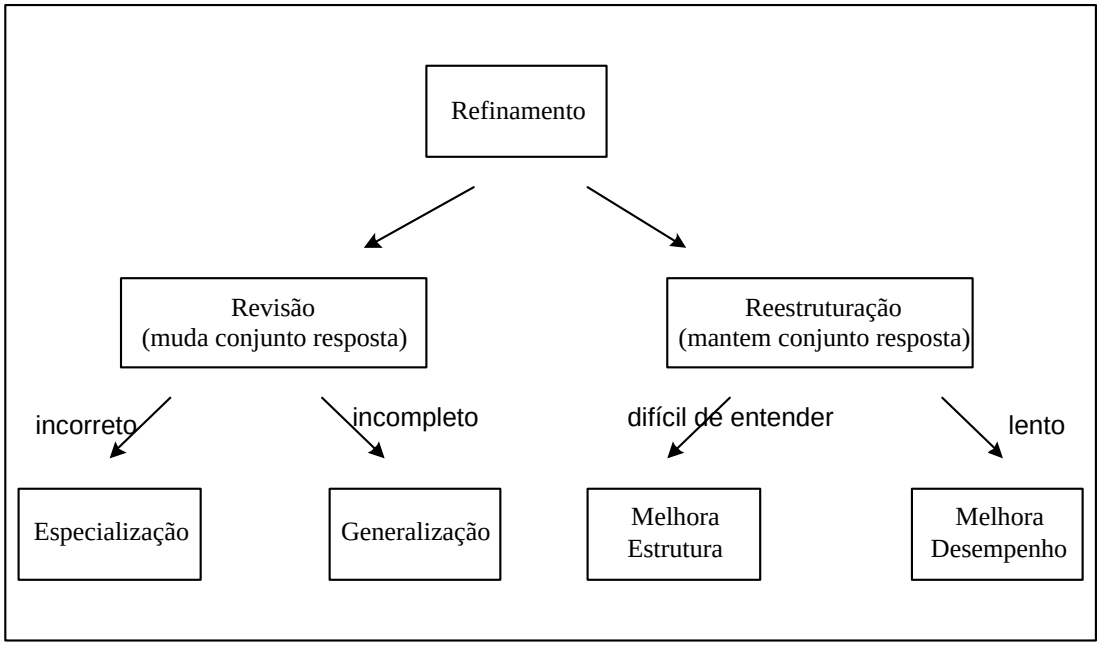


Figura 3.2: Esquema de Refinamento de Teoria definido em (WROBEL, 1996)

de qualidade tal como minimalidade, dessa forma a teoria final será tão semanticamente e sintaticamente similar a teoria inicial quanto possível. Já o aprendizado em ILP, pode ser visto como uma revisão de teoria onde a teoria inicial consiste apenas do conhecimento preliminar e portanto somente ocorre a adição de novas cláusulas.

A maioria dos sistemas de revisão de teoria, utiliza cláusulas definidas como a linguagem da teoria e átomos básicos (*ground atoms*) como a linguagem dos exemplos.

Vamos considerar o exemplo da teoria proposicional mostrada na figura 3.4, para entendermos mais facilmente o raciocínio de identificação das causas de uma classificação incorreta. Considere V como verdadeiro e F como falso.

A teoria classifica o primeiro exemplo, dito um exemplo positivo de H, como um exemplo negativo, indicando que a teoria é muito específica nesse caso. Isto poderia estar ocorrendo porque

1. Deveriam existir mais regras para a prova de H, ou

2. uma ou mais regras que provam H tem mais antecedentes do que o necessário, ou

3. as regras para X ou Y são muito específicas, isto é, tem mais antecedentes do

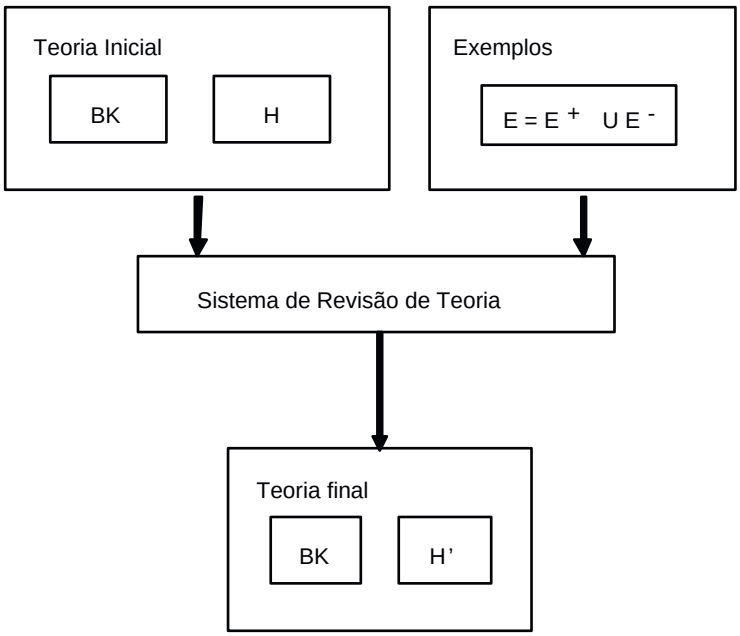


Figura 3.3: Esquema de revisão de teorias de primeira-ordem

Teoria :

H ← X, Z.
H ← Y, G.
X ← A, B.
Y ← C, D.

Exemplos:

1.G = V, A = F, B = F, D = F, Z = F, H = T
2.G = F, A = V, B = V, D = V, Z = V, H = F

Figura 3.4: Exemplo proposicional de identificação das causas de uma classificação incorreta

que o necessário, ou

4. deveriam existir mais regras para a conclusão de X ou Y.

O segundo exemplo, dito um exemplo negativo de H, é classificado como um exemplo positivo pela teoria. Isto significa que a teoria é muito geral neste caso. Isto pode estar ocorrendo porque:

1. Existem muitas maneiras de concluir H, isto é, uma das regras que esta provando H deveria ser excluída, ou

2. as regras para concluir H são muito gerais e precisam de antecedentes adicionais, ou

3. existem mais regras do que o necessário concluindo X ou Y, ou

4. as regras para concluírem X ou Y são muito gerais e precisam de antecedentes adicionais.

Estas observações não somente diagnosticam os erros na teoria, como também sugerem maneiras pelas quais a teoria poderia ser revista para corrigir as falhas.

Vamos exemplificar agora para o caso de primeira ordem. Considere a teoria de primeira ordem e o conjunto de exemplos de treinamento descritos na figura 3.5.

Teoria:
1. avo(A,B): − pai(A,C),pais(C,B).
2. avo(A,B): − pai(A,B).
3. pais(A,B) : − pai(A,B).
4. pais(A,B) : − mãe(A,B), gênero(B,feminino).

Exemplos:
Positivo: avo(walter,jim), pai(walter,carol),mae(carol,jim).
Negativo: avo(lee,jim) ,pai(lee,jim).

Figura 3.5: Exemplo de primeira ordem de identificação das causas de classificação incorreta

Como se pode ver o exemplo positivo $avo(walter, jim)$ não está sendo provado, pois falha no predicado *gênero(carol,feminino)*, já que não existe nenhum fato a respeito dele. Uma alteração na teoria poderia ser então, a retirada do antecedente *gênero(B,feminino)* da cláusula 4. Já o exemplo negativo é provado, o que não deveria acontecer, e podemos verifica que a responsável é a cláusula 2, pois ela diz que basta A ser pai de B para que A também seja avó de B o que sabemos não ser verdade, logo uma revisão seria a exclusão desta cláusula da teoria. Teríamos então como teoria resultante a mostrada na figura 3.6.

Teoria Revista:
1. avo(A,B) : − pai(A,C) , pais(C,B).
3. pais(A,B) : − pai(A,B).
4. pais(A,B) : − mae(A,B).

Figura 3.6: Exemplo de primeira ordem, teoria resultante

3.2.1 Pontos de Revisão

Como podemos ver pelos exemplos da seção anterior, quando um exemplo positivo ou negativo é selecionado, ele determina o tipo de revisão que deve ser feita, generalização ou especialização respectivamente.

Em uma teoria com várias cláusulas, vários predicados sendo definidos, muitas cláusulas podem estar envolvidas na prova de um exemplo negativo ou muitas cláusulas podem ser generalizadas para que um exemplo positivo passe a ser provado. A qualidade das revisões propostas depende muito das cláusulas que serão escolhidas para serem revistas, definindo os *pontos de revisão*. Dependendo do tipo de exemplo que está sendo considerado podemos definir dois tipos de pontos de revisão:

- *Generalização* - o literal em uma cláusula responsável pela falha na prova de um exemplo positivo (*ponto de falha*) e outros antecedentes (*pontos de contribuição*) que podem ter contribuído para esta falha atribuindo valores incorretos para as variáveis; Como exemplo, considere a relação abaixo também do problema da família

  irmã(X,Y) : − filha(X,Z), pais(Z,Z).

  Quando tentarmos executar o predicado irmã, o antecedente pais será guardado como um ponto falho, pois não tem como alguém ser pai dele mesmo, e o antecedente filha como um ponto de contribuição, pois foi dele a escolha da variável Z responsável pela falha no predicado pais.

  Qualquer predicado destes literais anotados também é considerado um ponto de revisão (baseado em predicado). Estes pontos de revisão são utilizados pelo operador de identificação (a ser definido), que procura generalizar a definição desses predicados.

- *Especialização* - cláusulas usadas em provas de exemplos negativos.

Cada ponto de revisão tem um *potencial*, que é o máximo crescimento em desempenho que pode ser proporcionado à teoria a partir de modificações neste ponto. Ou seja, é o número de exemplos que verificaram a necessidade de revisão neste ponto, e que podem passar a ser classificados corretamente após a revisão ser feita.

A especificação dos *pontos de revisão* determina o tipo de modificação que será aplicado para fazer a teoria consistente com o *dataset*, definindo tipos de *operadores de revisão*.

3.2.2 Operadores de Revisão

Revisão de teoria depende dos operadores que proporão modificações para cada ponto de revisão. Qualquer operador usado em aprendizado de máquina de primeira-ordem pode ser utilizado em sistemas de revisão de teoria. Em seguida, descrevemos os operadores de revisão mais comuns, originalmente definidos em (RICHARDS, MOONEY, 1995).

Os operadores para especialização são: ·

- *Exclusão de regra* - existem duas restrições para este operador. Ele não pode excluir uma cláusula que seja ou a cláusula base de um predicado recursivo ou a única cláusula que define um determinado predicado. Neste último substitui-se a cláusula a ser excluída pela regra *conceito :- fail.*

- *Adição de antecedente* - adiciona-se antecedentes a uma cláusula na tentativa de fazer com que não sejam provadas todas os exemplos negativos. Se o adição destes antecedentes fizer com que exemplos positivos deixem de ser provados, adiciona-se esta cláusula especializada à teoria e recomeça a especialização com a cláusula original, procurando especializações alternativas que retenham a prova das outros exemplos positivos enquanto eliminando os negativos. Existem dois algoritmos para adicionar antecedentes à uma cláusula: o primeiro é o adição de antecedentes *hill-climbing* - este adiciona um antecedente de cada vez procurando pelo que proporciona o melhor desempenho da teoria. Algumas vezes nenhum dos antecedentes diminui o desempenho, mas também não aumenta. Para estes casos é preciso que vários antecedentes sejam adicionados de uma só vez, utiliza-se então o segundo algoritmo, *relational pathfinding*, o qual tenta encontrar os antecedentes que tem relação entre si, para mais informações (RICHARDS, MOONEY, 1995).

Em contraste com operadores de especialização, os quais modificam uma cláusula

existente, operadores de generalização não necessariamente precisam modificar uma cláusula existente, podem também adicionar uma cláusula totalmente nova.

Os operadores para generalização são:

- *Exclusão de antecedente* - temos dois métodos: a) o primeiro é a exclusão de antecedentes *hill-climbing*. Este método vai excluir um antecedente de cada vez na cláusula, tornando-a mais geral. O antecedente a ser escolhido é aquele que aumenta o desempenho ao máximo, enquanto não prova nenhum exemplo negativo. Este processo é repetido até que o desempenho da teoria não possa mais ser melhorado. b) exclusão de múltiplos antecedentes- algumas vezes a prova de um exemplo apenas é afetada com a exclusão de mais de um antecedente de uma só vez. Primeiro todos os antecedentes cuja exclusão não permite que exemplos negativos sejam provados, são reunidos; então combinações destes antecedentes são feitas na procura daquela que permite o maior número de exemplos positivos provados sem provar negativos. O algoritmo não para quando todas os exemplos positivos foram provados, continua excluindo tantos antecedentes quanto puder. Este algoritmo é computacionalmente caro; entretanto só é chamado quando o *hill-climbing* não proporciona nenhuma revisão.

- *Adiciona regra* - é uma revisão baseada em cláusula. Deixa a cláusula original na teoria e gera novas baseadas nesta. O processo é feito em duas etapas. Primeiro copia a cláusula original e usando o algoritmo de exclusão de antecedente *hill-climbing*, exclui antecedentes sem permitir que nenhum exemplo negativo seja provado e permitindo também que alguns positivos, que antes não eram provados passem a ser (mesmo que isto permita a prova de negativos). Então cria uma ou mais especializações desta regra, utilizando o operador de adição de antecedentes, para permitir a prova dos exemplos positivos desejados enquanto elimina os negativos.

- *Identificação* - constrói uma cláusula nova para generalizar a definição de um antecedente que falhou na prova de um exemplo positivo. Melhor que desenvolver a cláusula do nada, executa um passo de resolução inversa (MUGGLE-

TON, 1992) usando duas regras existentes no domínio da teoria. Por exemplo, considere o conjunto de cláusulas tiradas do problema da família.

tio(X,Y) : − gênero(X,masculino),tios(X,Y).

tio(X,Y) : − gênero(X,masculino), irmãos(X,Z), pais(Z,Y).

tios(X,Y) : − casado(X,Z), irmãos(Z,W), pais(W,Y).

tia(X,Y) : − gênero(X,feminino), tios(X,Y).

Quando um exemplo de tia é apresentado onde esta é tia de sangue, este exemplo não será provado. Um dos pontos de falha é a chamada para *tios*. O operador de *Identificação* procurará por maneiras de fornecer uma outra cláusula para este predicado, e encontra uma das duas cláusulas para tio. A revisão proposta substituirá a segunda cláusula para tio por:

tios(X,Y) : − irmãos(X,Z), pais(Z,Y).

- *Absorção* - Vai substituir uma cláusula já existente por uma nova a partir da substituição dos seus antecedentes falhos pela cabeça de uma cláusula, cujo o conjunto dos seus antecedentes contém os antecedentes definidos como sendo falhos. Suponha como exemplo o conjunto de cláusulas abaixo.

  tio(X,Y) : − gênero(X,masculino), irmãos(X,Z), pais(Z,Y).

  tios(X,Y) : − irmãos(X,Z), pais(Z,Y).

  tios(X,Y) : − casado(X,Z), irmãos(Z,W), pais(W,Y).

  Quando um exemplo para tio, onde este não é um tio de sangue, é apresentado, este não será provado. O ponto de falha será em *irmãos* ou *pais*. O operador para absorção achará um antecedente similar na segunda cláusula para *tios*, então substituirá a cláusula para tio por

  tio(X,Y) : − gênero(X,masculino), tios(X,Y).

Revisão de teoria considerada nesta tese é diferente de revisão de crenças (GäRDENFORS, 1992). Revisão de crenças tem por objetivo encontrar a menor exclusão de crenças necessárias para tornar a teoria consistente com uma nova crença. Entretanto, não é considerado o problema de generalização ou especialização da teoria

através da adição de restrições, ou seja adição de antecedentes a alguma regra. Além disso, revisão de crença geralmente foca em uma alteração mínima da semântica, o que exige uma memorização das exceções invés de produzir uma especialização que generalizaria para novos casos, como revisão de teorias faz.

FORTE

Um exemplo de sistema de revisão é o FORTE (First Order Revision of Theories from examples) (RICHARDS, MOONEY, 1995), cujo o algoritmo em alto nível está descrito em Algoritmo 1. FORTE é um sistema totalmente automatizado que executa uma busca *hill-climbing* no espaço de operadores de especialização e generalização na tentativa de encontrar uma revisão minima da teoria que a faça consistente com um conjunto de treinamento. FORTE funciona da seguinte maneira: primeiro, ele identifica todos os pontos de revisão na teoria corrente. Em seguida, ele gera um conjunto de propostas de revisão para cada ponto de revisão começando pelo ponto de revisão com maior *potencial*. Depois de gerar as propostas de revisão, cada proposta recebe uma *avaliação*, definida como a melhora que será proporcionada à acurácia da teoria. A melhor proposta gerada até o momento, isto é a revisão que fornece a melhor avaliação, é mantida como a melhor proposta até o momento. Em caso de empate na acurácia, FORTE escolhe a proposta que fornece a menor teoria. FORTE para de gerar propostas de revisão quando o potencial do próximo ponto de revisão é menor do que a avaliação da melhor proposta de revisão até o momento. Se a melhor proposta de revisão realmente melhorar a teoria (considerando acurácia ou compactação) ela é implementada. Este processo iterativo continua até que não seja mais possível melhorar a teoria.

Quando aplicando um operador de revisão não é preciso utilizar todo o conjunto de exemplos de treinamento, somente um subconjunto consistindo daqueles exemplos onde as provas dependem das cláusulas que estão sendo revistas.

Trabalhos como: (WOGULIS, PAZZANI, 1993; RICHARDS, MOONEY, 1995) (TOWELL, SHAVLIK, 1994; GARCEZ, ZAVERUCHA, 1999; BUNTINE, 1991; WROBEL, 1996; RAMACHANDRAN, MOONEY, 1998) mostram que sistemas de revisão de teorias, de primeira-ordem e proposicionais respectivamente, podem,

Algoritmo 1 Algoritmo FORTE, proposto em (RICHARDS, MOONEY, 1995)

```
1: repete
2:     gera pontos de revisão;
3:     ordena pontos de revisão por potencial(maior para menor);
4:     para todo ponto de revisão faça
5:         se o potencial do ponto de revisão for menor que a pontuação da melhor revisão
           atualizada então
6:             break;
7:         gera possíveis revisões;
8:         atualiza melhor revisão encontrada;
9:     se a melhor revisão melhora a teoria então
10:        implementa melhor revisão;
11: até que nenhuma revisão melhore a teoria
```

partindo de uma teoria aproximadamente correta, aprender teorias mais precisas e utilizando menos dados do que sistemas puramente indutivos.

3.3 Invenção de Predicados

Às vezes quando aprendendo uma teoria a partir de um *dataset*, o vocabulário inicial não é suficiente. *Invenção de predicados* é uma abordagem que estende o vocabulário com novos predicados necessários para resolver com sucesso a tarefa de aprendizado. Nesta seção são descritas algumas das técnicas utilizadas para inventar predicados (KRAMER, 1995; STAHL, 1996) em ILP e que foram relevantes para o nosso trabalho.

Quando trabalhando com invenção de predicados três questões precisam ser discutidas:

- **quando inventar predicados:** definir situações onde é necessário e/ou útil introduzir novos predicados.
- **como inventar predicados:** definir mecanismos para inventar novos predicados. Isso inclui a especificação da estrutura do(s) argumento(s) do novo predicado assim como a definição deste.
- **avaliar os novos predicados:** dependendo dos mecanismos utilizados o espaço de busca de novos predicados pode ser muito grande, por esse motivo métodos para tentar reduzir este espaço de busca e/ou percorrê-lo de forma

eficiente é crucial, além da utilização de funções de avaliação para avaliar os novos predicados propostos

As três subseções seguintes detalham melhor estas questões.

3.3.1 Quando Inventar Predicados

Como mencionado acima, invenção de predicados é utilizada para introduzir novos predicados ao vocabulário inicial quando este é incapaz ou insuficiente para a tarefa de aprendizado. Dependendo do algoritmo de aprendizado utilizado esta insuficiência pode ser considerada de forma diferente e consequentemente invenção de predicados pode ser aplicada como:

a) *operação de reformulação* - tem por objetivo melhorar a estrutura da teoria

b) *operação de transformação* - deseja construir teorias eficientes

c) *bias shift* - essa abordagem é utilizada para tentar permitir que o aprendizado seja bem sucedido

Nos dois primeiros casos decide-se pela utilização de invenção de predicados quando uma determinada função de avaliação indica que a teoria é insuficiente com respeito a qualidade e/ou eficiência. A operação de reformulação utiliza um novo predicado para reformular a teoria de forma a encontrar uma teoria menor e/ou mais compacta. A teoria resultante continua inferindo as mesmas coisas que a teoria original. Já o operador de transformação altera a teoria corrente, utilizando os novos predicados para mudar a definição dos predicados correntes. A abordagem de *Bias shift* é utilizada quando não for possível encontrar uma teoria consistente com o vocabulário corrente.

A decisão por utilizar invenção de predicados para resolver a falha no aprendizado de uma teoria não é simples. Nem sempre a introdução deste novo predicado solucionará o problema. É importante então verificar a *utilidade* deste novo predicado, isto é, o seu potencial em fazer a tarefa de aprendizado ser bem sucedida.

3.3.2 Como Inventar Predicados

Dependendo do critério de decisão utilizado para definir a necessidade de invenção de predicados, diferentes mecanismos podem ser adotados. Abaixo descrevemos alguns.

a) Mecanismos para reformulação

Os mecanismos de reformulação introduzem um novo predicado como um predicado intermediário. A idéia básica é procurar por definições onde seja interessante introduzir um predicado intermediário. Uma abordagem que pode ser utilizada é resolução inversa (MUGGLETON, BUNTINE, 1988). Um exemplo é fornecido abaixo

$irmao(X,Y) : -genero(X, masculino), pais(Z,X), pais(Z,Y), X <> Y.$

$irma(X,Y) : -genero(X, feminino), pais(Z,X), pais(Z,Y), X <> Y.$

$tio(X,W) : -genero(X, masculino), pais(Z,X), pais(Z,Y), X <> Y, pais(Y,W).$

$tia(X,W) : -genero(X, feminino), pais(Z,X), pais(Z,Y), X <> Y, pais(Y,W).$

Ao analisar a teoria acima, verifica-se 3 antecedentes comuns as 4 cláusulas: $pais(Z,Y)$, $pais(Z,X)$ e $X <> Y$. Um novo predicado é então inventado ($irmaos$). O conjunto de argumentos desse novo predicado é um dos definidos no *predicate utility lattice* (MUGGLETON, 1993). O lattice contém todas as combinações possíveis das variáveis dos 3 antecedentes considerados: X,Y e Z. Como exibido na figura 3.7, o predicado inventado $irmaos$ pode ser ternário, binário ou unário.

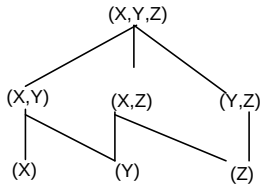


Figura 3.7: Lattice definindo os possíveis argumentos de um predicado inventado

O uso de uma função de avaliação definirá quais dentre os argumentos possíveis será o escolhido, como por exemplo (X,Y), definindo assim o predicado inventado $irmaos(X,Y)$. Nesse caso, a cláusula que define o novo predicado é

$$irmaos(X,Y) : -pais(Z,Y), pais(Z,X), X <> Y$$

Em seguida os três antecedentes são substituídos pelo predicado inventado *irmaos(X,Y)*, em cada uma das 4 cláusulas. A nova teoria é fornecida a seguir.

$irmao(X,Y) : -genero(X, masculino), irmaos(X,Y).$

$irma(X,Y) : -genero(X, feminino), irmaos(X,Y).$

$tio(X,W) : -genero(X, masculino), irmaos(X,Y), pais(Y,W).$

$tia(X,W) : -genero(X, feminino), irmaos(X,Y), pais(Y,W).$

$irmaos(X,Y) : -pais(Z,X), pais(Z,Y), X <> Y.$

A teoria reformulada tem o mesmo poder de inferência da teoria inicial sendo que com menos redundância, reduzindo o número de literais, definindo uma teoria mais compacta.

Mecanismos para Transformação

Esse mecanismo inventa um predicado que ao ser introduzido na teoria muda as respostas antes retornadas. Ele pode ser utilizado para melhorar a eficiência da teoria original ou para especializar uma cláusula geral.

Bias Shift

O mecanismo de *Bias Shift* especializa uma teoria inconsistente baseando-se em 3 passos:

a) determinar as cláusulas gerais que precisam ser corrigidas com o novo predicado

b) definir os argumentos para o predicado inventado

c) introduzir o novo predicado na cláusula escolhida em a)

Considere por exemplo, a cláusula

$$responsavel(Z,X) : -veiculo\_envolvido(X,Y), dono(Z,Y) \tag{3.1}$$

(STAHL, 1996), onde o motorista Z é responsável pela violação X quando o seu veiculo Y estiver envolvido nesta violação. Considere também que *estacionamento proibido* e *má conservação* do veículo são *violações leves* enquanto que *excesso de*

velocidade e *avançar sinal* são *violações graves*. De acordo com a cláusula 3.1, exemplos como :

$responsavel(irene, estacionamento\_proibido)$,

$responsavel(michel, ma\_conservacao)$,

$responsavel(irene, avanco\_sinal)$ e

$responsavel(michel, excesso\_velocidade)$

seriam provados, determinando que um motorista será responsabilizado independente se a violação foi grave ou leve, ou seja a cláusula está muito geral. Se o *dataset* contiver como exemplos positivos as *violações leves* e negativos as *violações graves*, precisamos especializar esta cláusula para que a teoria passe a ser consistente com o *dataset*. Como não existe no vocabulário inicial um predicado que possa ser inserido no corpo desta cláusula especializando-a, é preciso estender o vocabulário criando um novo predicado (new). O conjunto de argumentos desse novo predicado é um dos definidos no *predicate utility lattice* (MUGGLETON, 1993). No nosso exemplo poderia ser um argumento, Z ou X indicando o motorista ou a violação respectivamente, ou dois argumentos. Como este predicado está sendo criado para resolver um problema sobre as possíveis violações que um motorista pode efetuar, vamos considerar que o argumento escolhido foi X. A cláusula especializada é:

$$responsavel(Z, X) : -veiculo\_envolvido(X, Y), dono(Z, Y), new(X)$$

Utilizando os exemplos positivos, definimos $new(X)$ através da introdução de dois novos fatos: $new(estacionamento\_proibido)$ e $new(ma\_conservacao)$.

3.3.3 Avaliar os Novos Predicados

Devido ao grande espaço de potenciais novos predicados, técnicas eficientes de poda e buscas heurísticas são cruciais para uma invenção de predicados ser bem sucedida. Esses recursos são importantes em diferentes estágios do processo de invenção de predicados.

Busca por Argumentos

Quando inventando um predicado é preciso escolher a estrutura apropriada dos seus argumentos. Na abordagem *Bias shift*,por exemplo, em princípio todas as variáveis da cláusula sendo especializada, constantes e outros termos mais complexos podem ser relevantes para excluir as exceções. Com isso podemos ter um espaço de busca bastante grande. Um importante resultado que vale independente da linguagem que está sendo utilizada é que é suficiente considerar apenas termos simples, ou seja variáveis e constantes como argumentos potenciais do predicado sendo inventado (STAHL, WEBER, 1994). Com isso o espaço de busca reduz drasticamente e justifica a utilização do *predicate utility lattice* (MUGGLETON, 1993), que define uma ordem de acordo com o potencial de cada argumento possível. A figura 3.7 exibe o *predicate utility lattice* com os possíveis argumentos para o predicado inventado $new()$ para a cláusula 3.1.

Capítulo 4

Revisão Lógica Probabilística

Neste capítulo, revisamos o nosso sistema de revisão PFORTE, para que no Capítulo 7 possamos estendê-lo com dois novos *operadores de revisão*, que consideram invenção de predicados.

Na seção 2.2, vimos que rede Bayesiana é uma forma de representação de conhecimento probabilístico. Por exemplo, a rede Bayesiana da Figura 4.1 nos permite inferir com que probabilidade *carlos* é irmão de *fred*.

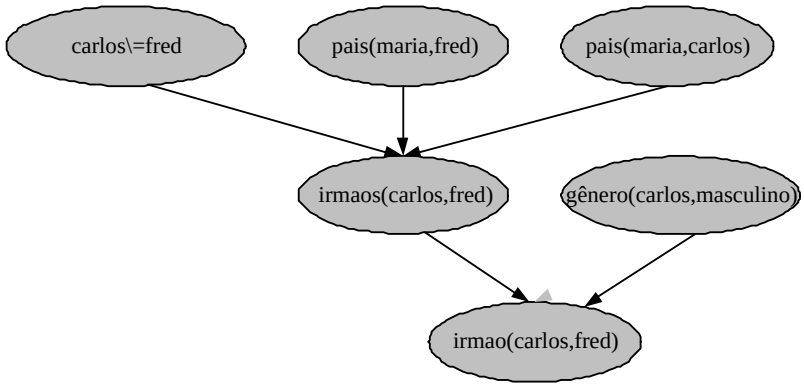


Figura 4.1: Conhecimento de irmandade representado por uma rede Bayesiana.

Se quiséssemos inferir a probabilidade com que outra dupla de objetos, por exemplo *bob-joão*, são irmãos precisaríamos de outra rede Bayesiana, já que redes Bayesianas são essencialmente uma representação proposicional do conhecimento.

Representações mais concisas podem ser conseguidas usando representações de primeira-ordem do conhecimento, tais como *cláusulas definidas* (seção 3.1), que permitem a modelagem de objetos suas propriedades e relações entre eles. Por

exemplo, a *cláusula definida*

$$irmao(X,Y) : -genero(X, masculino), pais(Z,X), pais(Z,Y), X <> Y$$

define que o objeto X é irmão do objeto Y, se X deter a propriedade de ser $masculino$, existir um terceiro objeto Z que é *pai* ou *mae* (relação ($pais$)) dos objetos X e Y e que X e Y são objetos distintos. De acordo com um *dataset* é possível verificar para que duplas de objetos vale a relação *irmao*. Vamos supor que a partir do exemplo relacionado a dupla $carlos - fred$ ($irmao(carlos, fred)$) e do conhecimento preliminar tenhamos a seguinte instanciação para a cláusula acima:

$irmao(carlos, fred) : -genero(carlos, masculino), pais(mary, carlos),$

$$pais(mary, fred), bob <> fred$$

Entretanto existe uma duvida se *mary* realmente é o pai de *carlos*, o que coloca em dúvida se *carlos* é realmente irmão de *fred*. Como representar essa incerteza?

Apesar de *cláusulas definidas* serem uma forma de representação de conhecimento bastante expressiva, não são capazes de representar incertezas, como as redes Bayesianas. Dessa forma, tem sido grande o interesse em integrar modelos baseados em lógica de primeira-ordem com mecanismos de raciocínio probabilístico definindo uma teoria probabilística de primeira-ordem. Alguns exemplos são: *Probabilistic Relational Model* (KOLLER, 1999) (FRIEDMAN, GETOOR, et al., 1999), *Independent Choice Logic* (POOLE, 1993), *Bayesian Logic Program (BLP)* (KERSTING, DE RAEDT, 2001c), *Constraint Logic Programming (CLP(BN))* (COSTA, PAGE, et al., 2003), *Stochastic Logic Program (SLP)* (MUGGLETON, 2002), *Markov Logic Network* (MLN) (RICHARDSON, DOMINGOS, 2006) e *Probabilistic Prolog* (DE RAEDT, KIMMIG, et al., 2007; DE RAEDT, KERSTING, et al., 2008). A tarefa de aprender teorias probabilísticas de primeira-ordem é geralmente chamada de Aprendizado Estatístico Relacional (*Statistical Relational Learning*-SRL).

Em (REVOREDO, ZAVERUCHA, 2002; PAES, REVOREDO, et al., 2005b; PAES, REVOREDO, et al., 2006) utilizamos *Bayesian Logic Programs* (BLPs) como nossa linguagem de primeira-ordem probabilística para propor um sistema de revisão de BLPs denominado PFORTE. As duas seções seguintes detalham BLPs e o sistema PFORTE.

4.1 Programa Lógico Bayesiano

Em (LANGLEY, 1995), as redes Bayesianas foram representadas como programas de cláusulas definidas proposicionais. A idéia principal do BLP é generalizar esse trabalho para a lógica de primeira-ordem, considerando cláusulas definidas de primeira-ordem invés de proposicional. Isto é, permitir variáveis lógicas e a interpretação do grafo de dependências de um programa de cláusula definidas como a estrutura de uma rede Bayesiana. As seguinte definições são importantes para a compreensão do formalismo do BLP.

- Um *predicado probabilístico* é um predicado com um domínio associado. No domínio da *Familia*, por exemplo, vamos considerar o predicado probabilístico $genero(X)$ invés do predicado $genero(X, masculino)$ ou $genero(X, feminino)$. Esse predicado probabilístico têm então associado o domínio $D_{genero} = \{masculino, feminino\}$.

- Uma *cláusula probabilística* é uma cláusula definida sem funções com a cabeça e algum antecedente como sendo um predicado probabilístico e uma distribuição de probabilidade associada. No domínio da *Família*, por exemplo, uma *cláusula probabilística* define relações tais como:

$$irmao(X, Y) : -genero(X), irmaos(X, Y)$$

onde *genêro* e *irmãos* são predicados probabilístico com domínio: $D_{genero} = \{masculino, feminino\}$ e $D_{irmaos} = \{verdadeiro, falso\}$ e ela tem uma distribuição de probabilidades condicional, como a exibida na Tabela 4.1, associada a ela.

Tabela 4.1: CPD associada a cláusula $irmao(X, Y) : -genero(X), irmaos(X, Y)$.

GENERO(X)	IRMAOS(X,Y)	P(IRMAO(X,Y))
MASCULINO	VERDADEIRO	0.97
MASCULINO	FALSO	0.03
FEMININO	VERDADEIRO	0.03
FEMININO	FALSO	0.01

Dessa forma, um programa em lógica Bayesiano é um conjunto de cláusulas probabilísticas. Em um BLP todas as cláusulas probabilísticas são *range-restricted*, isto é, todas as variáveis que ocorrem na cabeça devem ocorrer no corpo.

4.1.1 Procedimento de Resposta a Consultas do BLP

Dado um BLP pode-se fazer perguntas à ele tais como: qual é a distribuição de probabilidade de *carlos* ser *irmao* de $fred$? Uma *consulta probabilística* é definida como uma expressão da forma:

$$?-Q_1,...,Q_n|ev_1,...,ev_m$$

onde $n > 0$ e $m \geq 0$. O objetivo é encontrar a distribuição de probabilidade

$$P(Q_1,...,Q_n|ev_1,...,ev_m)$$

das *variáveis de consulta* $Q_1,...,Q_n$ considerando os valores $ev_1,...,ev_m$ para as variáveis de evidência $Ev_1,...,Ev_m$. No domínio da *Familia*, por exemplo, podemos ter a consulta:

$?-irmao(carlos,fred)|pais(mary,carlos)=V, pais(mary,fred)=V,$

$irmaos(carlos,fred)=V$

onde V representa o valor Verdadeiro do domínio.

Para responder uma consulta, BLP usa a noção de *Knowledge-Based Model Construction (KBMC)* (HADDAWY, 1999), onde primeiro uma rede Bayesiana é construída e então qualquer algoritmo de inferência é usado para computar a distribuição de probabilidade.

Uma questão que surge é se é necessário construir a rede Bayesiana completa, considerando todas as variáveis do domínio para ser capaz de efetuar inferência e conseqüentemente responder consultas probabilísticas. Para responder essa questão, em (KERSTING, DE RAEDT, 2001b) é provado que a *rede suporte* (NGO, HADDAWY, 1997; KERSTING, DE RAEDT, 2001a) de um conjunto finito de variáveis $\mathbf{X}$ é suficiente para computar $P(\mathbf{X})$. Dessa forma, invés de construir a rede Bayesiana completa é suficiente considerar a rede suporte da consulta probabilística.

A rede suporte da variável X é definida como a sub-rede induzida de

$$\{X\} \cup \{Y|Y\, influencia\, X\}$$

A rede suporte de um conjunto de variáveis $\{X_1, ..., X_k\}$ é a união das redes suportes para cada variável X_i.

O algoritmo 2, adaptado de (KERSTING, DE RAEDT, 2001b; KERSTING, DE RAEDT, 2005) mostra como construir a rede suporte usada para responder uma consulta probabilística.

Algoritmo 2 Algoritmo de rede suporte (adaptado de Kersting e De Raedt, 2001b)

Entrada: uma consulta probabilística $? - Q_1, ..., Q_n | Ev_1 = ev_1, ..., Ev_m = ev_m$
Saída: uma rede suporte N relacionada a consulta probabilística

1: **para todo** variável $X_i \in \{Q_1, ..., Q_n, Ev_1, ..., Ev_m\}$ **faça**
2: computar todas as provas para X_i;
3: S:= cláusulas básicas (*ground clauses*) usadas para provar X_i;
4: N_i := representação de S como uma rede suporte;
5: aplicar *combining rules* a S
6: $N \leftarrow \cup_{i=1}^{k} N_i$;
7: $N \leftarrow poda(N)$;

O algoritmo Algorithm 2 recebe uma consulta $? - Q_1, ..., Q_n | Ev_1 = ev_1, ..., Ev_m = ev_m$ e retorna uma rede suporte N. Para cada variável de consulta ou evidência X_i, todas as suas possíveis provas são computadas e armazenadas em S (passo 1 a 3). Uma rede suporte representando essas provas é então gerada com a aplicação de alguma *combining rules* (NGO, HADDAWY, 1997; KERSTING, DE RAEDT, 2001c), caso necessário, para combinar várias instâncias de uma mesma cláusula ou várias cláusulas com um mesmo predicado cabeça (passos 4 e 5). A rede suporte resultante é a união das redes suportes encontradas para cada variável X_i, seguidas de uma poda, para excluir algumas arestas redundantes e/ou nós isolados, caso necessário.

A rede suporte exibida na Figura 4.1 é a rede suporte gerada a partir do BLP exibido na Figura 4.2, onde as CPDs são mostradas, e a consulta é

$$? - irmao(carlos, fred) | pais(mary, carlos) = V, pais(mary, fred) = V,$$
$$irmaos(carlos, fred) = V.$$

Podemos ver que as variáveis aleatórias na rede Bayesiana construída (veja Figura 4.1) são átomos básicos. Dessa forma, intuitivamente, *predicados probabilísticos* representam um conjunto de variáveis aleatórias e cada átomo probabilístico básico representa uma única variável aleatória.

$genero(carlos).$
$pais(mary, carlos).$
$pais(mary, fred).$
$irmao(X, Y) : -genero(X), irmaos(X, Y).$
$irmaos(X, Y) : -pais(Z, X), pais(Z, Y), X \backslash = Y.$

Figura 4.2: Exemplo de um BLP para o domínio da *Família*

4.1.2 Aprendizado dos Parâmetros Probabilísticos

Nesta seção descrevemos como as CPDs associadas as cláusulas probabilísticas do BLP são aprendidas. No restante deste capítulo mencionaremos redes Bayesianas ao invés de rede suporte, quando referenciando a rede gerada a partir de um determinado exemplo.

Como podemos ter *variáveis não observadas* na rede Bayesiana gerada, precisamos estimar as contagens esperadas. Além disso, as redes Bayesianas construídas para os diferentes exemplos não possuem a mesma estrutura e portanto as variáveis que não aparecem em uma determinada rede não devem interferir no cálculo da CPD para as cláusulas utilizadas naquela rede. Portanto, qualquer método utilizado para encontrar os parâmetros de um conjunto de redes deve ser adaptado para este caso. O que deve ser feito é olhar para as estruturas de rede construídas para cada exemplo, pois dessa maneira sabemos quais são as variáveis relevantes para cada contagem, e assim, exemplos que não as têm não influenciarão nesta contagem.

Propostas já foram feitas para o aprendizado das CPDs nestas situações, como em (KOLLER, PFEFFER, 1997), onde uma adaptação do algoritmo EM (DEMPSTER, LAIRD, et al., 1977) é utilizada e em (KERSTING, DE RAEDT, 2002), que apresenta duas abordagens, uma baseada no gradiente (BINDER, KOLLER, et al., 1997) e a outra baseada no EM. Nós utilizamos a última abordagem.

O objetivo então é encontrar os parâmetros que maximizam alguma função de avaliação probabilística, que no nosso caso será a log-verossimilhança. Sendo $c_1, ..., c_n$ as *cláusulas probabilísticas* do BLP e D o conjunto de redes Bayesianas geradas conforme o algoritmo 2, nós temos os parâmetros

$$cpd(c_i)_{jk} = P(u_j|\mathbf{u}_k)$$

onde $u_j \in$ domínio(cabeça(c_i)) e $\mathbf{u}_k \in$ domínio(corpo(c_i)) afetando a distribuição de probabilidade condicional $cpd(c_i)$ associada à cláusula c_i. Definimos o conjunto Λ como sendo o conjunto de todas as probabilidades para o BLP em questão:

$$\Lambda = \bigcup_{i=1}^{n} cpd(c_{i)}$$

Queremos encontrar o conjunto de parâmetros Λ^* que maximizem a verossimilhança:

$$\Lambda^* = max_{\Lambda} LL(D, \Lambda) = logP(D|\mathbf{\Lambda}) = logP(D) \tag{4.1}$$

A seguir, descrevemos o algoritmo EM adaptado, que é utilizado no nosso sistema de revisão PFORTE e consequentemente será o utilizado no sistema proposto PFORTE_PI.

Algoritmo Expectation-Maximization (EM) Adaptado

Na seção 2.2.2, vimos que a estimativa da máxima verossimilhança dos parâmetros das redes Bayesianas, quando o *dataset* é completamente observado, resume-se a contagem. Chamando então de $N(\mathbf{a}|\mathbf{D})$ as contagens para um conjunto particular $\mathbf{a}$ de valores para as variáveis $\mathbf{A}$ nos dados, ou seja, o número de casos em que a evidência $\mathbf{a}$ é associada às variáveis $\mathbf{A}$ nos dados, temos o seguinte cálculo para a CPD de uma cláusula no BLP:

$$cpd(c_i)^*_{jk} = \frac{N(cabeca(c_i\theta) = u_j, corpo(c_i\theta) = \mathbf{u}_k|d)}{N(corpo(c_i\theta) = \mathbf{u}_k|d)} \tag{4.2}$$

onde d $\in$ D.

Entretanto na presença de um *dataset* parcialmente observado, a estimação da máxima verossimilhança não pode ser escrita de forma fechada. Portanto é necessário o uso do algoritmo EM, onde as contagens esperadas (equação 4.3) serão usadas ao invés das contagens:

$$\sum_{l=1}^{m} P(cabeca(c_i\theta) = u_j, corpo(c_i\theta) = \mathbf{u}_k|d_l) \tag{4.3}$$

onde $m = |D|$.

Cada variável da rede Bayesiana d foi gerada por exatamente uma *cláusula probabilística* c, e cada variável derivada a partir de c pode ser vista como um "experimento

separado" para a distribuição de probabilidade condicional $cpd(c)$. Considerando esta observação a fórmula deduzida em (KERSTING, DE RAEDT, 2002) para calcular a CPD de uma cláusula c_i é a seguinte:

$$cpd(c_i)_{jk} \leftarrow \sum_{subst.\ \theta\ tal\ que\ c_i\theta \in a\ algum\ d_i \in D} \frac{\sum_{l=1}^{m} P(cabeca(c_i\theta) = u_j, corpo(c_i\theta) = \mathbf{u}_k | d_l)}{\sum_{l=1}^{m} P(corpo(c_i\theta) = \mathbf{u}_k | d_l)} \tag{4.4}$$

Ou seja, para cada rede Bayesiana d, se algum nó de d foi gerado a partir de uma substituição θ na cláusula c_i, este nó e seus pais contribuirão para o cálculo da CPD de c_i.

O critério de parada do EM adaptado é o mesmo do EM original, ou seja, considera-se um erro e o algoritmo para quando

$$LL(D, \Lambda_{i+1}) - LL(D, \Lambda_i) \leq \xi$$

Mais detalhes sobre BLPs podem ser encontrados na literatura(KERSTING, DE RAEDT, 2001b; KERSTING, DE RAEDT, 2001c; KERSTING, DE RAEDT, 2001a; KERSTING, DE RAEDT, 2002; KERSTING, DE RAEDT, 2005).

4.2 PFORTE: Revisão de Teorias Probabilísticas de Primeira-ordem através de Exemplos

Nesta seção, descrevemos o nosso sistema de revisão de teorias probabilísticas de primeira-ordem, denominado PFORTE (REVOREDO, ZAVERUCHA, 2002; PAES, REVOREDO, et al., 2005b; PAES, REVOREDO, et al., 2005a; PAES, REVOREDO, et al., 2006).

Assim como revisamos modelos representados através de lógica de primeira-ordem, também podemos revisar modelos representados em linguagens que utilizam lógica de primeira-ordem e probabilidade. Nesse caso novas questões surgem, visto que o modelo não é composto apenas pelo programa lógico mas também por distribuições de probabilidade que devem ser levadas em consideração quando são propostas alterações na estrutura. Similar a revisão de teorias de primeira-ordem, caso somente alguns pontos no modelo probabilístico de primeira-ordem estejam fazendo com que tal modelo deixe de predizer os valores corretos para os exemplos, ou seja

o modelo é aproximadamente correto, seria mais eficiente buscar por tais pontos e revisá-los do que descartar o modelo e aprendê-lo novamente a partir do zero ou propor modificações no modelo inteiro. Além disso, podemos alcançar um modelo final melhor tendo como ponto de partida o modelo inicial aproximadamente correto. Propomos então em trabalhos anteriores (REVOREDO, ZAVERUCHA, 2002; PAES, REVOREDO, et al., 2005b; PAES, REVOREDO, et al., 2005a; PAES, REVOREDO, et al., 2006) revisar teorias probabilísticas de primeira-ordem, onde estas são representadas por BLPs. Originalmente, PFORTE era executado em dois passos, onde o primeiro tinha como objetivo resolver problemas de provas lógicas e o segundo de resolver problemas de classificação probabilística. Agora, essas duas tarefas são agrupadas em um só passo, visto que problemas de prova também indicam problemas de classificação.

4.2.1 Definição do Problema

Podemos definir o problema como:

- **Dado:** uma teoria inicial incorreta e um conjunto de exemplos consistentes.
- **Encontre:** uma teoria "minimamente revisada"que é completa considerando os exemplos fornecidos e tem a melhor avaliação, dado alguma função de avaliação probabilística.

Nossa terminologia é definida como:

- **Teoria.** A teoria é um conjunto de *cláusulas probabilísticas*, como definido em 4.1.
- **Conceito.** Um conceito é um predicado, em uma teoria, para o qual exemplos aparecem em um conjunto de treinamento. São os predicados objetivo. No domínio da *Familia*, os conceitos devem incluir *irmão*, *pai*, e *esposa*.
- **Exemplo.** O esquema de um exemplo (E_i) é:

$$E_i = \{Instancias, Fatos, Evidencias\}$$

Uma *instância* é uma instanciação de um conceito. Por exemplo, uma instância do conceito *irmao* é *irmao(carlos,fred).* Uma instância deve ser derivada a partir da teoria incrementada com seus *fatos* associados. No domínio da *família*, os fatos definem uma família particular, por exemplo, $pais(mary, carlos)$, $pais(mary, fred)$, $genero(carlos)$. Dado o conjunto de instâncias derivadas, uma rede Bayesiana é construída como descrito na seção 4.1.1, ou seja o conjunto de instâncias derivadas junto com o conjunto de evidências formam uma *consulta probabilística*. Cada nó nestas redes Bayesianas é uma variável aleatória onde seu domínio vem dos seu *predicado probabilístico* correspondente. Uma *evidência* determina quais valores do domínio algumas dessas variáveis aleatórias assumem. Por exemplo, no domínio da *família* uma rede suporte pode ser construída a partir das instâncias provadas *irmao(carlos,fred)* (veja Figura 4.1) e o nó *pais(mary,carlos)* pode ser uma variável evidência. Dessa forma, o conjunto de evidências pode conter a evidência $pais(carlos, fred) = V$. Caso algum *predicado lógico* seja utilizado na derivação de uma instância ele não aparecerá na rede suporte. Já que estamos considerando aprendizado supervisionado, o *dataset* fornece o valor para as instâncias. Resumindo, no domínio da *família* podemos ter o exemplo:

$$E_i = \{\{irmao(carlos, fred) = verdadeiro\},$$
$$\{pais(mary, carlos), pais(mary, fred), genero(carlos)\},$$
$$\{pais(mary, carlos) = verdadeiro\}\}$$

Instâncias em um exemplo são mutuamente dependentes, mas diferentes exemplos são independentes uns dos outros.

- **Completude.** Dado um conjunto de exemplos E, onde $E_i \in E$ e $E_i = \{I, F, Ev\}$ dizemos que a teoria T é completa considerando estes exemplos se e somente se

$$\forall E_i \in E, \forall I_j \in I : T \cup F \vdash I_j$$

Derivação é definida usando resolução-SLD, considerando a instância como objetivo inicial.

- **Classificada corretamente.** Dado uma instância podemos dizer que esta instância foi classificada corretamente se o valor inferido para ela é o mesmo fornecido no *dataset*. Por exemplo, considere E_i definido no item anterior. Podemos dizer que a instância $irmao(carlos, fred)$ foi classificada corretamente, se usando um método de inferência na rede Bayesiana construída a partir dela considerando o conjunto de evidências, nós inferirmos maior probabilidade para o valor *verdadeiro*. Caso contrário, nós dizemos que esta instância foi *classificada incorretamente*.

- **Teoria "minimamente revista".** Uma teoria completa para um conjunto de exemplos pode ser produzida de forma trivial através da exclusão de todas as cláusulas existentes e introduzindo novas cláusulas que memorizam as instâncias. Entretanto, tal teoria não é de interesse. Idealmente, queremos que a teoria generalize para instâncias ainda não conhecidas. Já que a teoria inicial é assumida aproximadamente completa, uma teoria revista deve ser tão semântica e sintaticamente similar a ela quanto possível.

4.2.2 Algoritmo PFORTE

A teoria fornecida ao sistema de revisão pode ter sido obtida a partir das seguintes fontes:

- Um especialista do domínio forneceu a teoria inicial que não necessariamente reflete o *dataset* ou

- um sistema de ILP ou de SRL aprendeu uma teoria inicial considerando um *dataset* antigo e agora surgiram novos exemplos ou

- um sistema de ILP ou de SRL aprendeu uma teoria inicial usando o mesmo *dataset* que temos agora, mas essa teoria ainda pode ser melhorada.

Note que se um sistema de ILP foi usado para aprender a teoria inicial, as técnicas de revisão aqui apresentadas podem obter benefícios, pois levarão em consideração a incerteza/ruído dos dados, o que não foi feito pelo sistema de ILP.

Podemos obter vantagens no processo de busca ao ter como ponto de partida uma teoria inicial. Esse teoria é modificada pelo sistema de revisão na tentativa de produzir um modelo final melhor do que o fornecido.

O Algoritmo 3 descreve em alto nível o processo executado pelo sistema PFORTE. Esse algoritmo segue basicamente as seguintes idéias chave, em um processo iterativo:

1. Identificação dos *pontos de revisão*. Para tanto, os pontos falhos devem ser descobertos através de um mecanismo de raciocínio probabilístico.

2. Geração das propostas de revisão para os *pontos de revisão*. As revisões são propostas no modelo corrente usando *operadores de revisão*.

3. Pontuação das revisões propostas. Cada revisão recebe uma pontuação, calculada por uma função de avaliação probabilística.

4. Escolha da revisão a ser implementada. A revisão com a melhor pontuação é selecionada e se sua pontuação for melhor do que a pontuação corrente ela é implementada.

5. Critério de parada. Caso a melhor revisão não tenha uma pontuação melhor do que a pontuação corrente ou ainda se não existir nenhum ponto de falha no modelo, o algoritmo termina sua execução.

Para revisar uma teoria devem ser fornecidos a teoria inicial, um conhecimento preliminar e um *dataset*. É importante destacarmos que a teoria inicial pode estar vazia e assim o PFORTE será usado para aprender a partir "do zero". Nesse caso, PFORTE pode ser visto também como um sistema de aprendizado de teorias, que parte de cláusulas iniciais contendo apenas a cabeça e as modifica iterativamente para obter uma teoria final que reflita de forma consistente o conjunto de exemplos.

As seções seguintes descrevem em mais detalhes os pontos importantes do algoritmo PFORTE.

Algoritmo 3 Algoritmo PFORTE em alto nível

Entrada: Uma teoria inicial H; Um conhecimento preliminar BK; um conjunto de exemplos E
Saída: Uma teoria revista H'
1: $Score_H = score(H, E)$
2: $H' = H$
3: **repete**
4: Gera *pontos de revisão*
5: **para todo** pontos de revisão **faça**
6: **para todo** *operador de revisão* **faça**
7: $H'' =$ revisão de H' usando *operador de revisão*
8: $Score_{Revision} = score(H'', E)$
9: $Score_{Best} = max(Score_{Revision})$
10: $Revision_{Best} =$ revisão associada ao maior $Score_{Revision}$
11: **se** $Score_{Best} > Score_H$ **então**
12: $H' = Revision_{Best}$
13: **até** $Score_{Best} - Score_H < limite$ ou não existam *pontos de revisão* ou não existam revisões possíveis

4.2.3 Identificação dos Pontos de Revisão

O objetivo da busca por *pontos de revisão* é identificar os locais da estrutura que devem ser modificados para que a mesma reflita de uma maneira melhor o domínio sobre o qual os exemplos foram gerados. Os locais a serem modificados devem ser os pontos da teoria que contribuam para a *classificação incorreta* de algum exemplo, visto que o objetivo é melhorar a predição da teoria. Assim, espera-se que ao final tenhamos uma teoria que consiga predizer melhor o valor das consultas. Para tanto é preciso identificar os exemplos que estão sendo classificados incorretamente e a seguir os pontos da teoria que estão contribuindo para a classificação incorreta de tais exemplos.

A escolha das cláusulas probabilísticas a serem modificadas deve seguir um critério de predição que considere as distribuições de probabilidades e dependências condicionais entre os átomos. Logo, para descobrir os pontos falhos da teoria são usadas as redes Bayesianas construídas a partir dos exemplos. Usando essas redes Bayesianas, os valores das variáveis de consulta são inferidos e comparados com os valores fornecidos com os exemplos. Se o valor inferido for diferente do valor associado à variável de consulta, a *Cobertura de Markov* dessa variável aleatória é coletada. A *Cobertura de Markov* de uma variável aleatória é composta pela própria

variável, seus pais, filhos e pais dos filhos. Porém, caso algumas das variáveis na *Cobertura de Markov* sejam não-observadas esse conjunto se expande, englobando também a *Cobertura de Markov* da *variável não-observada*, e assim sucessivamente. A *Cobertura de Markov* é usada porque queremos aumentar a taxa de acerto das redes ao predizerem os valores das variáveis de consulta. Como uma suposição chave em redes Bayesianas é que uma variável aleatória é condicionalmente independente das demais variáveis da rede dada a sua *Cobertura de Markov*, o conjunto coletado englobará todas as variáveis da rede que podem ter influenciado na classificação incorreta da variável de consulta. As cláusulas probabilísticas cujas cabeças instânciadas fazem parte desse conjunto são marcadas como *pontos de revisão*, ou seja, cláusulas que serão modificadas através dos *operadores de revisão*.

Caso existam instâncias não provadas, não será possível inferir o seu valor através de uma rede Bayesiana, pois a mesma é construída a partir da prova da instância usando KBMC (HADDAWY, 1999). Nesse caso, é introduzido um nó *a priori* na rede para representar a instância não provada. As cláusulas que poderiam provar tais exemplos são consideradas como *pontos de revisão*. Para descobrir tais cláusulas é usado o mesmo procedimento de descoberta de *pontos de revisão* de generalização do sistema FORTE.

O procedimento de coleta de *pontos de revisão* em uma teoria está descrito no Algoritmo 4.

Tendo encontrado o conjunto de *pontos de revisão* o próximo passo é aplicar os *operadores de revisão* à estes pontos.

4.2.4 Operadores de Revisão

Para propor modificações nos *pontos de revisão* são usados quatro *operadores de revisão*. Todos os operadores podem ser aplicados em qualquer uma das cláusulas da teoria marcadas como *pontos de revisão*, diferindo do FORTE onde os operadores eram divididos em dois grupos, operadores de generalização e de especialização. A justificativa é que as modificações executadas por todos os operadores podem contribuir para uma melhora na classificação das variáveis, pois estamos empregando um mecanismo de raciocínio probabilístico.

Algoritmo 4 Algoritmo de geração de *pontos de revisão* em uma teoria

Entrada: Uma teoria H; Um conhecimento preliminar BK; um conjunto de exemplos E
Saída: Um conjunto de *pontos de revisão* RP

```
1: MB ← ∅
2: se existirem instâncias não provadas então
3:    RP ← cláusulas que falharam em provar alguma dessas instâncias
4: senão
5:    RP ← ∅
6: para cada exemplo E_i ∈ E faça
7:    d ← rede Bayesiana construída a partir de E_i considerando H e BK
8:    para cada variável aleatória v em d que seja uma instância em E_i faça
9:       valueNode ← valor de v obtido pelo mecanismo de inferência probabilística
10:      valueExample ← valor da instância em E_i representado por v
11:      se ∃ valueExample e valueExample ≠ valueNode então
12:         MB_{E_i} ← variáveis aleatórias que fazem parte da Cobertura de Markov de v
13:         para cada variável aleatória v_mb ∈ MB_{E_i} faça
14:            se v_mb é uma variável não-observada então
15:               MB_{E_i} ← MB_{E_i} ∪ variáveis aleatórias que fazem parte Cobertura de
                  Markov de v_mb
16:         MB ← MB ∪ MB_{E_i}
17: para cada cláusula probabilística c ∈ H faça
18:    se head(c) unifica com alguma variável em MB então
19:       RP ← RP ∪ c
    RETURN RP
```

Após os operadores proporem possíveis revisões em todos os *pontos de revisão*, a proposta de modificação com a maior pontuação é escolhida e caso sua pontuação seja maior do que a pontuação corrente, a mesma é incorporada na teoria sendo revisada. A seguir os *operadores de revisão* são detalhados.

Operador de adição de antecedentes Literais criados a partir da base de conhecimento são adicionados ao corpo da cláusula marcada como *ponto de revisão*. Esse operador funciona de forma semelhante ao operador de adição de antecedentes do FORTE, diferindo principalmente em três aspectos:

1. Os antecedentes são avaliados usando uma função de avaliação probabilística;

2. os exemplos não são vistos como negativos, no sentido de que não devem ser explicados, visto que queremos construir a estrutura probabilística que explique todos os exemplos e isso é feito através de suas provas. Assim, os antecedentes não são adicionados na cláusula com o objetivo de deixar de

provar exemplos negativos, mas sim de melhorar o valor da função de avaliação probabilística;

3. o retorno do operador é a especialização de apenas uma cláusula, diferente do sistema FORTE que pode retornar mais de uma regra especializada a partir de uma única cláusula base.

Assim, o operador de adição de antecedentes do PFORTE executa as seguintes operações:

1. Tenta encontrar um caminho entre as variáveis da cláusula usando o algoritmo *Relational Pathfinding*. Cada caminho encontrado é avaliado de acordo com uma função de avaliação probabilística e o melhor caminho é selecionado e adicionado à cláusula, se sua pontuação for melhor do que a pontuação da cláusula base. Nesse caso, a cláusula juntamente com o caminho encontrado passa a ser a cláusula corrente; caso contrário, a cláusula retornada por esse algoritmo é a mesma cláusula passada como entrada;

2. tenta adicionar na cláusula um antecedente de cada vez usando um processo iterativo *hill-climbing*.

Os passos executados pelo operador de adição de antecedentes estão descritos no Algoritmo 5.

Operador de exclusão de antecedentes Dois algoritmos podem ser usados para excluir antecedentes de uma regra. O primeiro exclui antecedentes em um processo iterativo *hill-climbing*, onde a cada iteração a cláusula é avaliada sem cada um dos seus antecedentes e aquela exclusão que tiver a maior pontuação é escolhida. O antecedente é de fato removido da cláusula se essa pontuação for melhor do que a pontuação corrente. Uma outra forma de excluir antecedentes é usar um procedimento que seleciona mais de um antecedente para exclusão de uma única vez. Esse processo é muito custoso, visto que a avaliação da melhor exclusão deve levar em consideração uma combinação de antecedentes da cláusula. Devido a esse fator de complexidade, essa última forma de excluir antecedentes só é executada

Algoritmo 5 Algoritmo de adição de antecedentes ao propor revisões em uma teoria

Entrada: Uma teoria H; Um conhecimento preliminar BK; um conjunto de exemplos E; Uma cláusula c
Saída: Uma cláusula c' contendo zero ou mais antecedentes que c

$c'' \leftarrow ProbRelationalPathfinding(c, H, BK, E)$
se $c'' \neq c$ **então**
    $H' \leftarrow H$ com c substituído por c''
    $c' \leftarrow c''$
senão
    $H' \leftarrow H$
    $c' \leftarrow c$
$score \leftarrow score(H', E)$
repete
    $antes \leftarrow$ antecedentes gerados a partir de H' e da base de conhecimento
    se $antes = \emptyset$ **então**
        saia do laço
    para cada $antecedent \in antes$ **faça**
        $c'' \leftarrow c' \cup antecedent$
        $H'' \leftarrow H$ com c substituído por c''
        $score_{antecedent} \leftarrow score(H'', E)$
    $bestScore \leftarrow max(score_{antecedent})$
    $bestAntecedent \leftarrow antecedent$ com $bestScore$
    se $bestScore > score$ **então**
        $c' \leftarrow c' \cup bestAntecedent$
até $bestScore - score < valorLimite$
RETURN c'

quando o processo iterativo *hill-climbing* não consegue obter nenhuma melhora na pontuação.

Operador de exclusão de regras Esse operador simplesmente exclui a cláusula probabilística marcada como *ponto de revisão*. A exclusão só não é permitida quando a cláusula é a única na teoria cuja cabeça unifica com alguma instância.

Operador de adição de regras Cláusulas podem ser adicionadas na teoria através de dois processos:

1. criação de uma regra a partir de outra existente - esse processo parte de uma cláusula marcada como *ponto de revisão*, exclui antecedentes da mesma usando o operador de exclusão de antecedentes e após adiciona antecedentes na mesma, usando o operador de adição de antecedentes;

2. criação de uma regra inteiramente nova - uma regra nova é criada usando a instância (variabilizada) incorretamente classificada como cabeça da regra. O corpo da cláusula é criado através do operador de adição de antecedentes.

A adição de regras acrescenta novas regras na teoria sem retirar as que apresentaram problemas, diferente dos demais operadores que substituem as cláusulas que apresentaram problemas.

4.2.5 Cálculo do Valor da Função de Avaliação Probabilística

Para avaliar uma teoria é necessário utilizar uma função que leve em consideração não somente a estrutura da mesma, mas também as distribuições de probabilidades associadas às cláusulas. Para tanto, as estruturas de raciocínio probabilístico, no nosso caso redes Bayesianas, devem ser construídas a partir dos exemplos fornecidos, e as CPDs associadas às cláusulas devem ser aprendidas. O procedimento de cálculo da avaliação está exibido no Algoritmo 6.

Algoritmo 6 Algoritmo de cálculo de pontuação de uma teoria no PFORTE 2.0

Entrada: Uma teoria H; Um conhecimento preliminar BK; um conjunto de exemplos E
Saída: Uma avaliação probabilística $Score$
1: $D \leftarrow$ rede Bayesiana construída a partir de E, H, BK
2: λ parâmetros de probabilidade aprendidos para as cláusulas de H usando D
3: $Score$ pontuação de H calculada usando D e λ
4: return $Score$

Qualquer função de avaliação probabilística calculada sobre modelos gráficos probabilísticos pode ser utilizada. As funções de avaliação definidas na seção 2.3 estão atualmente implementadas no PFORTE.

Capítulo 5

DAHVI: Sistema de Revisão de Redes Bayesianas com Introdução de Variáveis Não-observadas

Modelos probabilísticos gráficos tem sido bastante usados para modelar domínios reais. Considerando as pesquisas extensivas feitas sobre o aprendizado destes modelos a partir de *datasets* (PEARL, 1988)(HECKERMAN, 1998), aprender com *variáveis não-observadas* continua sendo um desafio no aprendizado de modelos gráficos em geral e em particular de redes Bayesianas. Entidades escondidas são essenciais em muitos problemas que modelam o mundo real: (i) um mecanismo desconhecido de regulagem pode ser a chave para sistemas biológicos complexos; (ii) correlacionamento de sintomas podem sugerir problemas fundamentais, através de uma entidade escondida, em um sistema de diagnóstico; (iii) um poder econômico intencionalmente mascarado pode ser a causa de fenômenos financeiros relacionados. Certamente, *variáveis não-observadas* servem tipicamente como um mecanismo de síntese, que captura informações de algumas variáveis observadas e passa estas informações para alguma outra parte da rede. Dessa forma, *variáveis não-observadas* podem simplificar a estrutura da rede e conseqüentemente levar à uma generalização melhor.

Considere por exemplo, a rede exibida na figura 5.1, utilizada para estimar o risco de se efetuar um seguro para um determinado carro (essa mesma rede é exibida em (BINDER, KOLLER, et al., 1997), onde os nós são rotulados com nomes). Ela contém 27 nós e 51 arestas. Dado o domínio das variáveis, existem 1421 parâmetros à serem aprendidos. Se retirarmos, por exemplo, a *variável não-observada Accident*

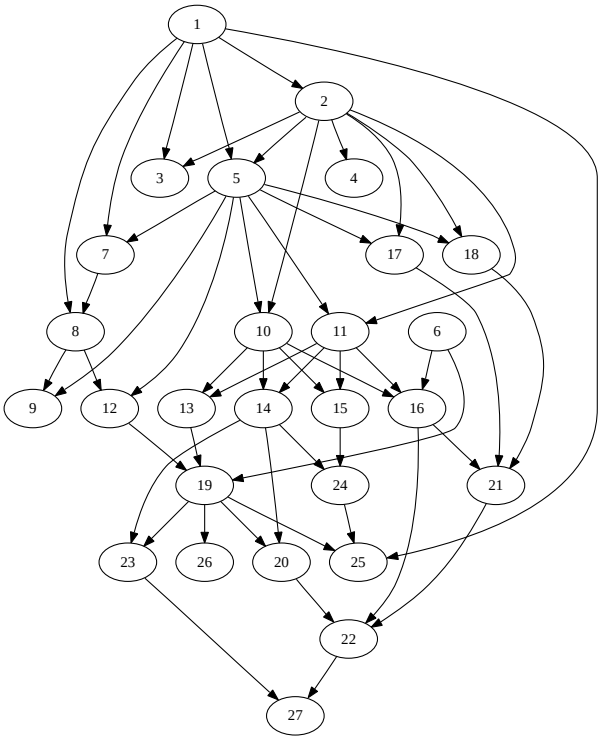


Figura 5.1: Rede Bayesiana modelando o risco de se efetuar um seguro de carro considerando a *variável não-observada accident*

(nó 19) da rede, conectando os seus pais aos seus filhos (Rede Bayesiana exibida na Figura 5.2), a rede passaria a ter 1501 parâmetros para serem aprendidos, o que representa um aumento de 5.7% no número de parâmetros e de 25% no tempo de cálculo da acurácia.

Aprender os parâmetros de uma rede Bayesiana considerando um *dataset* parcialmente observado é custoso e tem sido pesquisado extensivamente, sendo algoritmos, tais como, EM (**E***xpectation* **M***aximization*) (DEMPSTER, LAIRD, et al., 1977; LAURITZEN, 1995) ou gradiente ascendente (BINDER, KOLLER, et al., 1997) padrões na area. A tarefa de aprender tanto os parâmetros quanto a estrutura da rede é muito mais custosa, já que para cada rede proposta os parâmetros precisam ser re-aprendidos. O algoritmo *Structural EM* (SEM) (FRIEDMAN, 1997) (FRIEDMAN, 1998) foi proposto de forma a reduzir esse custo. A idéia central desse algoritmo é efetuar um único passo EM para as novas estruturas propostas. Como no caso do EM, a convergência é garantida, mas o algoritmo tipicamente converge para um máximo local. Apesar disso, o SEM é mais vantajoso do que aplicar EM

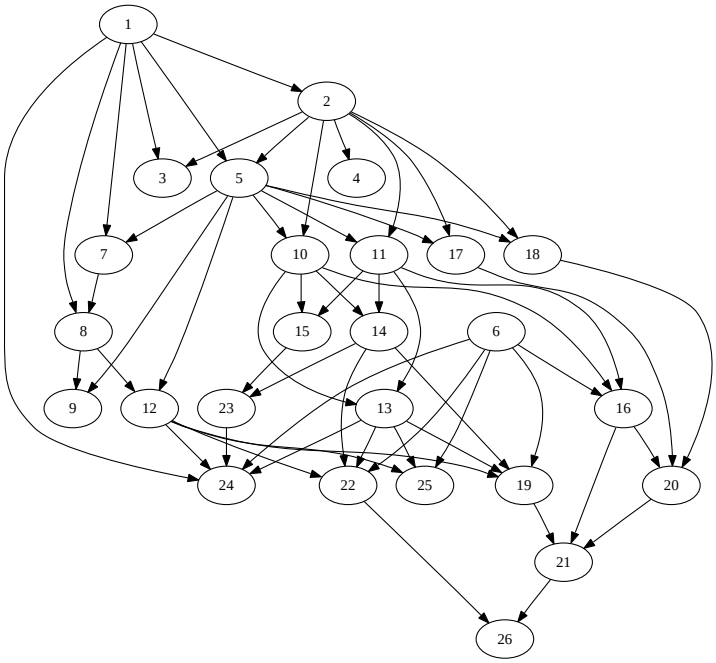


Figura 5.2: Rede Bayesiana modelando o risco de se efetuar um seguro de carro não considerando a *variável não-observada accident*

para cada nova estrutura proposta.

Uma questão, ainda mais complicada e desafiadora é aprender uma rede Bayesiana identificando novas *variáveis não-observadas*. Isto involve escolher o número de *variáveis não-observadas*, suas cardinalidades e a dependência entre elas e as outras entidades do domínio. Estas decisões são cruciais para encontrar uma boa generalização. Em particular, o *SEM* é capaz de introduzir novas *variáveis não-observadas*, mas ele não atinge bons resultados, a não ser que algum conhecimento a priori da localização das *variáveis não-observadas* e da cardinalidade delas seja conhecida (pelo menos aproximada). Se não forem fornecidas tais informações, outras abordagens existem. O algoritmo *Semi-clique* (ELIDAN, LOTNER, et al., 2001), por exemplo, baseia-se na observação que geralmente *variáveis não-observadas* podem simplificar segmentos da estrutura da rede que estão altamente conectados. Alternativamente, pode-se tirar vantagem da topologia da rede: BANNER (RAMACHANDRAN, MOONEY, 1998) introduz *variáveis não-observadas* em uma rede Bayesiana *noisy-or* e *noisy-and*.

Nesta tese, introduzimos uma nova abordagem para aprender redes Bayesia-

nas com *variáveis não-observadas*, baseada em revisão de teoria (WROBEL, 1996). Como visto na seção 3.2, revisão de teoria tem o objetivo de aumentar a performance de uma determinada teoria, modificando a sua estrutura de forma que esta reflita melhor o *dataset*. Ao invés de propor modificações em toda a estrutura, algoritmos de revisão de teoria utilizam os exemplos, que não foram inferidos corretamente (D_{mis}), para identificar pontos potenciais na teoria a serem modificados, chamados *pontos de revisão*. *Operadores de revisão*, tais como adição de antecedentes ou adição de regras, são então utilizados para propor estas modificações. Baseado nesta idéia, propomos um algoritmo de revisão de redes Bayesianas, denominado **DAHVI** (**D***iscriminative* **A***pproach for* **H***idden* **V***ariable* **I***ntroduction*) (REVOREDO, PAES, et al., 2009), que utiliza uma abordagem discriminativa para selecionar *pontos de revisão*: *variáveis de consulta* e suas *Coberturas de Markov* indicam pontos potenciais para a introdução de *variáveis não-observadas*, já que as variáveis que influenciam o valor inferido para as *variáveis de consulta* pertencem à *Cobertura de Markov*. Em seguida, um *operador de revisão*, definido a seguir na seção 5.3, que introduz *variáveis não-observadas* nestes *pontos de revisão*, é utilizado . A motivação para o uso desta abordagem é a seguinte. Se após o processo de aprendizado, a rede Bayesiana resultante não foi capaz de inferir o valor correto de pelo menos uma *variável de consulta*, considerando um determinado exemplo, então é possível que exista alguma informação "escondida"que não esteja sendo descrita corretamente pela estrutura de rede.

Nas seções seguintes deste capítulo, os procedimentos de seleção dos *pontos de revisão*, o nosso *operador de revisão* e o algoritmo de revisão DAHVI são definidos.

5.1 Seleção dos Pontos de Revisão

Uma parte importante do algoritmo de revisão proposto nesta tese é a definição dos *pontos de revisão*, que representaremos por MB^*. Nós consideramos uma abordagem discriminativa, onde as *variáveis de consulta* ($\mathbf{Y}$) e as variáveis pertencentes às *Coberturas de Markov* correspondentes ($MB^* = Y \cup coberturaMarkov(Y)$) são consideradas indicadores potenciais da localização de *variáveis não-observadas*, já que as variáveis que influenciam o valor inferido para as *variáveis de consulta* são

as pertencentes a sua *Cobertura de Markov*. Definimos então da seguinte forma os *pontos de revisão* (MB^*):

Definição 5.1.1 *Se o conjunto de exemplos inferidos incorretamente, D_{mis}, é completamente observado, então*

$$MB^* = Y \cup coberturaMarkov(Y)$$

Caso contrário,

$$MB^* = \bigcup_{d_{mis_i} \in D_{mis}} MB^*_{d_{mis_i}}$$

onde $MB^*_{d_{mis_i}}$ ($MB^*_{d_{mis_i}} = MB^*_n$) é definido de forma recursiva a partir de

$$MB^*_0 = Y \cup coberturaMarkov(Y)$$

considerando o exemplo d_{mis_i}:

$$MB^*_n = MB^*_{n-1} \bigcup coberturaMarkov(Var_{mis}(MB^*_{n-1}))$$

onde $Var_{mis}(MB^*_{n-1})$ é o conjunto de variáveis com valores não observados em d_{mis_i} considerando a iteração anterior.

A segunda parte da definição é dependente dos exemplos, já que as variáveis que tem informação perdida podem variar de exemplo para exemplo. Para cada exemplo inferido incorretamente (d_{mis_i}), $MB^*_{d_{mis_i}}$ é definido de forma recursiva. A cada chamada recursiva, $MB^*_{d_{mis_i}}$ é acrescido da *Cobertura de Markov* de variáveis cujo os valores não foram observados no exemplo d_{mis_i} e que tenham sido inseridas em $MB^*_{d_{mis_i}}$ na iteração anterior. A recursão é finalizada quando não forem identificadas mais variáveis cujo os valores não foram observados no exemplo d_{mis_i} na iteração anterior. Para ilustrar esta definição, considere a rede Bayesiana exibida na Figura 5.3 (a) e a Tabela 5.1, onde são fornecidos 4 exemplos, cujo o valor para a *variável de consulta* Y não foi inferido corretamente em nenhum dos 4 exemplos para esta rede. O respectivo $MB^*_{d_{mis_i}}$ é exibido para cada exemplo.

O exemplo d_{mis_1} exibido na Tabela 5.1, por exemplo, fornece valor para todas as variáveis da rede, logo é totalmente observado, recaindo na primeira parte da definição 5.1.1. Com isso, os *pontos de revisão* são a *variável de consulta* Y e a

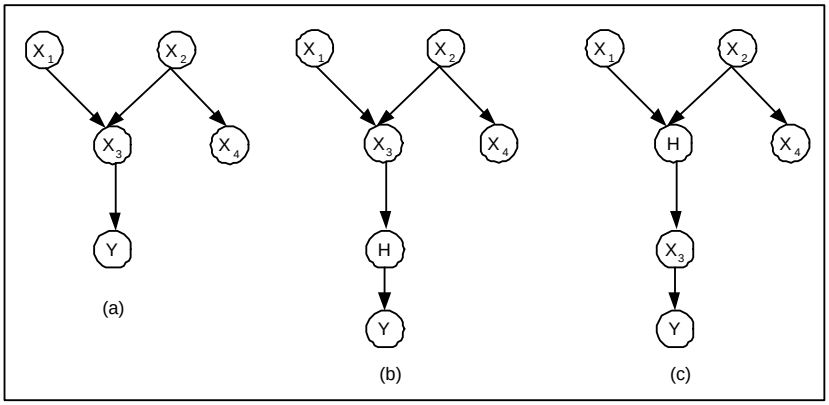


Figura 5.3: (a) Uma estrutura de rede Bayesiana, onde Y é a *variável de consulta*. (b) Uma estrutura de rede Bayesiana candidata considerando o nó $Y \in MB^*$ (c) Uma estrutura de rede Bayesiana candidata considerando o nó $X_3 \in MB^*$

Tabela 5.1: Exemplos inferidos incorretamente, a partir da rede Bayesiana da Figura 5.3, com seus respectivos $MB^*_{d_{mis_i}}$

$d_{mis_1} = [y_1, x_{3_1}, x_{1_1}, x_{2_1}, x_{4_1}]$	$\longrightarrow$	$MB^*_{d_{mis_1}} = \{Y, X_3\}$
$d_{mis_2} = [y_2, x_{1_2}, x_{2_2}, x_{4_2}]$	$\longrightarrow$	$MB^*_{d_{mis_2}} = \{Y, X_3, X_1, X_2\}$
$d_{mis_3} = [y_3, x_{1_3}, x_{4_3}]$	$\longrightarrow$	$MB^*_{d_{mis_3}} = \{Y, X_3, X_1, X_2, X_4\}$
$d_{mis_4} = [y_4, x_{3_4}, x_{1_4}, x_{4_4}]$	$\longrightarrow$	$MB^*_{d_{mis_4}} = \{Y, X_3\}$

sua *Cobertura de Markov* composta pela variável X_3, definindo $MB^*_{d_{mis_1}} = \{Y, X_3\}$, como exibido na Tabela 5.1. O exemplo d_{mis_4}, apesar de ser parcialmente observado, pois não tem informações a respeito de X_2, também vai retornar $MB^*_{d_{mis_4}} = \{Y, X_3\}$, já que todas as variáveis pertencentes ao conjunto encontrado inicialmente ($MB^*_0 = Y \cup coberturaMarkov(Y) = \{Y, X_3\}$) têm informação e com isso não é necessário buscar pelas *Coberturas de Markov* correspondentes. Já os exemplos d_{mis_2} e d_{mis_3} são parcialmente observados e variáveis com informações perdidas são consideradas em MB^*_n, para algum $n \in N$, logo é preciso buscar a *Cobertura de Markov* destas variáveis. Em d_{mis_2} buscamos por $coberturaMarkov(X_3)$ e em d_{mis_3} por $coberturaMarkov(X_3)$ e $coberturaMarkov(X_2)$. O conjunto final dos *pontos de revisão* é a união dos $MB^*_{d_{mis_i}}$, ou seja

$$MB^* = \{Y, X_3, X_1, X_2, X_4\}$$

Note que no pior caso, como neste exemplo, o conjunto de *pontos de revisão* inclui todas as variáveis. Este seria o caso para redes Bayesianas construídas por *Naive Bayes* e por TAN (*Tree-augmented network*) (FRIEDMAN, GEIGER, et al., 1997), como em ambos a *Cobertura de Markov* da *variável de consulta* inclui todas as variáveis. O nosso algoritmo funciona melhor com redes menos conectadas, construídas por algoritmos como K2 ou SEM (FRIEDMAN, 1998).

No Algoritmo 7 descrevemos o procedimento utilizado para selecionar os *pontos de revisão*, MB^*, no caso do conjunto de exemplos ser parcialmente observado.

Algoritmo 7 Algoritmo para definir os *pontos de revisão*: $\mathbf{meuMB(RB, d_{mis}, X)}$

Entrada: rede Bayesiana (RB); exemplo cujo valor da *variável de consulta* foi inferido incorretamente (d_{mis}); variável (X) para a qual será encontrada a *Cobertura de Markov*
Saída: conjunto de *pontos de revisão* (MB^*)

1: $MB^* = coberturaMarkov(X)$;
2: Var_{mis} = variáveis pertencentes a MB^* com valores não observados em d_{mis};
3: **para todo** variável $var_j \in Var_{mis}$ **faça**
4: $\quad MB^* = MB^* \cup meuMB(RB, d_{mis}, var_j)$;

O Algoritmo 7 recebe a rede Bayesiana que está sendo revisada, um dos exemplos onde o valor da *variável de consulta* foi inferido incorretamente e a variável X para a qual deseja-se encontrar a *Cobertura de Markov* MB^*. O algoritmo começa encontrando a *Cobertura de Markov* da variável X (passo 1), em seguida é definido o conjunto Var_{mis} das variáveis pertencentes a MB^* que não tiveram o seu valor observado no exemplo d_{mis} (passo 2). Para cada variável pertencente ao conjunto Var_{mis}, o algoritmo $meuMB$ é chamado para que a *Cobertura de Markov* desta seja encontrada (passos 3 e 4). Caso o conjunto Var_{mis} seja vazio, o algoritmo termina. A união então de todas as *coberturas de Markov* é retornada pelo algoritmo.

5.2 Operador de Revisão

O conjunto de *pontos de revisão* selecionado indica pontos potenciais para modificações. Nesta tese, propomos modificar esses *pontos de revisão* com um *operador de revisão* baseado em introdução de *variáveis não-observadas*. Este *operador de revisão*, define uma estrutura candidata com uma *variável não-observada* intermediando uma *ponto de revisão* e seus pais. Cada *variável não-observada* é introduzida

considerando cardinalidade 2, ou seja um domínio binário, já que assim aumentamos o número de parâmetros probabilísticos o mínimo possível. Em seguida, as CPDs são aprendidas, utilizando o algoritmo EM, e a rede Bayesiana candidata é avaliada utilizando uma função de avaliação probabilística. Dessa forma, o *operador de revisão* proposto retorna uma rede Bayesiana candidata com uma *variável não-observada* e sua avaliação.

Na seção 5.1, nós encontramos o conjunto de *pontos de revisão*

$$MB^* = \{Y, X_3, X_1, X_2, X_4\}$$

para a rede Bayesiana da Figura 5.3 (a). Logo, cinco *pontos de revisão* foram selecionados. Para cada um deles, o *operador de revisão* descrito aqui propõe uma rede Bayesiana candidata com uma *variável não-observada* incluída. Por exemplo, na Figura 5.3 (b) e (c) uma estrutura candidata para os *pontos de revisão* Y e X_3 respectivamente é exibida: na primeira, a *variável não-observada* H é introduzida intermediando a *variável de consulta* Y e seus pais ((b)) e no segundo exemplo H é introduzida entre X_3 e seus pais ((c)). Apesar de apenas acrescentar uma *variável não-observada* intermediária entre 2 variáveis, a avaliação das redes candidatas podem ser alteradas para melhor, já que as CPDs são outras. Além disso, dependendo do tamanho do domínio das variáveis envolvidas, ao acrescentar a *variável não-observada* reduz-se o número de parâmetros probabilísticos, permitindo uma inferência e aprendizado dos parâmetros melhor. Considere por exemplo que as variáveis Y e X_3 tenham um domínio de tamanho 6 e 5 respectivamente. A CPD associada a variável Y tem 30 parâmetros probabilísticos. Quando a *variável não-observada* H, com domínio binário, é inserida a CPD passa a ter então 12 parâmetros probabilísticos e a CPD associada a variável H 10. Dessa forma, temos uma redução de 8 parâmetros probabilísticos.

Agora que definimos o procedimento para determinação das variáveis que serão utilizadas para indicar aonde *variáveis não-observadas* podem ser incluídas (*pontos de revisão*) e descrevemos como as *variáveis não-observadas* são introduzidas (*operador de revisão*), apresentamos na seção seguinte o nosso algoritmo de revisão de redes Bayesianas, DAHVI.

5.3 Algoritmo de Revisão DAHVI

Nesta seção detalhamos o nosso algoritmo de revisão DAHVI.

O algoritmo DAHVI exibido em 8, recebe uma rede Bayesiana (RB), uma função de avaliação probabilística (*score*), as *variáveis de consulta* ($\mathbf{Y}$) e o *dataset* ($\mathbf{C}$). Quando o *dataset* considerado é de classificação, as *variáveis de classe* são consideradas as *variáveis de consulta* utilizadas pelo DAHVI para selecionar os *pontos de revisão*. Para *datasets* mais gerais, ou seja *datasets* que não são de classificação, o usuário define quais variáveis são mais adequadas para guiarem esta seleção. DAHVI começa avaliando a rede Bayesiana RB, recebida como entrada, e assume que RB é a melhor rede até o momento através da inicialização da variável RB_h, que será retornada ao final da execução do algoritmo (passos 1 e 2). Em seguida, no passo 4, os exemplos, onde pelo menos uma das *variáveis de consulta* não teve o seu valor inferido corretamente, são reunidos na variável D_{mis}. Caso D_{mis} seja vazio, o algoritmo termina já que não existe necessidade em revisa-lo (passos 5 e 6). Caso contrário, o algoritmo prossegue definindo os *pontos de revisão*. Se D_{mis} é totalmente observado, o conjunto de *pontos de revisão* é formado pelas *variáveis de consulta* e as *Coberturas de Markov* correspondentes (passos 7 ao 10). Já se D_{mis} for parcialmente observado, a *Cobertura de Markov* é calculada para cada exemplo $d_{mis_i} \in D_{mis}$ e cada *variável de consulta* $Y \in \mathbf{Y}$, utilizando o algoritmo **meuMB** exibido em Algoritmo 7 visto anteriormente (passos 11 a 13). Depois de definido o conjunto de *pontos de revisão* MB^*, o nosso *operador de revisão* é aplicado em cada *ponto de revisão*, como definido em 5.2 (passos 16 a 22). Se a melhor rede Bayesiana candidata melhorar a rede Bayesiana corrente, a rede Bayesiana candidata é implementada e o algoritmo prossegue (passos 23 a 24). O algoritmo é subida de encosta (*hill-climbing*) e sendo assim é executado enquanto a estrutura de rede puder ser melhorada. Ao final a rede RB_h é retornada.

O algoritmo DAHVI inclui quantas *variáveis não-observadas* quantas forem necessárias para melhorar a avaliação da rede Bayesiana. Diferentemente do algoritmo SEM, onde é necessário indicar a priori quantas são as *variáveis não-observadas* DAHVI descobre quantas são executando o algoritmo. Além disso o

Algoritmo 8 DAHVI Algoritmo

Entrada: rede Bayesiana (RB); função de avaliação probabilística; variáveis de consulta ($\mathbf{Y}$); *dataset* ($\mathbf{C}$);

Saída: uma rede Bayesiana com *variáveis não-observadas* (RB_h)

1: $RB_h = RB$;
2: **repete**
3: $Score_h = score(RB_h, \mathbf{C})$;
{Verifica se existem exemplos inferidos incorretamente}
4: D_{mis} = exemplos inferidos incorretamente;
5: **se** $D_{mis} = \varnothing$ **então**
6: break;
{Seleciona os pontos de revisão MB^*}
7: $MB^* = \mathbf{Y}$;
8: **se** D_{mis} totalmente observado **então**
9: $MB^* = MB^* \bigcup_{Y \in \mathbf{Y}} coberturaMarkov(Y)$;
10: **senão**
11: **para todo** exemplo $d_{mis_i} \in D_{mis}$ **faça**
12: **para todo** $Y \in \mathbf{Y}$ **faça**
13: $MB^* = MB^* \cup meuMB(RB, d_{mis_i}, \mathbf{Y})$;
14: $RB_{melhor} = RB_h$;
15: $Score_{melhor} = Score_h$;
{aplica o operador de revisão para cada ponto de revisão}
16: **para todo cada** variável $N_i \in MB^*$ **faça**
17: RB_{c_i} = estrutura candidata com uma *variável não-observado* intermediando N_i e seus pais;
18: aprende as CPDs de RB_{c_i};
19: $Score_i = score(RB_{c_i}, \mathbf{C})$;
{mantém a melhor estrutura candidata encontrada}
20: **se** $Score_i > Score_{melhor}$ **então**
21: $RB_{melhor} = RB_{c_i}$;
22: $Score_{melhor} = Score_i$;
{Implementa a melhor proposta caso ela forneça alguma melhora na rede Bayesiana corrente}
23: **se** $Score_{melhor} > Score_h$ **então**
24: $RB_h = RB_{melhor}$;
25: **até** a avaliação convergir

algoritmo DAHVI pode ser aplicado até mesmo em redes esparsas.

No capítulo 6 descrevemos os resultados obtidos com a aplicação do sistema DAHVI a 13 *datasets*.

5.4 Trabalhos Relacionados

Nesta seção descrevemos dois algoritmos relacionados com a nossa proposta para inclusão de *variáveis não-observadas* em uma rede Bayesiana. O primeiro, que denominaremos *Semi-clique* (ELIDAN, LOTNER, et al., 2001), relaciona-se com o algoritmo DAHVI por também ter como objetivo a detecção do melhor lugar para introduzir uma *variável não-observada*, restringindo assim o espaço de busca das possibilidades de modificação na estrutura da rede. Já o segundo, denominado BANNER (RAMACHANDRAN, MOONEY, 1998), relaciona-se com o DAHVI por também utilizar técnicas de revisão de teoria para melhorar uma rede Bayesiana, sendo uma das modificações propostas a inclusão de uma *variável não-observada*. Abaixo descrevemos esses dois algoritmos em mais detalhes.

5.4.1 Semi-clique

Na seção 5, verificamos que o algoritmo SEM é capaz de inserir *variáveis não-observadas*, mas ele não obtém bons resultados se um conhecimento preliminar a respeito da localização destas variáveis não for fornecido. Assim como, é preciso definir como entrada do algoritmo a quantidade de *variáveis não-observadas*. Em (ELIDAN, LOTNER, et al., 2001) foi proposto uma abordagem que procurava por indicações na estrutura da rede da necessidade de inclusão de uma *variável não-observada*, isto é, por sub-estruturas da rede aprendida que sugiram a presença de uma *variável não-observada*. Este algoritmo pode ser visto como uma instanciação concreta e eficiente do método mencionado em (HECKERMAN, 1995). As sub-estruturas consideradas foram denominadas *semi-cliques*:

Definição 5.4.1 *Para uma variável X e um conjunto de variáveis $\mathbf{Y}$, $\Delta(X;\mathbf{Y})$ é o conjunto de vizinhos de X (pais ou filhos) pertencentes ao subconjunto $\mathbf{Y}$. Um* Semi-clique *é um conjunto de variáveis $\mathbf{Q}$ onde cada variável $X \in \mathbf{Q}$ está ligada a mais da metade de $\mathbf{Q}$: $|\Delta(X;\mathbf{Q})| > \frac{1}{2}|\mathbf{Q}|$*

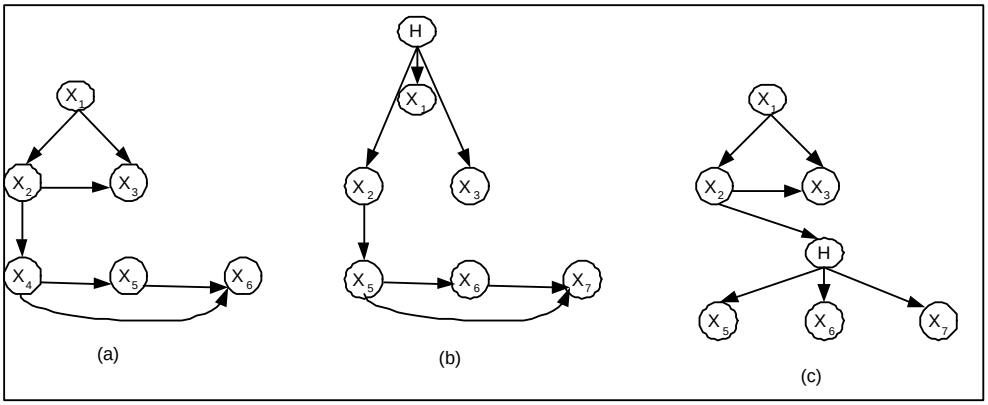


Figura 5.4: (a) Uma rede Bayesiana com 2 *Semi-cliques* $\{X_1, X_2, X_3\}$ e $\{X_4, X_5, X_6\}$; (b) Uma rede Bayesiana candidata quebrando o *Semi-clique* $\{X_1, X_2, X_3\}$ através da inclusão da *variável não-observada* H (c) Uma rede Bayesiana candidata quebrando o *Semi-clique* $\{X_5, X_6, X_7\}$ através da inclusão da *variável não-observada* H.

O algoritmo funciona da seguinte forma: na primeira parte, a rede Bayesiana aprendida usando um algoritmo de aprendizado convencional é analisada para encontrar *Semi-cliques*. Na segunda parte, cada *Semi-clique* ($\mathbf{Q}$) é convertido produzindo uma estrutura candidata, contendo uma *variável não-observada* (H), onde todas as aresta de entrada das variáveis em $\mathbf{Q}$ são substituídas por arestas a partir de H. Pais das variáveis em $\mathbf{Q}$ são então feitos pais de H, a não ser que resulte em um ciclo. Em seguida, todas as arestas internas do clique são removidas. Finalmente, na terceira fase, cada estrutura candidata é utilizada como ponto de partida para outra busca de estrutura. As redes resultantes desta busca são, então, avaliadas e a melhor implementada. Este processo continua até que a avaliação não possa ser mais melhorada.

Como exemplo, considere a Figura 5.4, onde (a) é a rede Bayesiana aprendida usando uma algoritmo de aprendizado de rede Bayesianas padrão. Esta rede tem dois *Semi-cliques* ($\{X_1, X_2, X_3\}$ e $\{X_4, X_5, X_6\}$), logo existem duas estruturas candidatas que são exibidas em (b) e (c) respectivamente, com H como *variável não-observada*. Um novo procedimento de aprendizado de redes Bayesianas é rodado para que então a rede Bayesiana de melhor avaliação seja escolhida.

Esta abordagem considera que *variáveis não-observadas* são introduzidas se elas simplificam sub-regiões complexas da rede. Por outro lado, não temos garantias

que *Semi-cliques* correspondem as *variáveis não-observadas*, e que toda *variável não-observada* corresponde a um *Semi-clique*.

O nosso algoritmo DAHVI apresenta uma proposta alternativa para detectar onde e quando uma *variável não-observada* deve ser introduzida na rede. Invés de guiar-se por cliques encontrados na estrutura da rede, DAHVI, guia-se por erros de inferência. E como veremos no Capítulo 6 mostrou ser uma abordagem boa para guiar a inclusão de *variáveis não-observadas* no algoritmo SEM.

Um benefício do DAHVI em comparação com o *Semi-clique* é que o primeiro pode ser aplicado a redes esparsas enquanto que o segundo não.

5.4.2 BANNER

BANNER (RAMACHANDRAN, MOONEY, 1998) é um algoritmo de revisão de redes Bayesianas *noisy-or* e *noisy-and*, que propõe modificações na estrutura de uma rede Bayesiana a partir de exemplos classificados incorretamente. Redes Bayesianas, que consideram variáveis *noisy-or* e *noisy-and*, assumem que todas as possíveis causas de um evento são conhecidas. Como nem sempre isso é verdade, pode-se utilizar as chamadas *variáveis leak*, que são variáveis extras, adicionadas ao conjunto de pais de uma determinada variável V, para indicar informações desconhecidas, mas relevantes para a variável V.

Na figura 5.5 podemos ver um exemplo de rede Bayesiana com variáveis *noisy-and* e *noisy-or*. Por ser uma rede Bayesian noisy-or noisy-and todas as suas variáveis tem um domínio binário associado.

BANNER, propõe 3 *operadores de revisão* baseados na inclusão de *variáveis leak* : adição de uma nova variável pai, adição de uma nova *variável não-observada* e exclusão de uma aresta. O algoritmo funciona da seguinte forma. Após serem detectados os exemplos que foram classificados incorretamente, são acrescentadas todas as *variáveis leak* possíveis, inicializando as CPDs correspondentes de forma a não alterar o significado original da rede. Essas variáveis são responsáveis por identificar os possíveis reparos: se a probabilidade inferida para uma *variável leak*, considerando um exemplo classificado incorretamente, diferir significativamente da probabilidade considerada a priori, então a *variável leak* representa uma informação

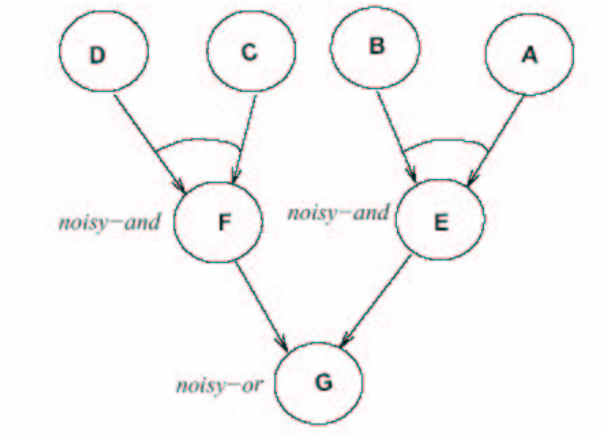


Figura 5.5: Uma rede Bayesiana onde os nós E e F são *noisy-and* e o nó G é *noisy-or*

desconhecida relevante, logo é considerada um *ponto de revisão*. São acrescentados três tipos de variáveis *leak*: (i) *node-leak*, que são variáveis inseridas como uma variável pai de uma variável *noisy-or* ou *noisy-and*; (ii) *intervening*, que são variáveis intermediária *noisy-and* (*noisy-or*) ligando uma variável *noisy-or* (*noisy-and*) aos seus pais; (iii) *link-leak*, que são variáveis *leak* associadas a uma variável *intervening*.

A Figura 5.6 exibe a rede Bayesiana da Figura 5.5 com as *variáveis leak* adicionadas. Para os *pontos de revisão* que foram gerados pelo maior número de exemplos classificados incorretamente são aplicados os *operadores de revisão*.

a) adição de uma variável aleatória pai - se a *variável leak* for uma variável *node leak* então é escolhida qualquer variável da rede para ser pai, desde que já não seja um pai ou um descendente da variável sendo revista, para evitar ciclos. É escolhida a melhor variável de acordo com a função *gain*.

b) adição de uma *variável não-observada* - se a variável sendo considerada é uma *link-leak*, acrescenta-se uma *variável não-observada* e uma nova variável pai para esta *variável não-observada*. Este pai também será escolhido através da função *gain*.

c) exclusão de uma aresta - se ao adicionar uma *variável não-observada* for escolhido a negação de uma das variáveis pai da variável sendo revista, então como temos um nó *noisy-or* pode-se excluir a aresta.

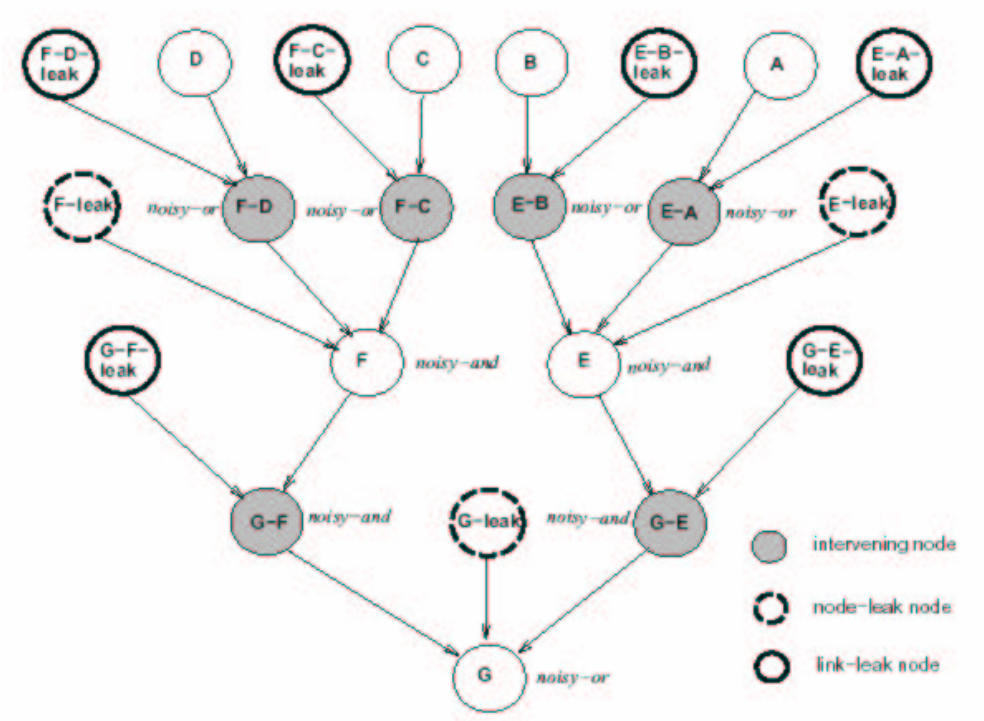


Figura 5.6: Uma rede Bayesiana com nós *leak* adicionados

A relação do BANNER com o DAHVI se dá ao fato de ambos serem algoritmos de revisão de redes Bayesianas, que utilizam os exemplos para indicarem potenciais pontos da estrutura da rede Bayesiana que podem ser modificados para que esta passe a refletir corretamente o *dataset*. A principal diferença entre eles é que enquanto DAHVI restringe o espaço de busca dos possíveis *pontos de revisão*, utilizando a *cobertura de Markov* da *variável de consulta*, BANNER precisa alterar toda a estrutura de rede, com a adição das *variáveis leak*, para então selecionar os *pontos de revisão*, tornando-se mais custoso. Além disso, DAHVI permite que uma *variável não-observada* seja incluída como sendo uma priori e não somente como uma intermediária, como feito pelo BANNER. Se por um lado BANNER é mais geral do que o DAHVI, já que ele propõe outras modificações que não somente inclusão de *variáveis não-observadas*, ele é mais restrito, pois só se aplica a redes noisy-or e noisy-and, onde todas as variáveis aleatórias tem domínio binário. Além disso BANNER só se aplica a *datasets* de classificação diferentemente do DAHVI. Devido a essas restrições uma comparação experimental não foi possível.

Capítulo 6

Resultados Experimentais do Sistema DAHVI

No capítulo 5, apresentamos o nosso sistema de revisão de redes Bayesianas, denominado DAHVI, o qual utiliza um *operador de revisão*, definido por nós, para incluir *variáveis não observadas*. Neste capítulo, faremos uma avaliação prática do DAHVI, através da aplicação dele em um *dataset* artificial e em 12 *datasets* reais. Para validar a nossa proposta experimentalmente, nós dividimos os experimentos em dois grupos: (i) no primeiro temos como objetivo mostrar que a revisão de uma rede Bayesiana através da inclusão de *variáveis não-observadas* torna a pontuação da rede Bayesiana melhor e (ii) no segundo temos como objetivo mostrar que a nossa heurística para detectar o melhor lugar para a inclusão da *variável não-observada* é boa. Na seção 6.1, descrevemos então a metodologia experimental adotada para alcançar cada um desses objetivos. Os *datasets* utilizados são descritos na seção 6.2 e na 6.3, apresentamos os resultados obtidos.

6.1 Metodologia Experimental

Para o primeiro objetivo, (i) uma rede Bayesiana é aprendida utilizando algum algoritmo de aprendizado conhecido. Em seguida, a rede Bayesiana aprendida é fornecida para o nosso sistema de revisão DAHVI, para que este a revise, caso necessário. Espera-se que a rede Bayesiana revisada tenha uma avaliação melhor do que a rede Bayesiana aprendida. Esta metodologia esta ilustrada na Figura 6.1.

Três algoritmos de aprendizado conhecidos na literatura foram utilizados para aprender uma rede Bayesiana: *hill-climbing*, *Structural Expectation Maximization*

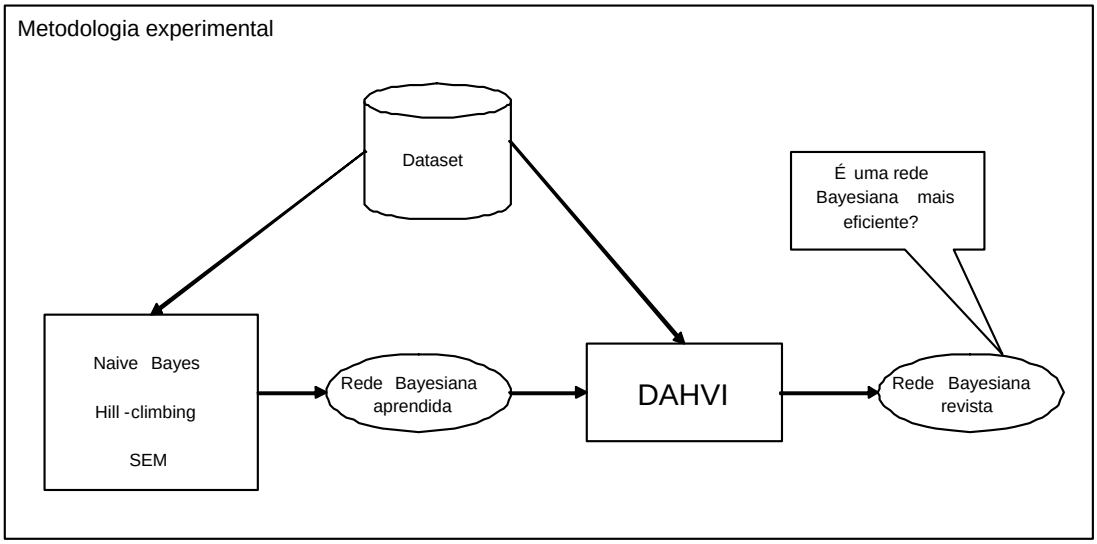


Figura 6.1: Metodologia experimental adotada para avaliar o nosso sistema DAHVI quando revisando uma rede Bayesiana através da introdução de *variáveis não-observadas*

(SEM) e o classificador *Naive Bayes*. Este último aprende apenas as CPDs das variáveis aleatórias de uma estrutura de rede, onde os atributos (A_i) são considerados independentes dada a *variável de classe* (V), como exibido na Figura 6.2.

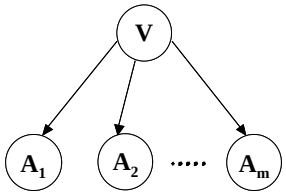


Figura 6.2: Classificador Naive Bayes

k-fold validação cruzada estratificada foi utilizada, separando os dados em conjuntos disjuntos de treinamento e teste mantendo a proporção entre as classes em cada fold. Além disso, em cada conjunto de treinamento,para evitar overfitting, é utilizado t-fold validação cruzada estratificada, separando este conjunto em conjuntos disjuntos de treinamento e validação (KOHAVI, 1995; BAIAO, MATTOSO, et al., 2003). Em cada k-fold a melhor rede Bayesiana, considerando o conjunto de validação, é avaliada no conjunto de teste. E o nosso objetivo é que esta avaliação indique que a rede Bayesiana retornada pelo nosso algoritmo DAHVI tenha uma avaliação melhor do que a rede Bayesiana aprendida. Em nossos experimentos, foram considerados k=10 e t=5.

Como mencionamos na seção 2.2.3, o algoritmo SEM também é capaz de incluir *variáveis não-observadas*, mas não de forma muito eficiente caso informações a respeito da localização e quantidade dessas *variáveis não-observadas* não sejam fornecidas. Para verificar (ii) o ganho proporcionado pelo nosso procedimento de seleção de *pontos de revisão*, nós comparamos o nosso algoritmo com o SEM, quando este aprende uma rede Bayesiana considerando uma *variável não-observada*. Denominamos este algoritmo de SEM_h. Caso a rede Bayesiana revisada pelo sistema DAHVI obtenha uma avaliação melhor do que a rede aprendida pelo algoritmo SEM_h, teremos mostrado que a nossa abordagem para inclusão de *variáveis não-observadas* é boa. A Figura 6.3 resume a metodologia experimental.

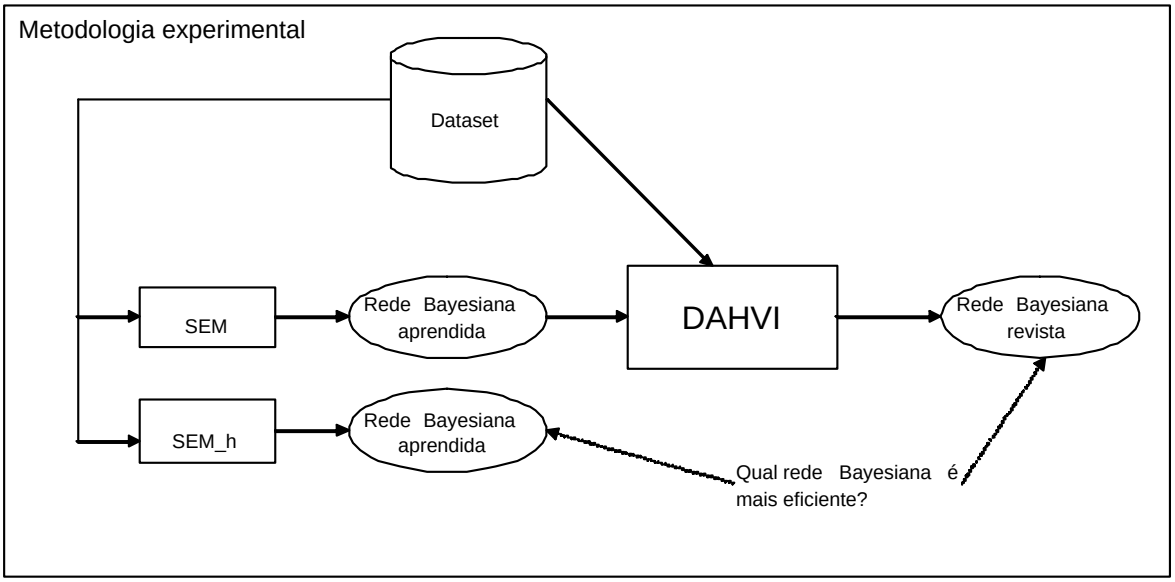


Figura 6.3: Metodologia experimental adotada para verificar os benefícios da nossa abordagem de detecção da localização da *variável não-observada*.

Assim como no primeiro experimento, k-fold validação cruzada com k=10 e t=5 foi utilizado no segundo experimento.

O sistema DAHVI foi implementado em Matlab utilizando funções do pacote BNT (MURPHY, 2001; MURPHY). Por esse motivo, foram utilizadas as implementações disponíveis neste pacote dos algoritmos de aprendizado considerados. Estas implementações consideram como função de avaliação para escolha do melhor modelo a log-verossimilhança. Já no nosso sistema DAHVI a função de avaliação utilizada é parametrizada, podendo ser: acurácia (ACC), log-verossimilhança (LL)

e log-verossimilhança condicional (CLL). O sistema DAHVI tem por objetivo maximizar a função de avaliação.

Para avaliar os modelos finais a média da log-verossimilhança obtida nos 10 conjuntos de teste e também a média da acurácia e da log-verossimilhança condicional para os problemas de classificação foram utilizadas. Com o intuito de trabalharmos com números menores, os valores encontrados para a log-verossimilhança e para a log-verossimilhança condicional foram divididos pelo tamanho do *dataset* correspondente. *Corrected t-test* foi utilizado para verificar a significância da diferença entra essas médias.

6.2 Datasets Utilizados

Nós utilizamos 13 *datasets* para avaliar a nossa proposta, sendo um deles artificial. Com exceção dos *datasets Stock* e *Tuberculose*, enviados por Gal Elidan (ver agradecimentos), todos os outros foram obtidos do repositório do UCI (REPOSITORIO UCI). A seguir descrevemos cada um dos *datasets* em detalhes.

a) *Audiology* (Audio) - considera um conjunto de informações a respeito de pacientes, como idade, histórico de sintomas (nausea, vômito, barulho no ouvido, problemas hereditários) para classificar o paciente de acordo com alguma característica auditiva.

b) *Breast-cancer* (BC) - considerando algumas informações sobre um tumor no seio, tais como idade do paciente, tamanho do tumor, localização no seio, classifica este tumor em um evento recorrente ou não.

c) *Breast-cancer-Wisconsin* (BCW) - diagnostica um tumor em maligno ou benigno baseando-se em características do tumor obtidas através da digitalização da sua imagem.

d) *Car* - define a aceitação de um carro (não-aceito, fracamente aceito, bem aceito e muito bem aceito) baseado no seu preço total (preço de compra, preço de manutenção) e características técnicas (conforto, número de portas, capacidade em termos de pessoas para carregar, tamanho do bagageiro e segurança

estimada)

e) *Lymphograph* (Lymph) - avalia o lymph definindo se ele é normal, metastases, maligno ou fibrose

f) *Nursery* - apresenta informações sobre candidatos a escola de enfermagem

g) *Post-operative* (Poper) - define para onde um determinado paciente deve ir após um cirurgia.

h) *Primary-tumor* (Ptumor) - baseando-se em algumas informações como idade, gênero, determina a localização de um determinado tumor.

i) *Stock* - mostra o desempenho de 20 empresas americanas de tecnologia na bolsa de valores, indicando se suas ações subiram, desceram ou manteram-se estáveis.

j) *Tuberculose* (TB) - contém informações sobre pessoas com tuberculose em um bairro de São Francisco, USA. Contém informações demográficas como sexo, idade, grupo étnico e informações médicas, como se está infectado com HIV, o tipo de tuberculose e o resultado de vários exames.

l) *Tic-tac-toe* (Tic) - contém o conjunto completo de possíveis configurações para o tabuleiro no final da execução do jogo da velha, onde "x"é assumido como tendo jogado primeiro. O conceito objetivo é "x venceu", isto é verdadeiro quando "x"tem uma das 8 possibilidades de ganhar).

m) *Zoo* - dada algumas características tais como, se bebe leite ou não, se é venenoso ou não, número de patas, se é doméstico ou não, classifica os animais em um dos 7 tipos possíveis.

n) *Insurance*: base artificial gerada a partir de uma rede com 27-nós utilizada para classificar seguros de carros(BINDER, KOLLER, et al., 1997).

Nós escolhemos *datasets* com características diversas: (i) *datasets* de classificação ou gerais; (ii) totalmente observados ou parcialmente observados; (iii) com muitos exemplos ou pouco. Além disso, todos os *datasets* considerados são discretos, já

que o sistema DAHVI apenas considera variáveis aleatórias discretas. A Tabela 6.1 fornece algumas informações relevantes sobre os *datasets* utilizados, tais como: (i) se a base é de classificação ou não; (ii) o número de atributos (no caso das bases de classificação este total engloba a *variável de classe*); (iii) o tamanho do domínio da *variável de classe* (Dom_{Classe}); (iv) o quantidade de exemplos; (v) e se esta base é totalmente observada ou não (Observada).

Tabela 6.1: Algumas informações relevantes sobre os *datasets* utilizados

Base	Classificação	N° de atributos	Dom_{Classe}	N° de exemplos	Observada
Audio	sim	70	24	226	não
BC	sim	10	2	286	não
BCW	sim	10	2	699	não
Car	sim	7	4	1728	sim
Lymph	sim	19	4	148	sim
Nursery	sim	9	5	12960	sim
Poper	sim	9	3	90	não
Ptumor	sim	18	22	339	não
Stock	não	20	-	1516	sim
TB	não	10	-	2302	sim
Tic	sim	10	2	958	sim
Zoo	sim	17	7	101	sim
Insurance	sim	27	4	100	sim

6.3 Resultados Experimentais

Nesta seção apresentamos os resultados obtidos com a aplicação do nosso sistema DAHVI aos 13 *datasets* descritos na seção 6.2. Para isto, dividimos esta seção em duas: na primeira avaliamos o nosso sistema em um *dataset* artificial, o *dataset Insurance*, e na segunda nos 12 *datasets* reais.

6.3.1 Dataset Artificial

Na seção 5, foi verificado um aumento de 5.7% no número de parâmetros e de 25% no tempo de cálculo da acurácia, quando retirada a *variável não-observada Accident* (nó 19 representado na Figura 5.2) da rede Bayesiana que modela o domínio de seguro de carros (*Insurance*). Com isso, percebemos a importância de uma *variável não-observada* e os ganhos que podem ser obtidos com a sua inclusão. Começamos,

então, os experimentos com a base artificial *Insurance*, onde esta foi gerada a partir da rede Bayesiana sem a variável *Accident* exibida na Figura 5.2, ou seja a rede Bayesiana com 26 nós. Consideramos o nó 25 como a *variável de consulta*.

No primeiro experimento (i) considerando a base artificial *Insurance*, desejamos verificar se o sistema DAHVI é capaz de re-colocar a *variável não-observada* e assim melhorar a eficiência da rede. No segundo (ii), indicamos para o algoritmo SEM a necessidade de inclusão de uma *variável não-observada* e comparamos os resultados com os apresentados pelo nosso sistema DAHVI, para verificarmos os ganhos de termos uma heurística guiando o local da inclusão desta *variável não-observada*. A Tabela 6.2 exibe a acurácia das redes Bayesianas retornadas pelos algoritmos *hill-climbing*, SEM, *Naive Bayes* e DAHVI, considerando log-verossimilhança como função de avaliação. Os valores em negrito indicam diferença estatisticamente significativa.

Tabela 6.2: Acurácia, log-verossimilhança e log-verossimilhança condicional das rede Bayesianas retornadas por cada um dos algoritmos de aprendizado e das redes retornadas pelo nosso sistema DAHVI, utilizando função de avaliação log-verossimilhança.

F	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	SEM_h	$DAHVI_{SEM}$
acc	83,00	$\mathbf{86,00}$	93,00	91,00	85,00	85,00	$\mathbf{88,00}$
ll	-5,68	$\mathbf{-4,8867}$	-4,4454	-4,4167	-4,9838	-4,9800	-4,9838
cll	$\mathbf{-0,0646}$	-0,0980	-0,0547	-0,0588	-0,1017	-0,0989	$\mathbf{-0,0812}$

Analisando as acurácias podemos perceber que o algoritmo *hill-climbing* foi o que retornou a rede com melhor acurácia. As redes Bayesianas aprendidas por esse algoritmo tiveram em média o dobro de arestas das redes aprendidas pelo algoritmo SEM, ou seja, o algoritmo *hill-climbing* foi capaz de encontrar mais dependências entre as variáveis do que o algoritmo SEM. Por outro lado, foi o algoritmo mais custoso dos três. O nosso algoritmo DAHVI, nos 3 casos, propõe a inclusão de uma *variável não-observada*, conseguindo uma rede com uma avaliação melhor no caso $DAHVI_{NB}$ e $DAHVI_{SEM}$, já que a diferença foi significativa. O $DAHVI_{SEM}$ conseguiu ainda retornar uma rede Bayesiana com acurácia melhor do que a rede Bayesiana retornada pelo SEM_h, mostrando que a nossa heurística para inclusão de *variáveis não-observadas* é promissora. Ao analisarmos a log-verossimilhança,

verificamos que o algoritmo *hill-climbing* continua sendo o mais promissor para este domínio, mas neste caso a diferença não é significativa. Já, ao analisarmos a log-verossimilhança condicional, verificamos que as retornadas pelo nosso algoritmo DAHVI pioram muito quando comparadas com as retornadas pelas redes aprendidas utilizando *hill-climbing* e Naive Bayes, neste último caso sendo a diferença estatisticamente significativa. Apenas revisando a rede aprendida pelo SEM é que o DAHVI melhora a log-verossimilhança condicional. Isso mostra que em geral, ao utilizar a log-verossimilhança como função de avaliação, prejudicamos a avaliação das redes Bayesianas em termos da log-verossimilhança condicional, que é considerada uma função adequada quando a tarefa é de classificação. Como o *dataset* do *Insurance* é um *dataset* para classificação pode ser vantajoso considerar esta função como a função de avaliação e assim aumentar a acurácia das redes retornadas pelo nosso algoritmo DAHVI. A Tabela 6.3 exibe a acurácia, e log-verossimilhança condicional quando utilizando como função de avaliação log-verossimilhança condicional.

Tabela 6.3: Acurácia e log-verossimilhança condicional das rede Bayesianas retornadas por cada um dos algoritmos de aprendizado e para as redes retornadas pelo nosso sistema DAHVI, utilizando função de avaliação log-verossimilhança condicional.

F	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	SEM_h	$DAHVI_{SEM}$
acc	83,00	$\mathbf{86,50}$	93,00	92,50	85,00	85,00	$\mathbf{87,50}$
cll	-0,0646	-0,0780	-0,0547	-0,0621	-0,1017	-0,0989	$\mathbf{-0,0857}$

Como a base *Insurance* é de classificação também verificamos o benefício de considerarmos a acurácia como função de avaliação. A Tabela 6.4 exibe os resultados

Tabela 6.4: Acurácia e log-verossimilhança condicional das rede Bayesianas retornadas por cada um dos algoritmos de aprendizado e das redes retornadas pelo nosso sistema DAHVI, utilizando função de avaliação acurácia.

F	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	SEM_h	$DAHVI_{SEM}$
acc	83,00	$\mathbf{85,50}$	93,00	92,50	85,00	85,00	86,50
cll	-6,46	-5,76	-5,47	-5,47	-0,1017	-0,0989	-0,0980

Resumindo, o nosso algoritmo DAHVI retornou uma rede Bayesiana com avaliação melhor do que a rede Bayesiana Naive Bayes considerando todas as funções de avaliação. Quando revisando a rede Bayesiana aprendida pelo algoritmo SEM ele

manteve a acurácia quando utilizando as funções de avaliação log-verossimilhança e acurácia e encontrou uma rede Bayesiana com melhor pontuação quando utilizando log-verossimilhança condicional como função de avaliação. E independente da função de avaliação a acurácia da rede revisada foi melhor do que a acurácia da rede aprendida considerando uma *variável não-observada*. Já revisando a rede aprendida pelo algoritmo *hill-climbing* o nosso sistema DAHVI não foi capaz de encontrar uma rede com avaliação melhor. A seguir, os experimentos foram repetidos considerando agora os outros *datasets* descritos na seção 6.2.

6.3.2 Datasets Reais

Nesta seção nós descrevemos os resultados obtidos ao aplicar o nosso sistema DAHVI em 12 *datasets* reais.

Datasets Gerais

Nesta seção aplicamos o nosso algoritmo em dois *datasets* gerais: Stock e TB. Como são *datasets* gerais, apenas utilizamos log-verossimilhança como função de avaliação. Para o *dataset TB*, consideraremos como *variável de consulta* o tipo de tuberculose. Já para o *dataset stock*, que fornece informações sobre 20 empresas, consideramos que a probabilidade de escolha entre as 20 variáveis é uniforme, logo um procedimento que pode ser feito é rodar o algoritmo 20 vezes variando a *variável de consulta* e considerando a melhor rede retornada.

O gráfico de dispersão exibido em 6.4 compara a log-verossimilhança retornada pelos sistemas SEM, *Hill-climbing* e Naive Bayes com a retornada pelo nosso sistema DAHVI para os *datasets* gerais: Stock e TB. Para o *dataset* Stock rodamos o DAHVI 20 vezes, onde em cada rodada uma das 20 variáveis aleatórias foi considerada como *variável de consulta*. A log-verossimilhança encontrada em cada rodada está exibida no gráfico de dispersão. Os três pontos mais em cima no gráfico dizem respeito ao *dataset* TB. Podemos ver que o DAHVI retornou uma rede melhor nos três casos, sendo o melhor resultado obtido quando revisando a rede *Hill-climbing*. Os outros pontos dizem respeito ao *dataset* Stock. Neste *dataset*, o sistema DAHVI conseguiu melhorar a log-verossimilhança da rede aprendida pelo algoritmo hill-

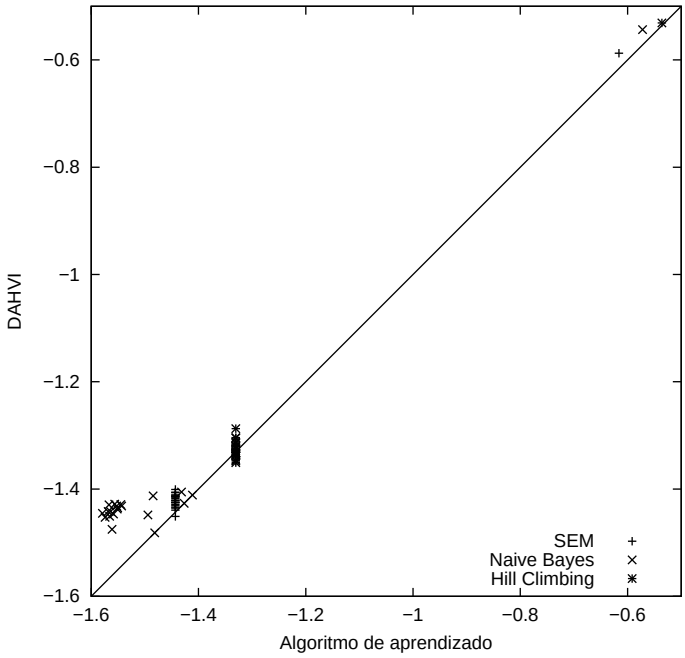


Figura 6.4: Comparação da log-verossimilhança retornada pelo sistema DAHVI, considerando cada uma das 20 variáveis do *dataset* Stock e do *dataset* TB, com a log-verossimilhança retornada pelos sistemas SEM, hill-climbing e Naive Bayes.

climbing quando considerando 11 das 20 variáveis e no caso da rede aprendida pelo SEM foram 19 das 20 variáveis. Já ao comparar com o Naive Bayes foi preciso aprender um Naive Bayes para cada *variável de consulta* considerada, já que a estrutura do Naive Bayes é dependente da classe. O DAHVI conseguiu melhorar a rede inicial em 17 das 20 rodadas. Concluímos, então que a apesar da nossa abordagem ser uma abordagem discriminativa ela é aplicável a *dataset* gerais, não sendo restrita a *datasets* de classificação.

Os valores utilizados para gerar o gráfico de dispersão 6.4 estão descritos em A.1.

Questionamentos podem ser feitos a respeito do nosso sistema DAHVI. Ao analisar as redes aprendidas pelos três algoritmos de aprendizado, qual delas retorna a melhor log-verossimilhança? Denominamos o algoritmo que a gerou por $Aprende_M$. O nosso sistema DAHVI consegue melhorar a rede aprendida por $Aprende_M$? Em caso negativo, revisando a rede aprendida por qual algoritmo de aprendizado o sistema DAHVI consegue a rede de melhor log-verossimilhança? Denominamos o algoritmo que aprendeu esta rede por $DAHVI_M$. Será que a rede revisada a partir da rede aprendida por $DAHVI_M$ obtém uma log-verossimilhança melhor do que

a rede aprendida por $Aprende_M$? Revisando quantas das três redes aprendidas, o sistema DAHVI consegue melhorar a log-verossimilhança? A Tabela 6.5 exibe essas análises.

Tabela 6.5: Resumo dos melhores algoritmos, onde a coluna $Aprende_M$ indica qual o melhor algoritmo de aprendizado, a coluna $Melhorou?$ indica se o DAHVI revisando a rede retornada por $Aprende_M$ melhora a log-verossimilhança, a coluna $DAHVI_M$ indica o algoritmo de aprendizado para o qual o DAHVI obteve a melhor rede ao revisa-lo; a coluna $Melhorou_2?$ indica se a rede retornada em $DAHVI_M$ tem melhor log-verossimilhança do que a rede aprendida por $Aprende_M$ e a coluna $Qtd.\ melhor?$ indica quantas redes revisadas obtiveram melhor log-verossimilhança do que a rede inicial correspondente.

Dataset	$Aprende_M$	$Melhorou?$	$DAHVI_M$	$Melhorou_2?$	$Qtd.\ melhor?$
Stock	HC	sim	HC	sim	3
TB	HC	sim	HC	sim	3

Analisando os resultados, vemos que para os dois *datasets*, o *hill-climbing* foi o algoritmo que retornou a melhor rede Bayesiana. Ele foi também o algoritmo mais lento, tendo em vista que ele tenta todas as possibilidades de alteração para todas as variáveis aleatórias. A melhor rede retornada pelo DAHVI também foi a rede revisada a partir da rede aprendida pelo *hill-climbing* e esta rede melhorou a log-verossimilhança da rede inicial. Para todos os *datasets* e considerando as três redes aprendidas, o DAHVI foi capaz de propor uma rede Bayesiana melhor em todos os casos.

Outra análise importante do sistema DAHVI diz respeito ao espaço de busca considerado para inclusão de *variáveis não-observadas*, ou seja o conjunto de *pontos de revisão* MB^*. Será que o nosso sistema DAHVI reduz o espaço de busca quando comparado aos sistemas de aprendizado? Além disso, quantas *variáveis não-observadas* o sistema DAHVI inclui na média? As Tabelas 6.6, 6.7 e 6.8 exibem essas informações considerando o DAHVI revisando a rede aprendida pelo *hill-climbing*, SEM e *Naive Bayes* respectivamente. A coluna MB^*_{media} exibe o tamanho em média do MB^* dos folds de treinamento referentes aos melhores folds de validação. Como em cada iteração, o tamanho do MB^* pode variar, nós consideramos o pior caso para computar o MB^*_{media}, ou seja consideramos sempre o maior tamanho. A co-

luna *Reducao%* indica o percentual de redução no tamanho do espaço de busca com relação ao número total de variáveis. Ou seja, um algoritmo de aprendizado considera todas as variáveis aleatórias da RB para propor modificações, o nosso sistema DAHVI considera apenas as variáveis pertencentes a MB^*. Em MB^*_{menor} e MB^*_{Maior} exibimos o menor e o maior tamanho de MB^* encontrados. A coluna *Qtd. Variaveis* fornece a média de *variáveis não-observadas* incluídas na RB através do nosso sistema DAHVI.

Tabela 6.6: Informações a respeito do conjunto de *pontos de revisão*, MB^*, gerados pelo DAHVI quando revisando a rede aprendida pelo *hill-climbing*. A coluna *Reducao%* indica a redução no espaço de busca proporcionada pela nossa abordagem. A coluna *Qtd. Variaveis* indica a quantidade média de *variáveis não-observadas* incluídas na RB.

DB	MB^*_{media}	MB^*_{menor}	MB^*_{Maior}	*Reducao%*	*Qtd. Variaveis*
Stock	5	3	7	75	1.7
TB	4.2	4	5	58	1.3

Ao analisarmos os resultados podemos perceber que para ambos os *datasets* o espaço de busca foi bastante reduzido, com exceção da rede NB, já que devido a sua topologia todas as variáveis são consideradas *pontos de revisão*. A maior redução foi proporcionada pelo DAHVI revisando a rede aprendida com SEM, 86%. Essa redução no espaço de busca não comprometeu a qualidade da rede retornada, pois como vimos pelo gráfico de dispersão 6.4 e no resumo da Tabela 6.5 as redes retornadas pelo DAHVI tiveram a sua log-verossimilhança aumentada. Isso mostra que a nossa abordagem para definição dos *pontos de revisão* é boa. Nos dois *datasets* a média de *variáveis não-observadas* inseridas foi superior a 1, o que indica que em pelo menos um dos folds DAHVI inseriu mais de uma variável.

A primeira iteração do sistema DAHVI quando revisando a rede Naive Bayes considera todas as variáveis como ponto de revisão devido a sua topologia. Com isso, a redução no espaço de busca só é percebida a partir da segunda iteração.

Como mencionado anteriormente esses experimentos também tem por objetivo verificar se a nossa abordagem para seleção dos *pontos de revisão* é válida. Para isso comparamos os resultados obtidos pelo DAHVI e o SEM, onde esse recebeu

Tabela 6.7: Informações a respeito do conjunto de *pontos de revisão*, MB^*, gerados pelo DAHVI quando revisando a rede aprendida pelo *SEM*. A coluna *Reducao%* indica a redução no espaço de busca proporcionada pela nossa abordagem. A coluna *Qtd. Variaveis* indica a quantidade média de *variáveis não-observadas* incluídas na RB.

DB	MB^*_{media}	MB^*_{menor}	MB^*_{Maior}	*Reducao%*	*Qtd.Variavel*
Stock	2.8	2	5	86	1.5
TB	3	2	5	70	1.5

Tabela 6.8: Informações a respeito do conjunto de *pontos de revisão*, MB^*, gerados pelo DAHVI quando revisando a rede aprendida pelo *Naive Bayes*. A coluna *Reducao%* indica a redução no espaço de busca proporcionada pela nossa abordagem. A coluna *Qtd. Variaveis* indica a quantidade média de *variáveis não-observadas* incluídas na RB.

DB	MB^*_{media}	MB^*_{menor}	MB^*_{Maior}	*Reducao%*	*Qtd.Variavel*
Stock	20	17	20	0	1.9
TB	10	6	10	0	1.8

a informação de que era preciso incluir *variáveis não-observadas*. Como a média de *variáveis não-observadas* inseridas pelo DAHVI ficou entre 1 e 2, consideramos que no geral 1 *variável não-observada* foi introduzida e rodamos o SEM indicando para ele que 1 *variável não-observada* precisava ser inserida. Os resultados obtidos estão descritos na Tabela 6.9. Os valores em **negrito**, *itálico* e sublinhados indicam diferença estatisticamente quando comparando a log-verossimilhança da rede retornada pelos sistemas: $DAHVIXSEM_h$, $SEMXSEM_h$ e $DAHVIXSEM$ respectivamente.

Tabela 6.9: Comparação da log-verossimilhança retornada pelo SEM, SEM considerando uma *variável não-observada* (SEM_h) e DAHVI

DB	*SEM*	*SEM_h*	*DAHVI*
Stock	*-1.4431*	−1.5389	**-1.4288**
TB	*-0,6161*	−0.6633	**-0,5875**

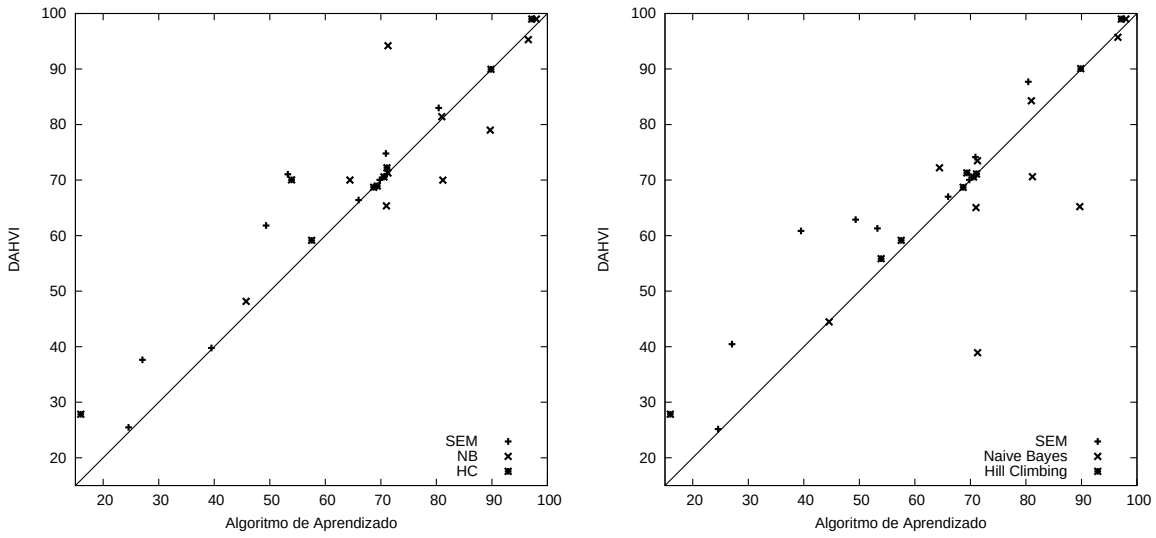
Verificamos que para os dois *datasets* a rede retornada pelo DAHVI obteve

a melhor log-verossimilhança e que a rede retornada pelo SEM_h piorou a log-verossimilhança. Dessa forma, concluímos que a nossa abordagem para selecionar os *pontos de revisão* é boa.

Datasets de Classificação

Nesta seção descrevemos os resultados obtidos aplicando o nosso sistema DAHVI a 10 *datasets* de classificação. Para esses *datasets* consideramos a *variável de classe* como sendo a *variável de consulta*. Três funções de avaliação foram utilizadas: log-verossimilhança, log-verossimilhança condicional e acurácia.

Os gráficos de dispersão exibidos nas Figuras 6.5(a), 6.5(b) e 6.6 mostram a comparação das acurácias das redes Bayesianas aprendidas utilizando SEM, *Hill-climbing* e Naive Bayes, com a acurácia das redes Bayesianas retornadas pelo nosso sistema DAHVI considerando log-verossimilhança, log-verossimilhança condicional e acurácia respectivamente como função de avaliação. Os valores que originaram esses gráficos estão descritos em A.2,A.4 e A.6.



(a) Log-verossimilhança

(b) Log-Verossimilhança Condicional

Figura 6.5: Comparando a acurácia retornada pelos sistemas SEM,Hill-climbing,Naive Bayes e DAHVI, considerando log-verossimilhança (a) e log-verossimilhança condicional (b).

Para o *dataset Audiology* não foi possível aprender uma rede *hill-climbing*, pois

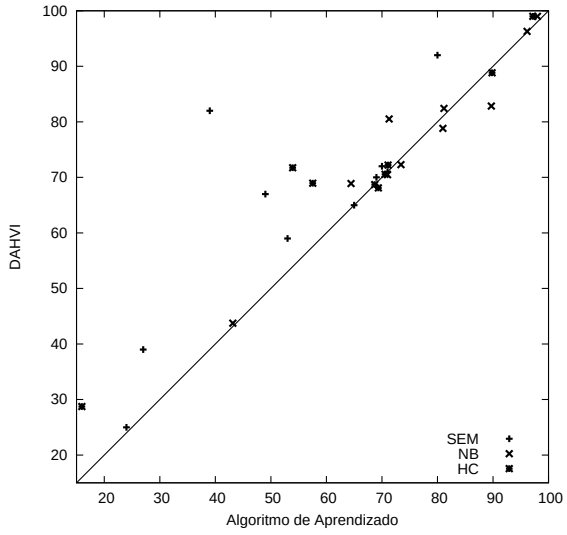


Figura 6.6: Comparando a acurácia retornada pelos sistemas SEM,Hill-climbing,Naive Bayes e DAHVI, considerando acurácia como função de avaliação.

esse algoritmo é muito custoso e passados 5 dias o algoritmo ainda não tinha terminado de aprender a teoria para o primeiro fold de treinamento. Uma justificativa é o fato desse *dataset* ter muitas variáveis, 70 variáveis, e o domínio delas ser grande, sendo a *variável de classe* a de maior domínio, 24 valores. Dessa forma os gráficos e Tabelas referentes ao algoritmo *hill-climbing* não tem o *dataset Audiology*.

Analisando os gráficos vemos que no geral o sistema DAHVI melhorou a acurácia das redes Bayesianas independente da função de avaliação considerada, mostrando que a revisão de uma rede Bayesiana através da inclusão de uma *variável não-observada* traz benefícios. Apenas em alguns *datasets* a rede Naive Bayes obteve melhor acurácia do que a rede revisada pelo DAHVI.

Assim como fizemos para os *datasets* gerais na seção 6.3.2, analisaremos o sistema DAHVI de acordo com as mesmas considerações. Qual das redes aprendidas pelos três algoritmos de aprendizado retorna a melhor acurácia? Denominamos o algoritmo que a gerou por $Aprende_M$. O nosso sistema DAHVI consegue melhorar a rede aprendida por $Aprende_M$? Em caso negativo, revisando a rede aprendida por qual algoritmo de aprendizado o sistema DAHVI consegue a rede de melhor acurácia? Denominamos o algoritmo que aprendeu esta rede por $DAHVI_M$. Será que a rede revisada a partir da rede aprendida por $DAHVI_M$ obtém uma acurácia melhor do que a rede aprendida por $Aprende_M$? Revisando quantas das três redes

aprendidas, o sistema DAHVI consegue melhorar a acurácia? A Tabela 6.10 exibe essas análises.

Tabela 6.10: Resumo dos melhores algoritmos, onde a coluna $Aprende_M$ indica qual o melhor algoritmo de aprendizado, a coluna $Melhorou?$ indica se o DAHVI revisando a rede retornada por $Aprende_M$ melhora a acurácia, a coluna $DAHVI_M$ indica o algoritmo de aprendizado para o qual o DAHVI obteve a melhor rede ao revisa-lo; a coluna $Melhorou_2?$ indica se a rede retornada em $DAHVI_M$ tem melhor acurácia do que a rede aprendida por $Aprende_M$ e a coluna $Qtd.\ melhor?$ indica quantas redes revisadas obtiveram melhor acurácia do que a rede inicial correspondente.

Dataset	$Aprende_M$			$Melhorou?$			$DAHVI_M$			$Melhorou_2?$			$Qtd._M$		
	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc			
Audio	NB	NB	NB	sim	não	sim	NB	SEM	NB	sim	não	sim	2	1	2
BC	SEM	NB	NB	sim	sim	não	SEM	SEM	NB	sim	sim	não	1	2	1
BCW	NB	NB	NB	não	não	sim	NB	NB	NB	não	não	sim	1	1	1
Car	NB	NB	NB	não	não	sim	HC	NB	NB	não	não	sim	2	2	3
Lymph	NB	NB	NB	sim	sim	não	NB	NB	NB	sim	sim	não	3	3	2
Nursery	HC	HC	HC	sim	sim	não	NB	HC	HC	sim	sim	não	1	2	1
Poper	SEM	HC	HC	sim	não	sim	HC	NB	HC	sim	não	sim	1	1	2
Ptumor	NB	NB	NB	sim	não	sim	NB	NB	NB	sim	não	sim	3	2	3
Tic	NB	NB	NB	não	não	não	HC	HC	NB	não	sim	não	1	2	0
Zoo	NB	NB	NB	sim	sim	sim	HC	HC	HC	sim	sim	sim	3	3	3

Analisando a Tabela 6.10 vemos que considerando qualquer função de avaliação a rede Naive Bayes é a de maior acurácia na maioria dos *datasets*, das 30 redes retornadas 23 foram Naive Bayes. O melhor resultado para o DAHVI também foi obtido na maioria das vezes revisando uma rede Naive Bayes, sendo que em menos casos, das 30 redes retornadas 17 foram revisando Naive Bayes. DAHVI melhora a acurácia da rede aprendida por $Aprende_M$ em 7, 5 e 6 dos 10 *datasets* considerando log-verossimilhança, log-verossimilhança condicional e acurácia como função de avaliação. Em todos esses casos, a melhor rede para o DAHVI é a rede obtida revisando a rede aprendida por $Aprende_M$, com exceção do *dataset* Tic, quando utilizando log-verossimilhança condicional, para o qual a melhor rede é a que revisa a rede aprendida com *hill-climbing*. Com exceção do *dataset Tic*, que quando considerando acurácia como função de avaliação, não conseguiu melhorar nenhuma das 3 redes que revisou, em todos os outros *datasets* o algoritmo DAHVI melhorou a acurácia de pelo menos uma rede revisada.

Outra análise a ser feita é com relação ao tamanho do conjunto de *pontos de revisão*. Será que ele reduz tanto assim o espaço de busca, principalmente nos

datasets onde verificamos o benefício do DAHVI? Além disso, quantas *variáveis não-observadas* o sistema DAHVI inclui na média? As Tabelas 6.6, 6.7 e 6.8 exibem essas informações considerando o DAHVI revisando a rede aprendida pelo *hill-climbing*, SEM e *Naive Bayes* respectivamente. A coluna MB^*_{media} exibe o tamanho em média do MB^* dos folds de treinamento referentes aos melhores folds de validação. Como em cada iteração, o tamanho do MB^* pode variar, nós consideramos o pior caso para computar o MB^*_{media}, ou seja consideramos sempre o maior tamanho. A coluna *Reducao%* indica o percentual de redução no tamanho do espaço de busca com relação ao número total de variáveis. Ou seja, um algoritmo de aprendizado considera todas as variáveis aleatórias da RB para propor modificações, o nosso sistema DAHVI considera apenas as variáveis pertencentes a MB^*. Em MB^*_{menor} e MB^*_{Maior} exibimos o menor e o maior tamanho de MB^* encontrados. A coluna *Qtd. Variaveis* fornece a média de *variáveis não-observadas* incluídas na RB através do nosso sistema DAHVI.

Tabela 6.11: Informações a respeito do conjunto de *pontos de revisão*, MB^*, gerados pelo DAHVI quando revisando a rede aprendida pelo *Hill-climbing*. A coluna *Reducao%* indica a redução no espaço de busca proporcionada pela nossa abordagem. A coluna *Qtd. Variaveis* indica a quantidade média de *variáveis não-observadas* incluídas na RB.

DB	MB^*_{media}			MB^*_{menor}			MB^*_{Maior}			*Reducao%*			*Qtd. Variaveis*		
	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc
BC	2	2	2	1	1	1	2	2	2	80	80	80	1.5	1.3	1
BCW	2	2	2	1	1	1	2	2	2	80	80	80	1.7	1.7	1
Car	2	2	2.4	2	2	2	2	2	6	71.43	71.43	65.71	1.1	1.2	1
Lymph	2	2	2	1	1	1	2	2	2	89.47	89.47	89.47	1.1	1.6	1.2
Nursery	6	6	6	5	1	5	6	6	6	33.34	33.34	33.34	1.7	1.4	1
Poper	2	2	2	1	1	1	2	2	2	77.78	77.78	77.78	1.7	1.4	1
Ptumor	2.1	2.6	2.1	2	2	2	3	3	3	88.34	86.32	88.34	1.4	1.7	2.2
Tic	2.3	2.4	2.6	2	2	2	3	3	3	77	76	74	1.4	1.3	1
Zoo	2	2	2	1	1	1	2	2	2	88.24	88.24	88.24	1	1	1

Considerando as redes aprendidas utilizando SEM e *hill-climbing*, podemos ver que o espaço de busca é sempre reduzido. Para todos os *datasets*, com exceção do *dataset Nursery* quando revisando a rede aprendida pelo *hill-climbing*, a redução é maior que 50%, independente da função de avaliação utilizada. Essa redução não compromete o bom resultado da rede revisada, como vimos pelos gráficos de dispersão. Por exemplo, o DAHVI retornou acurácia 70.04 quando revisando a rede aprendida pelo *hill-climbing*, de acurácia 53.93, para o *dataset* Car e função de

avaliação log-verossimilhança. O aumento de 23% em acurácia foi obtido propondo revisões em apenas 28,57% dos locais possíveis, ou seja em 28,57% das variáveis aleatórias da rede. Já considerando a revisão de uma rede Naive Bayes, o espaço de busca não é reduzido, já que pela topologia da rede a *Cobertura de Markov* da *variável de classe* são todos os atributos. Já na maioria dos *datasets*, ocorreu uma redução quando analisando o número mínimo do tamanho do MB^*. Ou seja, após a inclusão da primeira *variável não-observada* é possível reduzir o espaço das variáveis que serão consideradas *ponto de revisão*.

Tabela 6.12: Informações a respeito do conjunto de *pontos de revisão*, MB^*, gerados pelo DAHVI quando revisando a rede aprendida pelo SEM. A coluna *Reducao%* indica a redução no espaço de busca proporcionada pela nossa abordagem. A coluna *Qtd. Variaveis* indica a quantidade média de *variáveis não-observadas* incluídas na RB.

DB	MB^*_{media}			MB^*_{menor}			MB^*_{Maior}			*Reducao%*			*Qtd. Variaveis*		
	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc
Audio	2.4	2.1	2.5	1	1	1	4	3	3	96,57	97	96,43	2	1.2	1
BC	2.1	2.1	2.4	1	1	1	3	2	3	79	79	76	2.1	1.5	1.1
BCW	2	2	2	2	2	2	2	2	2	80	80	80	1.9	1.2	1.1
Car	2.4	2	3.3	1	2	2	6	2	6	65,71	71,43	52,86	1.9	1.1	1
Lymph	2.6	2	2.6	1	1	2	4	2	3	86,32	89,47	86,32	1.4	1.6	1
Nursery	2.1	3.7	3.8	1	2	1	3	4	4	76,66	58,89	57,78	1.6	1.2	1
Poper	2	2	2	1	1	1	2	2	2	77,78	77,78	77,78	2	1.3	1
Ptumor	2	2.1	2.1	1	1	1	2	3	3	88,89	88,34	88,34	1.4	1.4	1
Tic	2.7	2.5	2.8	2	2	2	3	3	3	73	75	72	1.6	1.5	1.2
Zoo	2.3	2.7	2.5	1	2	1	3	4	4	84,18	86,47	85,29	1	1.3	1

Analisando a média de *variáveis não-observadas* introduzidas considerando todas as funções de avaliação e os três algoritmos de aprendizado, verificamos que DAHVI introduziu mais de 1 variável em 65 casos dos 90 possíveis. Ou seja a média de variáveis inseridas foi maior do que 1 em 72.22% dos casos. Isso indica que para cada *dataset* pelo menos um fold retornou uma rede Bayesiana com 2 ou mais *variáveis não-observadas*, o que mostra a capacidade do sistema DAHVI em detectar a necessidade de *variáveis não-observadas* durante a execução. No fold 7 do *dataset Poper* quando revisando uma rede Naive Bayes utilizando log-verossimilhança como função de avaliação, por exemplo, o sistema DAHVI optou por incluir 5 *variáveis não observadas*, o que reduziu a log-verossimilhança da rede em 11%.

Outra análise que fizemos foi com relação a abordagem para seleção dos *pontos de revisão*. Comparamos os resultados obtidos pelo DAHVI e o SEM, onde esse recebeu

Tabela 6.13: Informações a respeito do conjunto de *pontos de revisão*, MB^*, gerados pelo DAHVI quando revisando a rede aprendida pelo *Naive Bayes*. A coluna *Reducao%* indica a redução no espaço de busca proporcionada pela nossa abordagem. A coluna *Qtd. Variaveis* indica a quantidade média de *variáveis não-observadas* incluídas na RB.

DB	MB^*_{media}			MB^*_{menor}			MB^*_{Maior}			*Reducao%*			*Qtd. Variaveis*		
	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc	ll	cll	acc
Audio	70	70	70	70	2	66	70	70	70	0	0	0	1	2	1.8
BC	10	10	10	10	10	10	10	10	10	0	0	0	2	1.6	1.9
BCW	10	10	10	2	2	2	10	10	10	0	0	0	1.9	2.6	1.1
Car	7	7	7	2	2	2	7	7	7	0	0	0	2.1	1.3	1.4
Lymph	19	19	19	19	19	19	19	19	19	0	0	0	1	1.4	1.3
Nursery	9	9	9	9	2	7	9	9	9	0	0	0	1.2	1.6	1.2
Poper	9	9	9	2	2	2	9	9	9	0	0	0	2.3	1.6	1.2
Ptumor	18	18	18	18	18	18	18	18	18	0	0	0	1.7	1.3	1.6
Tic	10	10	10	2	2	7	10	10	10	0	0	0	2.1	1.4	1.5
Zoo	2	2	2	1	1	1	2	2	2	0	0	0	1	1	1

a informação de que era preciso incluir *variáveis não-observadas* (SEM_h). Como a média de *variáveis não-observadas* inseridas pelo DAHVI ficou entre 1 e 2, consideramos que no geral 1 *variável não-observada* foi introduzida e rodamos o SEM indicando para ele que 1 *variável não-observada* precisava ser inserida. Como o algoritmo SEM aprende utilizando log-verossimilhança como função de avaliação, para que pudéssemos avaliar o real benefício da nossa abordagem, nós utilizamos apenas esta função de avaliação. Os resultados obtidos estão descritos na Tabela 6.14. Os valores em **negrito**, *itálico* e sublinhados indicam diferença estatisticamente quando comparando a acurácia da rede retornada pelos sistemas: $DAHVIXSEM_h$, $SEMXSEM_h$ e $DAHVIXSEM$ respectivamente.

Verificamos que em 6 *datasets* a rede retornada pelo DAHVI obteve a melhor acurácia do que a rede retornada pelo SEM_h. Além disso, SEM_h ao tentar incluir uma *variável não-observada* piorou a acurácia da rede em todos os *datasets*.

6.3.3 DAHVI Comparado ao Algoritmo Relacionado Semi-clique

Nós aplicamos o sistema *Semi-clique* nas redes aprendidas utilizando o algoritmo SEM e hill-climbing utilizando os datasets reais descritos na seção 6.2. Para o

Tabela 6.14: Comparação da acurácia retornada pelo SEM, SEM considerando uma *variável não-observada* (SEM_h) e DAHVI

DB	*SEM*	*SEM_h*	*DAHVI*
Audio	*37.63*	25.58	**37.63**
BC	70.92	70.21	**74.14**
BCW	*90.71*	65.43	**87.71**
Car	70.04	70.04	70.05
Lymph	53.25	55.39	**61.30**
Nursery	*87.52*	41.59	**60.84**
Poper	71.11	71.11	71.11
Ptumor	24.87	24.87	25.48
Tic	67.19	68.42	67.02
Zoo	*65.67*	54.83	**62.88**

Naive Bayes não foi possível, já que a rede Naive Bayes não contém *semi-cliques* tornando inviável a aplicação do sistema *Semi-clique*. Em 10 *datasets*, a rede aprendida pelo algoritmo SEM não contém *semi-cliques*, o que inviabilizou a introdução de *variáveis não-observadas* através do sistema *Semi-clique*. Já o algoritmo *hill-climbing* aprendeu redes com cliques em 4 *datasets*. A Tabela 6.15 exibe a acurácia das redes retornadas pelos sistemas *Semi-clique* e DAHVI nestes *datasets*. Os valores em **negrito** indicam diferenças estatisticamente significativas.

Tabela 6.15: Comparando a acurácia da RB retornadas pelos sistemas *Semi-clique* e DAHVI quando recebendo uma rede aprendida pelos algoritmos SEM e Hill-climbing, considerando cada uma das 3 funções de avaliação: log-verossimilhança (ll), log-verossimilhança condicional (cll) e acurácia (acc). Mostramos apenas os resultados para os datasets, cuja as RB iniciais tinham cliques, o que permitia que o algoritmo *Semi-clique* fosse aplicado

DB	SEM					
	Função ll		Função cll		Função acc	
	Semi-clique	*DAHVI*	*Semi-clique*	*DAHVI*	*Semi-clique*	*DAHVI*
Nursery	34.21	39.77	34.05	**60.84**	33.57	**82.18**
Stock	**79.47**	75.89	-	-	-	-
DB	Hill−climbing					
	Função ll		Função cll		Função acc	
	Semi-clique	*DAHVI*	*Semi-clique*	*DAHVI*	*Semi-clique*	*DAHVI*
Nursery	81.75	**89.95**	79.41	**90.06**	81.75	**89.78**
Stock	**79.83**	58.02	-	-	-	-
Zoo	83.33	**99.00**	83.99	**99.00**	83.33	**99.00**
TB	**77.68**	55.49	-	-	-	-

Como podemos visualizar na Tabela 6.15, o sistema DAHVI retorna uma rede mais acurada do que o sistema *Semi-clique* em todos os *datasets* de classificação e para todas as funções de avaliação utilizadas. Já para os datasets gerais (Stock e TB) a acurácia retornada pelo DAHVI foi pior, mas ao analisarmos a log-verossimilhança exibida na Tabela 6.16 para estes dois *datasets*, verificamos que o DAHVI é melhor do que o *Semi-clique*.

Tabela 6.16: Comparando a log-verossimilhança da RB retornadas pelos sistemas *Semi-clique* e DAHVI quando recebendo uma rede aprendida pelos algoritmos SEM e Hill-climbing, considerando log-likelihood como funções de avaliação para os *datasets*.

DB	SEM		Hill−climbing	
	Semi-clique	*DAHVI*	*Semi-clique*	*DAHVI*
Stock	−1.5362	**−1.4008**	−1.3395	**−1.2875**
TB	-	-	−0.5816	**−0.5311**

Capítulo 7

PFORTE_PI: Revisão de Teorias Probabilísticas de Primeira-ordem com Introdução de Predicados Probabilísticos Inventados

Neste capítulo, com o objetivo de encontrar teorias probabilísticas de primeira-ordem mais acuradas, compactas e que representem informações "escondidas", nós propomos estender o nosso sistema de revisão de teorias probabilísticas, PFORTE, com dois novos *operadores de revisão* baseados em invenção de predicados. Na seção 7.1 definimos esses *operadores de revisão* e na seção 7.2 descrevemos alguns trabalhos relacionados.

PFORTE revisa teorias compostas por *cláusulas probabilísticas*, ou seja cláusulas com distribuição de probabilidades associadas. Os exemplos classificados incorretamente, ou seja, exemplos que tiveram inferido o valor do domínio para o *predicado objetivo* diferente do valor definido no exemplo, são utilizados para determinar os *pontos de revisão*. Esses *pontos de revisão* são cláusulas cuja a cabeça ou é o *predicado objetivo* ou é um predicado pertencente à *Cobertura de Markov* do *predicado objetivo*. Dessa forma, a seleção dos *pontos de revisão* é feita através de uma abordagem discriminativa. Em seguida, *operadores de revisão*, como inclusão/exclusão de cláusulas ou antecedentes são aplicados, com o melhor, de acordo com uma função de avaliação probabilística, escolhido e implementado. A seção 4.2 descreve o algoritmo PFORTE em mais detalhes.

Como a maioria dos algoritmos de aprendizado, o nosso sistema PFORTE ape-

nas usa o vocabulário fornecido nos dados para revisar as teorias. Dessa forma, ele também pode ser beneficiado com a utilização de técnicas para estender de forma automática o vocabulário inicial, inventando novos predicados, seus domínios e distribuições de probabilidade associadas, definindo assim *predicados probabilísticos inventados*.

Os mecanismos de invenção de predicados (ver seção 3.3 para mais detalhes) em ILP, percorrem toda a estrutura da teoria para identificar aonde um *predicado inventado* poderia ser introduzido. Por exemplo, o mecanismo de reformulação percorre toda a teoria verificando por antecedentes comuns que possam ser intermediados pelo *predicado inventado*. No exemplo visto na seção 3.3, ao percorrer a teoria verificou-se que os antecedentes $pais(Z,X), pais(Z,Y), X <> Y$ eram comuns as cláusulas para $irmao(X,Y)$ e $irma(X,Y)$, optando-se por substituí-los pelo novo predicado ($irmaos(X,Y)$), como exibido a seguir:

$irmao(X,Y) : -genero(X, masculino), irmaos(X,Y).$

$irma(X,Y) : -genero(X, feminino), irmaos(X,Y).$

$irmaos(X,Y) : -pais(Z,X), pais(Z,Y), X <> Y.$

Ao combinarmos invenção de predicados com técnicas de revisão de teoria, permitimos que a introdução do *predicado inventado* seja feita em pontos específicos da estrutura da teoria, ou seja, nos *pontos de revisão*, reduzimos assim o espaço de busca e tornamos o mecanismo de invenção de predicados mais eficiente. Os sistemas KRT (WROBEL, 1994) e CWS (BAIN, MUGGLETON, 1992), por exemplo, propõem *operadores de revisão* para especialização de uma cláusula, que invés de adicionar antecedentes nesta, adicionam antecedentes na cláusula que define o *predicado inventado* e acrescentam esse ao corpo da cláusula sendo especializada, ou seja especializam através do *predicado inventado*. Esses *operadores de revisão* podem tornar a teoria mais compacta, já que em revisões futuras, a melhor proposta pode ser a especialização de uma cláusula através da adição deste predicado inventado como antecedente, invés da adição dos antecedentes pertencentes ao corpo da cláusula que define este predicado inventado. A diferença entre esses sistemas encontra-se na aridade do *predicado inventado*: KRT considera *predicados inventado* de aridade 1 e 2 enquanto que CWS considera *predicados inventado* com aridade igual ao número

de variáveis da cláusula *ponto de revisão*.

Motivados pelos benefícios (i) da utilização de técnicas de revisão nos mecanismos de invenção de predicados e (ii) da associação de distribuições de probabilidade condicionais às cláusulas permitindo a representação de incerteza, propomos combinar invenção de predicados com revisão de teorias de primeira-ordem probabilísticas através da extensão do nosso sistema PFORTE, criando o por nós denominado PFORTE_PI (**P**robabilistic **F**irst **O**rder **R**evision of **T**heories from **E**xamples with **P**redicate **I**nvention).

A extensão do sistema de revisão PFORTE, para que este considere invenção de predicados probabilísticos se dá através da definição de novos *operadores de revisão*. Nas seções seguintes desse capítulo apresentamos os dois novos *operadores de revisão*, propostos por nós, que utilizam técnicas de invenção de predicados para propor *predicados probabilísticos inventados* (REVOREDO, PAES, et al., 2006; REVOREDO, PAES, et al., 2007).

7.1 Operadores de Revisão Baseados em Invenção de Predicados

A base dos dois operadores, propostos por nós, é o mecanismo de reformulação, tendo por objetivo a simplificação da teoria, com a inclusão de um *predicado probabilístico inventado* intermediário, tornando-a mais compacta. Como estamos considerando teorias probabilísticas a compactação também ocorre através da redução do número de parâmetros probabilísticos. Considere, por exemplo, a teoria probabilística exibida na Tabela 7.1.

Tabela 7.1: Trecho da teoria referente ao domínio da Família

1.	$esposa(A, B) : -genero(A), casado(A, B).$
2.	$marido(A, B) : -genero(A), casado(A, B).$
3.	$mae(A, B) : -genero(A), pais(A, B).$
4.	$pai(A, B) : -genero(A), pais(A, B).$
5.	$filha(A, B) : -genero(A), pais(B, A).$
6.	$filho(A, B) : -genero(A), pais(B, A).$
7.	$irma(A, B) : -genero(A), pais(C, A), pais(C, B), A <> B$
8.	$irmao(A, B) : -genero(A), pais(C, A), pais(C, B), A <> B$

Um procedimento baseado no mecanismo de reformulação, percorreria toda a estrutura da teoria detectando que todas as cláusulas tem o antecedente $genero(A)$. Poderíamos então, colocar um *predicado probabilístico inventado* intermediário, que denominaremos $new1(A)$, entre cada um dos predicados *cabeça* e o *predicado probabilístico* $genero(A)$. O domínio de $new1(A)$ pode ser escolhido de forma aleatória ou segundo alguma heurística. A Tabela 7.2 exibe a nova teoria, onde $genero(A)$ foi substituído em todas as cláusulas por $new1(A)$ e em seguida a cláusula 9. definindo $new1(A)$ foi criada.

Tabela 7.2: Trecho da teoria referente ao domínio da Família, com o *predicado probabilístico inventado* $new1(A)$.

1.	$esposa(A,B) : -new1(A), casado(A,B).$
2.	$marido(A,B) : -new1(A), casado(A,B).$
3.	$mae(A,B) : -new1(A), pais(A,B).$
4.	$pai(A,B) : -new1(A), pais(A,B).$
5.	$filha(A,B) : -new1(A), pais(B,A).$
6.	$filho(A,B) : -new1(A), pais(B,A).$
7.	$irma(A,B) : -new1(A), pais(C,A), pais(C,B), A <> B$
8.	$irmao(A,B) : -new1(A), pais(C,A), pais(C,B), A <> B$
9.	$new1(A) : -genero(A)$

Esta alteração não altera o poder de inferência da teoria, do ponto de vista lógico, ou seja os exemplos que eram provados antes continuam sendo. Já do ponto de vista de inferência probabilística, como existem CPDs associadas a cada uma das cláusulas, e conseqüentemente associadas à cláusula definindo o *predicado probabilístico inventado* também, a classificação dos exemplos pode variar. Principalmente se o *dataset* for parcialmente observado, já que ao introduzirmos o *predicado probabilístico inventado* estamos introduzindo uma nova *variável não-observada*, pois os exemplos não contém valores para este predicado. Dessa forma, o aprendizado de parâmetros é feito utilizando o algoritmo EM adaptado (ver seção 4.1.2 para mais detalhes) e com isso as CPDs são aproximadas e conseqüentemente podem ser diferentes das anteriores. Um exemplo que antes não era classificado corretamente pode passar a ser. Com relação à complexidade, considerando que o domínio do *predicado probabilístico inventado* $new1(A)$ é binário, o número de parâmetros probabilísticos aumentou de 112 para 116, logo mesmo que tenhamos um ganho em classificação,

acabamos aumentando um pouco a complexidade da teoria. Já o par de antecedentes $genero(A), pais(A, B)$ se repete em 4 cláusulas (6 cláusulas se quisermos englobar $genero(A), pais(C, A)$ das duas ultimas cláusulas, apesar das variáveis diferentes) e ao colocarmos um *predicado probabilístico inventado* intermediário entre eles e os predicados *cabeça*, temos uma redução do número de parâmetros probabilísticos de 112 para 104 e se considerarmos a substituição também para as cláusulas que definem *irma* e *irmao* reduzimos para 72, o que diminui a complexidade da teoria em 36% (veja Tabela 7.3 com a nova teoria). Com isso, concluímos que para termos uma redução na complexidade da teoria é interessante que o *predicado probabilístico inventado* intermedie mais de um antecedente.

Tabela 7.3: Teoria da teoria referente ao domínio da família, com o *predicado probabilístico inventado* substituindo 2 antecedentes.

1.	$esposa(A, B) : -genero(A), casado(A, B).$
2.	$marido(A, B) : -genero(A), casado(A, B).$
3.	$mae(A, B) : -new1(A).$
4.	$pai(A, B) : -new1(A).$
5.	$irma(A, B) : -new1(A).$
6.	$filho(A, B) : -new1(A).$
7.	$irma(A, B) : -new1(A), pais(C, B), A <> B$
8.	$irmao(A, B) : -new1(A)), pais(C, B), A <> B$
9.	$new1(A) : -genero(A), pais(A, B).$

Ao estendermos o sistema PFORTE com *operadores de revisão* baseados em invenção de predicados, deixamos de percorrer toda a estrutura da teoria para detectar aonde o *predicado probabilístico inventado* poderia ser inserido. Os *operadores de revisão*, então, propõem a inclusão dos *predicados probabilísticos inventados*, nos *pontos de revisão* e em seguida verificam se outras cláusulas na teoria se beneficiariam com a substituição de alguns dos seus antecedentes pelo *predicado probabilístico inventado*. Como estamos nos baseando no mecanismo de reformulação, e este não altera o poder de inferência lógica da teoria, não existe benefício lógico em aplicar os novos *operadores de revisão* em *pontos de revisão* gerados a partir da falha em provar algum exemplo. Os nossos *operadores de revisão* consideram então, apenas os *ponto de revisão* gerados a partir da classificação incorreta de algum exemplo. Esses *pontos de revisão* são cláusulas da teoria.

A seguir apresentamos os nosso dois *operadores de revisão*, que utilizam invenção de predicados: (i) operador de *compactação* e (ii) operador de *incremento*. A incorporação desses dois *operadores de revisão* ao sistema PFORTE gera um novo sistema, que denominaremos PFORTE_PI.

7.1.1 Operador de Compactação

O *operador de compactação* é um *operador de revisão*, e como tal, recebe a teoria corrente e um *ponto de revisão* retornando uma proposta de teoria modificada. Como visto anteriormente, o *ponto de revisão* recebido pelo *operador de compactação* é uma *cláusula probabilística*. O algoritmo em alto nível pode ser visto em *Algoritmo* 9.

Algoritmo 9 Operador de revisão - compactação

Entrada: teoria corrente T e um *ponto de revisão* (cláusula C);
Saída: uma teoria proposta T_{mod}

1: inventar um novo predicado (new_i);
2: adicionar new_i a linguagem;
3: A := todos os antecedentes em C;
4: alterar a cláusula C substituindo os antecedentes representados por A por new_i;
5: criar a cláusula C_{new} para definir new_i com os antecedentes representados por A no corpo;
6: aprender as CPDs;
7: calcular o valor da função de avaliação ($score_{mod}$);
8: buscar por outras cláusulas na teoria que tenham os antecedentes representados por A no corpo ($T_{outras} \subseteq T$);
9: **se** $T_{outras} \neq \emptyset$ **então**
10: **para** $C_i \in T_{outras}$ **faça**
11: trocar os antecedentes representados por A em C_i por new_i;
12: aprender as CPDs;
13: calcular o valor da função de avaliação ($score_{outras}$);
14: **se** $score_{outras} > score_{mod}$ **então**
15: a teoria resultante (T_{mod}) considera C_{new} e C_{mod} e as cláusulas C_i alteradas;
16: **senão**
17: a teoria resultante (T_{mod}) considera apenas C_{new} e C_{mod};

Para detalhar o algoritmo, vamos considerar como entradas a teoria exibida na Tabela 7.1, e a cláusula 8. .

O passo 1 do algoritmo do *operador de compactação* é composto de três etapas:

(i) denominar por *newi* o $funtor$ do *predicado probabilístico inventado*, onde i é um número seqüencial inicializado com valor 1 no começo da execução do

sistema PFORTE_PI. Por exemplo: $new1$, $new2$,... No exemplo da Familia visto anteriormente, o functor do *predicado probabilístico inventado* é *new1*.

(ii) definir a aridade do *predicado probabilístico inventado* como sendo a aridade do predicado da *cabeça* da cláusula *ponto de revisão*. Lembramos que, como vimos na seção 3.3, os argumentos do *predicado inventado* são qualquer um dos definidos no *predicate utility lattice* (MUGGLETON, 1993). Da mesma forma, a aridade do *predicado probabilístico inventado* também pode ser definida de acordo com o *predicate utility lattice*. Como percorrer todo o *predicate utility lattice* é muito custoso, optamos por escolher um dos seus argumentos e essa escolha foi feita de forma que pudéssemos aproveitar a CPD da cláusula *ponto de revisão* como ponto de partida para o aprendizado da CPD da cláusula que define o *predicado probabilístico inventado*, tornando assim o aprendizado de parâmetros mais eficiente. No exemplo da Familia, como a aridade de $irmao(A, B)$ é 2, a aridade do *predicado probabilístico inventado* também é definida como sendo 2, $new1(A, B)$.

(iii) definir o domínio do *predicado probabilístico inventado* como sendo o mesmo domínio do *predicado probabilístico* da *cabeça* da cláusula *ponto de revisão*. No exemplo da Familia, como o domínio de $irmao(A, B)$ é binário, o domínio de $new1(A, B)$ também será binário.

O passo 2 do algoritmo associa o mesmo tipo das variáveis da cabeça da cláusula *ponto de revisão* às variáveis do *predicado probabilístico inventado* e em seguida este é inserido na linguagem como um predicado intermediário. No exemplo da família, o tipo das variáveis do predicado $irmao(A, B)$ é *pessoa*, logo o tipo das variáveis do *predicado probabilístico inventado* $new1(A, B)$ também será *pessoa*.

No passo 3, o corpo da cláusula *ponto de revisão*, que denominaremos por A, é guardado para ser então substituído por $newi$ no passo 4, definindo a cláusula C_{mod}. A CPD da cláusula C_{mod} é alterada para vazia, já que a alteração na quantidade de antecedentes altera a quantidade de parâmetros probabilístico e conseqüentemente é preciso re-iniciá-la. No exemplo da familia temos:
$A = genero(A), pais(C, A), pais(C, B), A <> B$ e

C_{mod} como sendo a cláusula $irmao(A, B) : -new1(A, B)$.

A cláusula que define o *predicado probabilístico inventado* é criada no passo 5 do algoritmo, onde o corpo são os antecedentes armazenados em A, definindo a cláusula C_{new}. A CPD da cláusula C_{new} também poderia ser inicializada com vazio, mas verificamos os benefício do aprendizado de parâmetros partir da CPD original da cláusula *ponto de revisão*, já que a quantidade de parâmetros probabilísticos da CPD é a mesma da cláusula *ponto de revisão* C e o corpo desta é exatamente o corpo da cláusula C_{new}. No exemplo da familia temos:
C_{new} como $new1(A, B) : -genero(A), pais(C, A), pais(C, B), A <> B$.

No passo 6 as CPDs são re-aprendidas utilizando o algoritmo EM adaptado. Em seguida o valor da função de avaliação probabilística é calculado (passo 7). No passo 8, todas as cláusulas da teoria, com exceção das cláusulas C_{mod} e C_{new}, são analisadas verificando se outras cláusulas contém no seu corpo os mesmos antecedentes contidos em A. Caso existam, o corpo destas cláusulas é alterado para substituir os antecedentes que correspondem a A pelo *predicado probabilístico inventado* e as CPDs correspondentes são inicializadas com vazio (passo 11). No exemplo da *Família*, o corpo da cláusula

$$irma(A, B) : -genero(A), pais(C, A), pais(C, B), A <> B$$

também é alterado originando a cláusula

$$irma(A, B) : -new1(A, B)$$

Em seguida os parâmetros são re-aprendidos (passo 13) e o valor da função de avaliação ($score_{outras}$) é calculado (passo14). Caso o valor da avaliação tenha melhorado, a teoria resultante será a encontrada no passo 13, caso contrário será a teoria encontrada ao final do passo 6. O algoritmo termina retornando a teoria resultante como sendo a proposta de revisão do *operador de compactação*.

Com a introdução da cláusula definindo $new1(A, B)$ e a alteração das cláusulas que definem $irma(A, B)$ e $irmao(A, B)$ a teoria deixou de ter 112 parâmetros probabilísticos para passar a ter 88, uma redução de 22% no número de parâmetros probabilísticos. Com isso passamos a ter uma teoria probabilística mais compacta e de menor complexidade, o que aumenta a sua eficiência.

7.1.2 Operador de Incremento

Vimos anteriormente que a introdução de um *predicado probabilístico inventado* intermediário começa a reduzir a complexidade da teoria a partir da substituição de dois antecedentes. Além disso, vimos que o mecanismo de reformulação precisa percorrer toda a teoria para verificar grupos de antecedentes que se repetem, para então verificar o benefício de substituí-los por um *predicado inventado* intermediário. Seria interessante se fosse possível partir do primeiro antecedente onde verificou-se a repetição e junto com o *predicado inventado* encontrar outros através da especialização da cláusula que define o *predicado inventado*. Espera-se então uma redução no custo para encontrar grupos de antecedentes que se repetem.

O *operador de incremento* baseia-se nesta idéia. Ele usa a cláusula *ponto de revisão*, analisando um antecedente por vez, como ponto de partida para a especialização da cláusula que define o *predicado probabilístico inventado*. Como o *operador de compactação*, o *operador de incremento* é um *operador de revisão*, e como tal, recebe a teoria corrente e um *ponto de revisão*, sendo este uma *cláusula probabilística*, retornando uma proposta de teoria modificada. O algoritmo em alto nível é exibido em Algoritmo 10.

Para detalhar o algoritmo, vamos considerar como entradas a teoria exibida na Tabela 7.1, e a cláusula 8. .

Os passo 2 até 7 do algoritmo do *operador de incremento* são executados para cada um dos antecedentes da cláusula *ponto de revisão*. Ao final de cada iteração deste loop temos uma proposta de teoria com uma cláusula definindo o *predicado probabilístico inventado* e a cláusula *ponto de revisão* alterada, onde um determinado antecedente foi substituído pelo *predicado probabilístico inventado*. A seguir descrevemos em mais detalhes os passos do algoritmo.

O passo 2 é similar ao passo 1 do algoritmo do *operador de compactação* diferindo nos ítens (ii) e (iii):

(ii) definir a aridade do *predicado probabilístico inventado* como sendo a aridade do predicado do antecedente representado por A_j. Essa escolha foi feita de forma que pudéssemos aproveitar a CPD da cláusula *ponto de revisão*. No

Algoritmo 10 Operador de revisão - incremento

Entrada: teoria corrente T e um *ponto de revisão* (cláusula C);
Saída: uma teoria proposta T_{mod}

1: **para todo** antecedente A_j em C **faça**
2: inventar um novo predicado (new_i);
3: adicionar new_i a linguagem;
4: substituir A_j em C por new_i definindo C_{mod_j};
5: criar cláusula C_{new_j} para definir new_i com A_j no corpo;
6: especializar C_{new_j} de forma a melhorar o valor da função de avaliação;
7: guardar a teoria modificada T_j considerando o par (C_{mod_j}, C_{new_j}) com a sua avaliação ($score_j$);
8: escolher a teoria modificada T_j com melhor avaliação (T_{mod}, onde $C_{mod} \in T_{mod}$ e $C_{new} \in T_{mod}$), baseada na sua avaliação ($score_{mod}$);
9: $A :=$ todos os antecedentes da cláusula C_{new};
10: buscar por outras cláusulas em T_{mod}, que tenham os antecedentes representados por A no corpo ($T_{outras} \subseteq T_{mod}$);
11: **se** $T_{outras} \neq \emptyset$ **então**
12: **para** $C_k \in T_{outras}$ **faça**
13: trocar os antecedentes representados por A em C_k por new_i;
14: aprender as CPDs;
15: calcular o valor da função de avaliação ($score_{outras}$);
16: **se** $score_{outras} > score_{mod}$ **então**
17: atualizar T_{mod} com as cláusulas C_k alteradas;

exemplo da Familia, assumindo que A_j seja $genero(A)$, como a aridade deste é 1, a aridade de $new1$ também será 1: $new1(A)$.

(iii) definir o domínio do *predicado probabilístico inventado*, como sendo o mesmo domínio do *predicado probabilístico* do antecedente representado por A_j. No exemplo da Familia, assumindo que A_j seja $genero(A)$, como o domínio deste é binário, o domínio de $new1(A)$ também será binário.

O passo 3 do algoritmo associa o mesmo tipo das variáveis do antecedente representado por A_j às variáveis do *predicado probabilístico inventado* e em seguida este é inserido na linguagem como um predicado intermediário. O tipo da variável A em $irmao(A)$ é *pessoa*, logo o tipo da variável do *predicado probabilístico inventado* $new1(A)$, no exemplo da família, também é *pessoa*.

O passo 4 do algoritmo substituí A_j por $newi$, definindo a cláusula C_{mod_j}. A CPD da cláusula C_{mod_j} é mantida como ponto de partida para o algoritmo de aprendizado EM adaptado, já que a quantidade de parâmetros probabilístico é inalterada. No

exemplo da familia, assumindo que A_j seja $genero(A)$,temos:

$C_{mod_j} : irmao(A, B) : -new1(A), pais(C, A), pais(C, B), A <> B$

A cláusula que define o *predicado probabilístico inventado* é criada no passo 5 do algoritmo, onde o corpo é inicializado com o antecedente armazenado em A_j, definindo a cláusula C_{new_j}. A CPD da cláusula C_{new_j} é inicializada com vazio. No exemplo da familia, assumindo que A_j seja $genero(A)$, temos:

$C_{new_j} : new1(A) : -genero(A)$.

No passo 6, a cláusula definindo o *predicado probabilístico inventado* é especializada. Essa especialização é feita pelo *operador de revisão* que adiciona antecedentes. Esse *operador de revisão* re-aprende as CPDs, utilizando o algoritmo EM adaptado, e calcula o valor da função de avaliação probabilística a cada proposta de inclusão de antecedente, para ao final ter a especialização que melhor benefício proporcionou para a teoria.

No passo 7, é guardada a teoria proposta T_j, onde a cláusula C foi trocada pela cláusula C_{mod_j} e a cláusula C_{new_j} foi inserida. Armazena-se também o valor da função de avaliação probabilística correspondente.

Após propor uma teoria modificada para cada antecedentes da cláusula *ponto de revisão*, no passo 9, a teoria proposta que obteve a melhor avaliação é selecionada como sendo a teoria modificada proposta, denominada T_{mod}. O algoritmo utiliza C_{mod} e C_{new} para denominar a cláusula C alterada e a cláusula definição do *predicado probabilístico inventado* respectivamente. No problema da familia, o *operador de incremento* irá propor quatro novos pares de cláusulas, cada um baseado em um dos quatro antecedentes do *ponto de revisão*. Após avaliar cada um, vamos supor que a especialização feita a partir do antecedente $genero(A)$

$C_{new} : new1(A) : -genero(A), pais(B, A)$

tenha sido a escolhida.

No passo 10, o corpo da cláusula C_{new}, que denominaremos por A, é guardado para então no passo 11 todas as cláusulas da teoria, com exceção das cláusulas C_{mod} e C_{new}, serem analisadas verificando se outras cláusulas contém no seu corpo os mesmos antecedentes contidos em A. Caso existam, o corpo destas cláusulas é alterado para substituir os antecedentes que correspondem a A pelo *predicado pro-*

babilístico inventado e as CPDs correspondentes são inicializadas com vazio (passo 14). No nosso exemplo, o corpo de todas as cláusula com exceção das que definem $esposa(A, B)$ e $marido(A, B)$ são alteradas definindo a teoria exibida na Tabela 7.4. Em seguida os parâmetros são re-aprendidos (passo 16) e o valor da função de avaliação probabilística ($score_{outras}$) é calculado (passo17). Caso o valor da avaliação tenha melhorado, então a teoria resultante será a encontrada no passo 19, caso contrário será a teoria selecionada no passo 9. O algoritmo termina retornando a teoria resultante como sendo a proposta de revisão do *operador de incremento*.

Tabela 7.4: Teoria da teoria referente ao domínio da família, com o *predicado probabilístico inventado* substituindo 2 antecedentes.

1.	$esposa(A, B) : -genero(A), casado(A, B).$
2.	$marido(A, B) : -genero(A), casado(A, B).$
3.	$mae(A, B) : -new1(A).$
4.	$pai(A, B) : -new1(A).$
5.	$irma(A, B) : -new1(A).$
6.	$filho(A, B) : -new1(A).$
7.	$irma(A, B) : -new1(A), pais(C, B), A <> B$
8.	$irmao(A, B) : -new1(A)), pais(C, A), pais(C, B), A <> B$
9.	$new1(A) : -genero(A), pais(A, B).$

O *operador de incremento* também traz benefícios com relação a especialização da cláusula *ponto de revisão*, já que em algumas situações fica inviável especializar a cláusula tendo em vista o aumento em complexidade. A especialização passa a se tornar viável por ser feita através de um predicado intermediário. Suponha por exemplo a cláusula (7.1).

$$esposa(A, B) : -genero(A), genero(B), A <> B \tag{7.1}$$

O número de parâmetros probabilísticos é 16. Se nós especializássemos esta cláusula acrescentando o predicado $casado(A, B)$, a nova CPD passaria a ter 32 parâmetros probabilísticos. Se esta especialização for feita através do *operador de incremento* como exibido a seguir:

$esposa(A, B) : -genero(A), new1(B), A <> B$

$new1(B) : -genero(B), casado(A, B)$

teríamos no total 24 parâmetros probabilísticos, o que reduz a complexidade da teoria e conseqüentemente a torna mais eficiente.

O *operador de incremento* se relaciona em alguns aspectos com os *operadores de revisão* que inventam predicados nos sistemas de ILP, KRT (WROBEL, 1994) e CWS (BAIN, MUGGLETON, 1992). Nesses dois sistemas, um *operador de revisão* especializa uma cláusula da teoria através de um *predicado inventado*. A diferença desses *operadores de revisão* para o *operador de incremento* é que enquanto o último substitui um antecedente da cláusula sendo especializada pelo *predicado inventado*, os primeiros apenas acrescentam o *predicado inventado* no corpo da cláusula sendo especializada. Em uma abordagem lógica, os resultados obtidos são os mesmos. Já em uma abordagem probabilística, aumentamos mais o número de parâmetros probabilísticos da teoria quando adicionamos o predicado inventado do que quando substituímos algum antecedente da cláusula por ele. Considere por exemplo, que invés de substituir o predicado $genero(A)$ na cláusula 8 por $new1(A)$, nós apenas adicionamos $new1(A)$ ao corpo desta cláusula. Como conseqüência, invés das cláusulas 8 e 9 da teoria exibida na Tabela 7.4, teríamos as cláusulas:

8.1 $irmao(A, B) : -genero(A), pais(C, A), pais(C, B), A <> B, new1(A)$

9.1 $new1(A) : -pais(B, A)$.

O par (8.1,9.1) tem 68 parâmetros probabilísticos enquanto que o par (8,9) tem 40. Essa diferença pode ser equilibrada após a execução do passo 10 do algoritmo, onde antecedentes em outras cláusulas podem ser substituídos pelo *predicado probabilístico inventado*. Entretanto, isto nem sempre ocorre, como no caso da Familia. Enquanto o nosso operador substitui dois antecedentes em 5 cláusulas, o outro operador substitui apenas um, nas mesmas 5 cláusulas.

Alguns trabalhos da literatura, como o sistema SAYU-VISTA (DAVIS, ONG, et al., 2007) e o sistema MRC (KOK, DOMINGOS, 2007) consideram invenção de predicados em SRL. A principal diferença desses sistemas para o sistema PFORTE_PI, é o fato deles não considerarem uma heurística para reduzir o espaço de busca e além disso, apesar de serem algoritmos de SRL, aprenderem a estrutura da teoria a partir de um algoritmo lógico: Aleph (SRINIVASAN, 2001) no caso do SAYU-VISTA e CLAUDIEN (DE RAEDT, DEHASPE, 1997) no caso do MRC. Na seção 7.2 descre-

vemos esses algoritmos e as diferenças entre eles e o PFORTE_PI em mais detalhes.

No capítulo 8 descrevemos alguns experimentos feitos com os *operadores de revisão* apresentados neste capítulo.

7.2 Trabalhos Relacionados

7.2.1 Multiple Relational Clustering

Em (KOK, DOMINGOS, 2007) foi proposto um sistema, chamado MRC (**M**ultiple **R**elational **C**lustering), para inventar predicados estatísticos, baseado em Markov Logic (RICHARDSON, DOMINGOS, 2006). Markov Logic é uma extensão probabilística de lógica de primeira-ordem. Uma Markov logic Network (MLN) é um conjunto de fórmulas de primeira-ordem com pesos associados. Junto com um conjunto de constantes representando objetos em um domínio, MLN define uma rede Markov com um nó por átomo básico e um atributo por fórmula básica. O peso do atributo é o peso da fórmula de primeira-ordem que o originou. As fórmulas de primeira-ordem são aprendidas utilizando o sistema de ILP CLAUDIEN (DE RAEDT, DEHASPE, 1997) e a rede Markov correspondente é utilizada para aprender os pesos dessas fórmulas de primeira-ordem e efetuar inferências. MRC cria agrupamentos das variáveis da rede Markov e para cada agrupamento é definido um predicado inventado. A motivação para esta abordagem é similar a utilizada em (ELIDAN, LOTNER, et al., 2001), onde os agrupamentos são vistos como cliques na rede Markov. A diferença para outras abordagens de agrupamento é que nesta é permitido que um mesmo elemento pertença a mais de um agrupamento, por isso o nome do sistema *Multiple Relational Clustering*. Dessa forma, a invenção de predicados estatísticos é feita na rede Markov, obtendo os mesmo benefícios da inclusão de uma *variável não-observada* em uma rede Bayesiana, não sendo refletida na MLN correspondente.

MRC automaticamente inventa predicados agrupando objetos, atributos e relações. Os predicados inventados capturam regularidades arbitrárias sob todas as relações. MRC baseia-se na observação que, em domínios relacionais, múltiplos agrupamentos são necessários para capturar totalmente interações entre objetos. Considere o seguinte exemplo. Pessoas tem colegas de trabalho, amigos, habilidades técnicas e

hobbies. As habilidades técnicas de uma pessoa são melhor descobertas pelos seus colegas de trabalho, e seus hobbies pelos seus amigos. Se considerarmos um único agrupamento de pessoas, colegas e amigos ficariam misturados e nossa capacidade de predizer tanto habilidades quanto hobbies ficará comprometida. Invés disso, deve-se agrupar pessoas que trabalham juntas e simultaneamente pessoas que são amigas entre si. Cada pessoa pertence a ambos os agrupamentos. Parceria em um agrupamento de trabalho é um alto indicativo de habilidades técnicas, e parceria em um agrupamento de amigos um indicativo de hobbies.

Uma diferença do MRC para o PFORTE_PI é a forma como as probabilidades são incorporadas ao modelo. Em MRC são associados pesos as fórmulas de primeira-ordem, fazendo que um atributo seja originado de uma fórmula de primeira-ordem. Já no PFORTE as probabilidades são associadas aos predicados, fazendo com que as cláusulas tenham uma distribuição de probabilidade associada e que um atributo seja originado de um predicado. As probabilidades no PFORTE_PI são utilizadas para encontrar a melhor teoria, enquanto que no MRC a melhor teoria é encontrada por um sistema ILP, logo não utiliza probabilidades para avaliação das teorias. PFORTE_PI inventa um novos predicado e o introduz na teoria, enquanto que o MRC apenas utiliza este novo predicado na rede Markov, não fazendo com que ele faça parte do MLN correspondente. Em ambos os sistemas a dependência entre as instâncias de um exemplo são levadas em consideração. No MRC instâncias de exemplos diferentes não estão conectadas na rede Markov e no PFORTE elas estão em redes Bayesianas diferentes. Entretanto, MRC não precisa que esta distinção esteja explicita no *dataset*, enquanto que o PFORTE_PI precisa.

7.2.2 SAYU-VISTA

Em (DAVIS, ONG, et al., 2007) foi proposto um sistema SRL, chamado SAYU-VISTA (SAYU-**V***iew* **I***nvention by* **S***coring* **TA***bles*), que aprende visões (*views*) que introduzem novas tabelas relacionais, invés de simplesmente novos campos para uma tabela existente. Nesta abordagem, novas tabelas ou novos campos não são limitados em ser aproximações de um conceito objetivo, invés disso é executado um tipo de invenção de predicado. Assim como no sistema PFORTE_PI, SAYU-VISTA é capaz

de descobrir e incorporar à linguagem, predicados intermediários que são relevantes. Considere, por exemplo, a tarefa de predizer se 2 citações referenciam o mesmo artigo. A relação *co-autor* é potencialmente útil para remover ambigüidades entre citações (se S.Russell e S.J.Russell ambos tem listas similares de co-autores, então talvez as citações possam ser trocadas). Mais ainda, *co-autor* pode ser usado para construir características explicitas para o sistema, tais como um novo predicado *mesma_Pessoa*. Entretanto, a relação *co-autor* pode não ter sido fornecida para o sistema de aprendizado. O sistema SAYU-VISTA fornece um mecanismo para aprender uma visão nova como uma tabela totalmente inventada, tal como *co-autor*. Depois permite que uma relação ou predicado inventado seja usado na invenção de outras novas relações tais como *mesma_Pessoa*.

SAYU-VISTA, é uma extensão do sistema SAYU (DAVIS, BURNSIDE, et al., 2005) (**S***core* **A***s* **Y***ou* **U***se*). SAYU é um sistema que combina ILP, através do sistema Aleph, com aprendizado de redes Bayesianas, especificamente aprendizado de tree-augmented naive Bayes (redes TAN) (FRIEDMAN, GEIGER, et al., 1997). Diferentemente de outras abordagens que utilizam ILP para definir novas cláusulas, SAYU avalia cada cláusula não utilizando uma função de avaliação padrão de ILP, mas sim verificando o quanto esta cláusula ajuda a teoria na qual foi inserida avaliando-a através de uma rede TAN. O predicado objetivo representará a variável de classe na rede TAN e cada cláusula inserida na teoria será representada por uma variável aleatória binária na rede TAN. Uma determinada variável assume valor 1 para um exemplo se a cláusula correspondente for utilizada para provar este exemplo, e 0 caso contrário. SAYU, efetua uma busca gulosa por novas variáveis, que melhoram a predição da variável de classe. SAYU modificou o sistema Aleph da seguinte forma: Invés de usar cobertura, Aleph passa cada cláusula construída para o SAYU, o qual a converte em uma variável aleatória binária. Esta variável é acrescentada ao conjunto de treinamento corrente e SAYU aprende uma rede TAN, incorporando esta nova variável. Para avaliar o desempenho da rede TAN é utilizado a área embaixo da curva de precisão e revocação (*precision recall*) em um conjunto de validação. Se a variável piorar o desempenho da rede TAN ela é descartada, caso contrário ela é mantida, o que significa manter ou não a cláusula correspondente na teoria sendo

aprendida. Em contraste com Aleph, depois de aceitar uma nova cláusula, o exemplo que foi utilizado para gerá-la não é descartado e SAYU seleciona aleatoriamente um novo exemplo e reinicializa a busca. Dessa forma, dado um exemplo como ponto de partida para a busca de uma cláusula, SAYU não busca pela melhor cláusula, mas sim pela primeira cláusula que prova um determinado exemplo. Além disso, nada impede que o mesmo exemplo seja escolhido várias vezes.

A diferença do SAYU-VISTA para o SAYU é que o primeiro permite que sejam aprendidas cláusulas cuja a cabeça não corresponde a predicados dos exemplos. No domínio da Mamografia, que foi utilizado em (DAVIS, BURNSIDE, et al., 2005) e (DAVIS, ONG, et al., 2007), se os exemplos indicam se uma determinada anomalia ($ab1$) é maligna ou não através do predicado ($malignant(ab1)$), então o sistema SAYU apenas aprende cláusulas com este predicado na cabeça, tais como:

$malignant(A) : -arch\_distortion(A, present), same\_study(A, B),$
$calc\_fine\_linear(B, present)$

A variável aleatória da rede TAN correspondente a esta cláusula terá valor 1 (verdadeiro) para o exemplo $malignant(ab1)$, se o corpo desta cláusula for satisfeito quando a variável A é instanciada pela constante $ab1$.

Já o sistema SAYU-VISTA é capaz de aprender cláusulas com outros predicados na cabeça, sendo estes predicados inventados. Como por exemplo, a cláusula a seguir que define o predicado inventado $p11(Ab1, Ab2)$, onde pares de anomalias são relacionados

$p11(Ab1, Ab2) : -density(Ab1, D1), prior\_abnormality\_same\_loc(Ab1, Ab2),$
$density(Ab2, D2), D1 > D2$

Estes predicados inventados, podem ter (i) aridade maior do que a aridade do exemplo, como o predicado $p11(Ab1, Ab2)$ ou (ii) tipos diferente como, por exemplo, no predicado inventado $p(Visit)$, onde o termo $Visit$ refere-se a todas as anomalias encontradas em uma dada mamografia, invés de significar uma única anomalia. como no exemplo $malignant(ab1)$. No primeiro caso, para relacionar o predicado inventado com o exemplo utilizam-se os termos. Considere a cláusula definindo o predicado inventado $p11(Ab1, Ab2)$ e o exemplo $malignant(ab1)$. A variável $Ab1$ é instanciada pela constante $ab1$ e o sistema SAYU-VISTA utiliza agregação para

encontrar instanciações para as outras variáveis. No segundo caso, é considerada uma ligação entre os termos de tipos diferentes. Por exemplo, considera-se uma ligação entre o termo $Ab1$ e o termo $Visit$, sendo esta ligação fornecida pelo usuário.

SAYU-VISTA, aprende novos predicados efetuando uma busca nos corpos das cláusulas geradas, escolhendo os corpos que fornecem a melhor avaliação em uma rede TAN. O algoritmo de invenção de predicados espera receber um conjunto pré-definido de tipos de variáveis, que podem aparecer na cabeça de uma cláusula. É preciso também fornecer a ligação entre os termos. A aridade do predicado inventado é considerada como sendo igual ao do predicado objetivo ou igual ao do predicado objetivo mais 1. O tipo das variáveis é escolhido de forma aleatória no conjunto dos tipo possíveis.

A diferença principal entre o sistema SAYU-VISTA e o nosso sistema PFORTE_PI é que o primeiro aprende uma teoria lógica enquanto que o segundo aprende uma teoria probabilística. A cláusula que define o predicado inventado no sistema PFORTE_PI tem uma distribuição de probabilidades associada, distribuição essa que guia a geração dela, diferente do sistema SAYU-VISTA que apenas incorpora probabilidades após a cláusula já ter sido gerada para indicar se esta cláusula deve pertencer à teoria ou não. Resumindo, o sistema SAYU-VISTA introduz *predicados inventados* enquanto que o PFORTE_PI introduz *predicados inventados probabilísticos*. Uma outra diferença é que a nova cláusula criada pelo SAYU-VISTA relaciona-se com todas as outras referentes ao mesmo exemplo de forma independente, enquanto que a cláusula do predicado inventado criada pelo PFORTE_PI está ligada a pelo menos uma das outras cláusulas, já que alteramos o corpo das cláusulas existentes. Outra diferença importante é que o PFORTE_PI é um sistema de revisão, logo ele parte de exemplos classificados incorretamente para encontrar uma cláusula que deva ser modificada com a inserção de um predicado inventado. Já o sistema SAYU-VISTA é um sistema de aprendizado, logo ele vai utilizar todos os exemplos para propor novas cláusulas com predicados inventados na cabeça. Como o sistema SAYU-VISTA utiliza o sistema Aleph para gerar as cláusulas ele apenas prediz cláusulas para um único predicado objetivo, diferente do sistema PFORTE_PI que é capaz de revisar teorias com vários predicados objetivo.

7.3 Revisando ProbLog

Outra contribuição desta tese é a revisão de programas Prolog probabilísticos ou ProbLog (DE RAEDT, KERSTING, et al., 2008).

Uma teoria ProbLog (T)(DE RAEDT, KIMMIG, et al., 2007) é uma extensão do Prolog onde as cláusulas têm uma probabilidade associada, que indica a probabilidade delas serem verdadeiras: $T = \{p_i : c_i | 1 \leq i \leq n\}$, onde c_i são cláusulas definidas e p_i probabilidades. As cláusulas são consideradas totalmente independentes.

Considere $L(T) = \{c_i | 1 \leq i \leq n\}$ como o programa lógico associado a teoria ProbLog T. Uma distribuição de probabilidades sobre todos os sub-programas lógicos (L, onde $L \subseteq L(T)$) possíveis da teoria ProbLog T é definida, onde:

$$P(L|T) = \prod_{ci \in L} pi \prod_{c_i \in L(T) \setminus L} (1 - pi)$$

Dado T, $L \subseteq L(T)$ e um exemplo e, a probabilidade do exemplo dado o programa lógico L ($P(e|L)$) é 1 quando $L \models e$ e 0 caso contrário. Temos então que

$$P(e, L|T) = P(e|L) * P(L|T).$$

A probabilidade de um exemplo considerando T depende da probabilidade de e em cada um dos sub-programas lógicos.

$$P(e|T) = \sum_{L \subseteq L(T)} P(e, L|T).$$

A verossimilhança da teoria ProbLog com relação aos exemplos é definida da seguinte forma.

$$LL(E|T) = \prod_{e \in E} LL(e|T)$$

onde $LL(e|T) = P(e|T)$ se e é um exemplo positivo ou $LL(e|T) = 1 - P(e|T)$ se e é um exemplo negativo.

A semântica do ProbLog é similar as utilizadas em Probabilistic Datalog (FUHR, 2000) e PRISM (SATO, KAMEYA, 2001), sendo que ProbLog utiliza um método eficiente de inferência, possibilitando a aplicação em problemas reais de grande porte como problemas de descoberta de links em grafos biológicos.

A utilização de técnicas de revisão de teoria à teorias ProbLog tem como objetivo encontrar a menor teoria que maximiza a verossimilhança. Dessa forma, apenas o operador de exclusão de regras é utilizado proporcionando uma compressão da teoria ProbLog (DE RAEDT, KERSTING, et al., 2008). A tarefa de revisão é definida da seguinte forma:

- **Dado:** uma teoria ProbLog, um conjunto de exemplos negativos e positivos e uma constante K.
- **Encontre:** uma teoria ProbLog com até K cláusulas que maximize a verossimilhança

O algoritmo de revisão, exibido em Algoritmo 11, é guloso e remove uma cláusula por vez. Ele considera que apenas as cláusulas utilizadas na prova de pelo menos um exemplo devam pertencer a teoria final, dessa forma todas as outras cláusulas são excluídas (passo 1). Se após a exclusão dessas cláusulas a teoria ainda tiver tamanho superior a K, o algoritmo prossegue excluindo outras cláusulas (passo 2). São encontradas n teorias a partir da exclusão de uma das n cláusulas (passos 4 a 6). Em seguida, a teoria resultante com uma cláusula a menos e com a melhor verossimilhança é selecionada (passos 7 e 8). O algoritmo continua excluindo cláusulas até a teoria atingir o tamanho K.

Algoritmo 11 Algoritmo de revisão de uma teoria ProbLog

Entrada: uma teoria ProbLog T, um conjunto de exemplos negativos e positivos E, uma constante K
Saída: Uma teoria ProbLog T'

1: $T' :=$ todas as cláusulas usadas na prova de algum exemplo;
2: **se** $|T'| > K$ **então**
3: **repete**
4: **para todo** $c_i \in T'$ **faça**
5: $T^{c_i} := T' - c_i$;
6: $Score^{c_i} := LL(E|T^{c_i})$;
7: $k := \arg\max_i \{Score^{c_i}\}$;
8: $T' := T^{c_k}$;
9: **até** $|T'| \leq K$

Capítulo 8

Resultados Experimentais do Sistema PFORTE_PI

No capítulo 7, nós propomos a extensão do nosso sistema de revisão de teorias probabilísticas de primeira-ordem, denominado PFORTE, com dois novos *operadores de revisão*, baseados em invenção de predicados, denominados: *operador de compactação* e *operador de incremento*, gerando um novo sistema que denominamos PFORTE_PI. Neste capítulo, nós conduzimos alguns experimentos para verificar na prática os benefícios desses novos operadores.

Os experimentos foram executados considerando dois *datasets* relacionais totalmente observados:

- **Alzheimer:** Compara 37 análogos de Tacrine, que é uma droga contra o mal de Alzheimer, de acordo com 4 propriedades (KING, STERNBERG, et al., 1995). Nesta tese consideramos a propriedade inibir a re-aceitação de **amine**. Os conjuntos de instâncias são compostos de 686 instâncias igualmente divididas nos valores de domínio positivos e negativos. Todas as instâncias pertencem ao mesmo exemplo.

- **Mutagênesis:** É um domínio bem conhecido para predizer o relacionamento entre a estrutura e atividade (SAR) de moléculas (SRINIVASAN, MUGGLETON, et al., 1996). O conjunto de instâncias é composto por 125 instâncias com valor de domínio positivo e 63 instâncias com valor de domínio negativo, onde cada instância corresponde a um exemplo. A meta é classificar compostos como "mutagênicos"ou não, dada sua estrutura química.

Uma primeira versão do sistema PFORTE_PI foi implementada utilizando o interpretador Quintus Prolog e algumas funções do *toolbox* BNT (MURPHY, 2001), sendo desenvolvido para o sistema operacional Windows. Com intuito de melhorar a performance, durante o desenvolvimento desta tese, o sistema PFORTE_PI foi todo convertido para rodar no sistema operacional Linux, substituindo o interpretador Quintus Prolog pelo Yap versão 5.1.3.

Para o *dataset* Mutagenesis rodamos 10-fold validação cruzada enquanto que para o *dataset Amine*, por questão de tempo (a execução de 1 fold demora aproximadamente 8 dias) rodamos 5-fold validação cruzada. A função de avaliação utilizada foi a log-verossimilhança condicional negativa, logo o melhor modelo é o que minimiza esta função.

8.1 Experimento1: PFORTE_PI melhora uma teoria inicial?

O primeiro objetivo da nossa investigação é verificar que os *operadores de revisão* baseados em invenção de predicados são capazes de propor alguma modificação à teoria inicial, retornando uma teoria mais compacta e acurada. Para isto, nós primeiramente, rodamos o sistema Aleph (SRINIVASAN, 2001) para aprender uma teoria inicial. Esta teoria inicial, foi fornecida para o sistema PFORTE_PI. Como o sistema Aleph é um sistema de ILP ele considera um *dataset* composto de exemplos positivos e negativos, diferente do nosso sistema PFORTE_PI, que considera apenas um conjunto de exemplos, deixando a cargo das distribuições de probabilidade a distinção entre positivo e negativo. Dessa forma, juntamos os exemplos positivos e negativos em um único grupo distinguindo entre positivo e negativo através dos valores do domínio associados as instâncias. A Tabela 8.1 exibe a média da log-verossimilhança condicional (cll), da acurácia (acc), do número de cláusulas (nº C), do número de literais (nº L) e do número de parâmetros probabilísticos (nº P) da teoria inicial retornada pelo Aleph (I) e da teoria final (F) retornada pelo nosso sistema PFORTE_PI, quando considerando apenas os 2 novos *operadores de revisão* nos *folds* de teste. Os valores em **negrito** indicam diferenças estatisticamente significativas. No *dataset* Mutagenesis os predicados *ring_size_5*, *phenantherene*,

$=<$ e $carbon\_6\_ring$ são considerados determinísticos, logo não têm distribuição de probabilidades associadas e não são considerados na rede Bayesiana gerada para cada um dos exemplos. Dessa forma, não são considerados no total de literais e parâmetros probabilísticos.

Tabela 8.1: Média da log-verossimilhança condicional (cll), acurácia (acc), número de cláusulas (nº C), número de literais (nº L) e número de parâmetros probabilísticos (nº P) da teoria inicial retornada pelo Aleph (I) e da teoria final (F), para os *folds* de teste.

DB	cll_I	acc_I	$n^\circ C_I$	$n^\circ L_I$	$n^\circ P_I$	cll_F	acc_F	$n^\circ C_F$	$n^\circ L_F$	$n^\circ P_F$
Muta	0.893	67.39	**6.7**	**13.8**	**40.6**	**0.64**	**81.20**	7.8	18.7	96.6
Amine	1.366	65.03	**15.8**	65.0	376	**0.984**	66.88	16.6	64.8	**331.2**

Como podemos verificar pela Tabela 8.1, o PFORTE_PI revisou a teoria retornada pelo Aleph, e apesar de ter aumentado o número de cláusulas, e no *dataset* Mutagenesis também o número de literais e parâmetros probabilísticos, encontrou uma teoria mais acurada e com uma melhor avaliação, sendo que para o *datasets* Amine a diferença em acurácia não foi estatisticamente significativa . Em todos os folds do Mutagenesis o operador escolhido foi o *operador de incremento* enquanto que nos folds do Amine foi o *operador de compactação*.

A Tabela 8.2 exibe a teoria inicial e a teoria final para um dos folds do *Dataset* Mutagenesis. A cláusula 6 é a cláusula *ponto de revisão*. O *operador de incremento* cria uma cláusula $new1(A) : -lumo(A)$ e em seguida tenta especializa-la gerando a cláusula

$$new1(A) : -lumo(A), ind1(A), logp(A)$$

A cláusula 6 é alterada substituindo $lumo(A)$ pelo *predicado probabilístico inventado* $new1(A)$. Como o predicado *lumo* tem domínio ternário e aridade 1 o *predicado probabilístico inventado* $new1$ também é criado com domínio ternário e aridade 1. Não tendo outras cláusulas com esse mesmo corpo a teoria final é a exibida na Tabela 8.2.

Como o *operador de incremento* especializou a cláusula que define o *predicado probabilístico inventado* $new1$ e não foi possível encontrar outras cláusulas onde este predicado substituísse antecedentes seus corpos, (i) o número de cláusulas aumentou em uma unidade, já que criamos uma nova cláusula para definir o *predicado*

Tabela 8.2: Teoria inicial retornada pelo sistema Aleph e teoria final retornada pelo nosso sistema PFORTE_PI considerando apenas os *operadores de revisão* baseados em invenção de predicados para um dos folds do *dataset* Mutagenesis

1	$class(A) : -logp(A), ring\_size\_5(A, C).$
2	$class(A) : -phenanthrene(A, B).$
3	$class(A) : -atm(A, B, h, 3, 0.0141).$
4	$class(A) : -atm(A, B, n, 38, C), C =< 0.787.$
5	$class(A) : -lumo(A), ring_size\_5(A, C).$
6	$class(A) : -lumo(A).$
7	$class(A) : -atm(A, B, n, 38, 0.805), bond(A, D, B, 1).$
8	$class(A) : -carbon\_6\_ring(A, B).$
1	$class(A) : -logp(A), ring\_size\_5(A, B).$
2	$class(A) : -phenanthrene(A, B).$
3	$class(A) : -atm(A, B, h, 3, 0.0141).$
4	$class(A) : -atm(A, B, n, 38, C), C =< 0.787.$
5	$class(A) : -lumo(A), ring\_size\_5(A, B).$
6	$\mathbf{new1(A) : -lumo(A), ind1(A), logp(A).}$
9	$\mathbf{class(A) : -new1(A).}$
7	$class(A) : -atm(A, B, n, 38, 0.805), bond(A, C, B, 1).$
8	$class(A) : -carbon\_6\_ring(A, B).$

probabilístico inventado, (ii) o número de literais aumentou em 4 unidades e (iii) consequentemente os parâmetros probabilísticos aumentaram de 54 para 80. Com isso, apesar de termos encontrado uma teoria mais acurada, a teoria final é mais complexa do que a inicial, como pode ser visto na Tabela 8.1.

A Tabela 8.3 exibe a teoria inicial para um dos folds do *dataset* Amine. A cláusula 1 é a cláusula *ponto de revisão*. O *operador de compactação* cria uma cláusula

$$new1(A, B) : -alk\_groups(B, C), ring\_substitutions(A, D)$$

O corpo da cláusula 1 é substituindo pelo *predicado probabilístico inventado* $new1(A, B)$. Como a cabeça da cláusula 1 é o predicado $great\_ne$, que tem domínio binário e aridade 2, o *predicado probabilístico inventado* $new1$ também é criado com domínio binário e aridade 2. Existem outras 5 cláusulas com $alk\_groups(B, C)$, $ring\_substitutions(A, D)$ no corpo, logo esses dois predicados são substituídos pelo *predicado probabilístico inventado* $new1(A, B)$. Esta alteração melhora a avaliação da teoria e por esse motivo é mantida, sendo a teoria final a exibida na Tabela 8.4.

Como o corpo da cláusula que define o *predicado probabilístico inventado* $new1$

Tabela 8.3: Teoria inicial retornada pelo sistema Aleph para um dos folds do *dataset* Amine

1	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,D).$
2	$great\_ne(A,B) :- x\_subst(A,C,D), ring\_subst\_4(B,D).$
3	$great\_ne(A,B) :- r\_subst_2(A,C), ring\_subst\_5(B,D).$
4	$great\_ne(A,B) :- alk\_groups(B,C), r\_subst\_3(A,D), ring\_substitutions(A,C).$
5	$great\_ne(A,B) :- x\_subst(A,C,D), r\_subst\_3(B,E), ring\_substitutions(A,F).$
6	$great\_ne(A,B) :- alk\_groups(B,C), alk\_groups(A,C).$
7	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,D), ring\_subst\_6(A,C), polarisable(C,D).$
8	$great\_ne(A,B) :- alk\_groups(A,C), ring\_substitutions(B,D).$
9	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,D), ring_subst\_6(A,C), h\_acceptor(C,D), r\_subst\_2(B,E), r\_subst\_2(A,E).$
10	$great\_ne(A,B) :- alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(B,D), ring\_substitutions(A,E), ring\_subst\_2(A,F).$
11	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,D), ring\_subst\_6(A,C), h\_acceptor(C,D), great\_h\_acc(D,E).$
12	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,D), h\_acceptor(C,D), great\_h\_acc(D,E), ring\_subst\_4(B,C).$
13	$great\_ne(A,B) :- alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(A,D), ring\_substitutions(B,E).$
14	$great\_ne(A,B) :- alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E).$
15	$great\_ne(A,B) :- alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E), ring_subst\_3(B,F), gt(D,G).$
16	$great\_ne(A,B) :- alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E), x\_subst(A,F,G).$
17	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,C), x\_subst(A,D,E), ring\_subst\_4(A,F), r\_subst\_2(B,G).$
18	$great\_ne(A,B) :- alk\_groups(B,C), ring\_substitutions(A,D), ring\_subst\_6(A,C).$

é encontrado em outras cláusulas, apesar do número de cláusulas aumentar em uma unidade, o número de literais e principalmente o número de parâmetros probabilísticos diminui, em 3 e 148 unidades respectivamente. Dessa forma, neste fold, não só a teoria final é mais acurada como ela é mais compacta.

8.2 Experimento2: PFORTE_PI X PFORTE

O objetivo deste segundo experimento é verificar se os *operadores de revisão* baseados em invenção de predicados proporcionam uma melhora significativa à teoria de forma que eles sejam os escolhidos quando comparados com os demais *operadores de revisão*

Tabela 8.4: Teoria final retornada pelo nosso sistema PFORTE_PI considerando apenas os dois novos *operadores de revisão* para um dos folds do *dataset* Amine

19	$\mathbf{new1(A,B)} : -\mathbf{alk\_groups(B,C)}, \mathbf{ring\_substitutions(A,D)}.$
1	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}.$
2	$great\_ne(A,B) : -x\_subst(A,C,D), ring\_subst\_4(B,D).$
3	$great\_ne(A,B) : -r\_subst\_2(A,C), ring\_subst\_5(B,D).$
4	$great\_ne(A,B) : -alk\_groups(B,C), r\_subst\_3(A,D), ring\_substitutions(A,C).$
5	$great\_ne(A,B) : -x\_subst(A,C,D), r\_subst_3(B,E), ring\_substitutions(A,F).$
6	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C).$
7	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}, \mathbf{polarisable(C,D)}.$
8	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D).$
9	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}, \mathbf{h\_acceptor(C,D)}, \mathbf{r\_subst\_2(B,E)}, \mathbf{r\_subst\_2(A,E)}.$
10	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(B,D), ring\_substitutions(A,E), ring\_subst\_2(A,F).$
11	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}, \mathbf{h\_acceptor(C,D)}, \mathbf{great\_h\_acc(D,E)}.$
12	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{h\_acceptor(C,D)}, \mathbf{great\_h\_acc(D,E)}, \mathbf{ring\_subst\_4(B,C)}.$
13	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(A,D), ring\_substitutions(B,E).$
14	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E).$
15	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E), ring\_subst\_3(B,F), gt(D,G).$
16	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E), x\_subst(A,F,G).$
17	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,C), x\_subst(A,D,E), ring\_subst\_4(A,F), r\_subst\_2(B,G).$
18	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}.$

do PFORTE. Para isto nós rodamos o sistema PFORTE_PI considerando todos os *operadores de revisão*, inclusive os novos operadores e comparamos com o sistema PFORTE, ou seja o nosso sistema de revisão original. A Tabela 8.5 exibe a média da log-verossimilhança condicional (cll), da acurácia (acc), do número de cláusulas (nº C), do número de literais (nº L) e do número de parâmetros probabilísticos (nº P) da teoria retornada pelo PFORTE e da teoria (PI) retornada pelo nosso sistema PFORTE_PI nos *folds* de teste. Os valores em **negrito** indicam diferenças estatisticamente significativas.

Para os dois *datasets* a teoria retornada melhorou a acurácia e a avaliação, sendo que a diferença apenas foi estatisticamente significante para a avaliação da teoria

Tabela 8.5: Média da log-verossimilhança condicional (cll), acurácia (acc), número de cláusulas (nº C), número de literais (nº L) e número de parâmetros probabilísticos (nº P) da teoria retornada pelo PFORTE_PI (PI) e da teoria retornada pelo PFORTE dos *folds* de teste.

DB	cll_PI	acc_{PI}	$n°C_{PI}$	$n°L_{PI}$	$n°P_{PI}$	cll	acc	$n°C$	$n°L$	$n°P$
Muta	0.486	85.30	5.8	14.4	66.2	0.487	84.19	**4.8**	**11.3**	**45.4**
Amine	**0.968**	66.88	16.60	64.80	**331.20**	1.211	66.78	15.4	63	364.80

no *dataset* do Amine. No caso do *dataset* Mutagenesis o *operador de incremento* foi escolhido em 8 dos 10 folds e no *dataset* Amine 4 dos 5 folds. Nos demais folds nenhum dos dois *operadores de revisão* propostos foi escolhido. O tempo médio de execução do PFORTE_PI é maior do que do PFORTE, como era de se esperar, já que no geral os *pontos de revisão* são os mesmos e para cada um deles o sistema PFORTE_PI propõe duas novas modificações.

Considere por exemplo as teorias retornadas pelo sistema PFORTE_PI e pelo sistema PFORTE para o *dataset* Mutagenesis, exibidas na Tabela 8.6.

Tabela 8.6: Teoria retornada pelo sistema PFORTE_PI e pelo sistema PFORTE para um dos folds do *dataset* Mutagenesis

1	$class(A) : -carbon\_6\_ring(A, B).$
7	$\mathbf{new1(A)} : -\mathbf{ind1(A)}, \mathbf{inda(A)}.$
2	$\mathbf{class(A)} : -\mathbf{lumo(A)}, \mathbf{new1(A)}, \mathbf{logp(A)}.$
3	$class(A) : -lumo(A), ring\_size\_5(A, B).$
4	$class(A) : -atm(A, B, h, 3, 0.0141).$
5	$class(A) : -phenanthrene(A, B).$
6	$class(A) : -logp(A), ring\_size\_5(A, B).$
1	$class(A) : -carbon\_6\_ring(A, B).$
2	$\mathbf{class(A)} : -\mathbf{lumo(A)}, \mathbf{ind1(A)}, \mathbf{logp(A)}, \mathbf{inda(A)}.$
3	$class(A) : -lumo(A), ring\_size\_5(A, B).$
4	$class(A) : -atm(A, B, h, 3, 0.0141).$
5	$class(A) : -phenanthrene(A, B).$
6	$class(A) : -logp(A), ring\_size\_5(A, B).$

A diferença entre elas é referente a cláusula 2, modificada a partir da cláusula *ponto de revisão* $class(A) : -lumo(A)$: enquanto o PFORTE aplicou o *operador de revisão adição de antecedente* retornando a cláusula especializada

$$class(A) : -lumo(A), ind1(A), logp(A), inda(A)$$

o PFORTE_PI também aplicou o *operador de revisão adição de antecedente*, mas especializando com um antecedente a menos

$$class(A) : -lumo(A), ind1(A), logp(A) \tag{8.1}$$

Essa diferença ocorreu porque foram encontradas CPDs diferentes para as cláusulas da teoria após o aprendizado dos parâmetros, consequentemente a avaliação da cláusula sendo especializada foi diferente também, fazendo com que o *operador de adição de antecedente* no PFORTE_PI atingisse a convergência antes do mesmo *operador de revisão* no PFORTE. Após outras modificações, o PFORTE_PI implementou a proposta do *operador de incremento* de substituir o antecedente $ind1(A)$ da cláusula 8.1 pelo *predicado probabilístico inventado* $new1(A)$ e especializar a cláusula que o define, encontrando a cláusula

$$new1(A) : -ind1(A), inda(A).$$

Podemos ver que a teoria encontrada pelos dois sistemas foi a mesma sendo que o sistema PFORTE_PI incluiu um *predicado probabilístico inventado* intermediando os antecedentes $ind1(A), inda(A)$ e o predicado $class(A)$. Essa diferença em termos do número de parâmetros probabilísticos foi importante já que, enquanto a proposta do PFORTE tem 92 parâmetros probabilísticos a do PFORTE_PI tem 64. A acurácia nos dois casos foi a mesma, 88.89, já a avaliação do PFORTE_PI foi melhor, 0.36, comparada com 0.37 da teoria retornada pelo PFORTE. Com esse exemplo vemos que o *operador de incremento* mostra-se vantajoso quando a revisão alternativa é a *adição de antecedentes* de forma que com a aplicação do *operador de incremento* o número de parâmetros probabilísticos é reduzido.

Analisando as teorias retornadas pelos sistemas PFORTE_PI e PFORTE para o *dataset* Amine, exibidas nas Tabela 8.7 e 8.8 respectivamente, verificamos que enquanto o PFORTE decidiu excluir a cláusula

$great\_ne(A, B) : -alk\_groups(B, C), ring\_substitutions(A, D), ring\_subst\_6(A, C),$
$polarisable(C, D)$

o PFORTE_PI optou por alterar a cláusula

$$great\_ne(A, B) : -alk\_groups(B, C), ring\_substitutions(A, D)$$

incluindo o *predicado probabilístico inventado* $new1(A, B)$ como intermediário e em seguida alterando outras 5 cláusulas que tinham os mesmos antecedentes no corpo (cláusulas 7,9,11,12 e 18). A acurácia das duas redes foi a mesma, 70.07 e a avaliação da teoria retornada pelo PFORTE melhor, 0.919 comparado com 0.929 da teoria retornada pelo PFORTE_PI. Já a rede retornada pelo PFORTE tem 23% mais parâmetros probabilísticos do que a rede retornada pelo PFORTE_PI. Isso ocorreu devido ao fato de terem sido substituídos pelo *predicado probabilístico inventado* mais de 2 antecedentes em mais de 2 cláusulas. Logo, através desse fold percebemos que o benefício do *operador de compactação* é alcançado quando temos na teoria cláusulas que contenham no corpo antecedentes que aparecem no corpo da cláusula *ponto de revisão*.

Tabela 8.7: Teoria final retornada pelo nosso sistema PFORTE_PI para um dos folds do *dataset* Amine

19	$\mathbf{new1(A,B)} : -\mathbf{alk\_groups(B,C)}, \mathbf{ring\_substitutions(A,D)}.$
1	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}.$
2	$great\_ne(A,B) : -x\_subst(A,C,D), ring\_subst\_4(B,D).$
3	$great\_ne(A,B) : -r\_subst\_2(A,C), ring\_subst\_5(B,D).$
4	$great\_ne(A,B) : -alk\_groups(B,C), r\_subst\_3(A,D), ring\_substitutions(A,C).$
5	$great\_ne(A,B) : -x\_subst(A,C,D), r\_subst_3(B,E), ring\_substitutions(A,F).$
6	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C).$
7	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}, \mathbf{polarisable(C,D)}.$
8	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D).$
9	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}, \mathbf{h\_acceptor(C,D)},$ $\mathbf{r\_subst\_2(B,E)}, \mathbf{r\_subst\_2(A,E)}.$
10	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(B,D),$ $ring\_substitutions(A,E), ring\_subst\_2(A,F).$
11	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}, \mathbf{h\_acceptor(C,D)},$ $\mathbf{great\_h\_acc(D,E)}.$
12	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{h\_acceptor(C,D)}, \mathbf{great\_h\_acc(D,E)},$ $\mathbf{ring\_subst\_4(B,C)}.$
13	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(A,D),$ $ring\_substitutions(B,E).$
14	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E).$
15	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E),$ $ring\_subst\_3(B,F), gt(D,G).$
16	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E),$ $x\_subst(A,F,G).$
17	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,C), x\_subst(A,D,E),$ $ring\_subst\_4(A,F), r\_subst\_2(B,G).$
18	$\mathbf{great\_ne(A,B)} : -\mathbf{new1(A,B)}, \mathbf{ring\_subst\_6(A,C)}.$

Tabela 8.8: Teoria final retornada pelo nosso sistema PFORTE para um dos folds do *dataset* Amine

1	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,D)..$
2	$great\_ne(A,B) : -x\_subst(A,C,D), ring\_subst\_4(B,D).$
3	$great\_ne(A,B) : -r\_subst\_2(A,C), ring\_subst\_5(B,D).$
4	$great\_ne(A,B) : -alk\_groups(B,C), r\_subst\_3(A,D), ring\_substitutions(A,C).$
5	$great\_ne(A,B) : -x\_subst(A,C,D), r\_subst_3(B,E), ring\_substitutions(A,F).$
6	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C).$
8	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D).$
9	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,D), ring\_subst_6(A,E),$ $h\_acceptor(E,F), r_subst\_2(B,G), r\_subst\_2(A,G).$
10	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(B,D),$ $ring\_substitutions(A,E), ring\_subst\_2(A,F).$
11	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,D), ring\_subst\_6(A,E),$ $h\_acceptor(E,F), great\_h\_acc(F,G).$
12	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,D), h\_acceptor(E,F),$ $great\_h\_acc(F,G), ring\_subst\_4(B,E).$
13	$great\_ne(A,B) : -alk\_groups(B,C), alk\_groups(A,C), ring\_subst\_4(A,D),$ $ring\_substitutions(B,E).$
14	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E).$
15	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E),$ $ring\_subst\_3(B,F), gt(D,G).$
16	$great\_ne(A,B) : -alk\_groups(A,C), ring\_substitutions(B,D), ring\_subst\_4(A,E),$ $x\_subst(A,F,G).$
17	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,C), x\_subst(A,D,E),$ $ring\_subst\_4(A,F), r\_subst\_2(B,G).$
18	$great\_ne(A,B) : -alk\_groups(B,C), ring\_substitutions(A,D), ring\_subst\_6(A,E)..$

Capítulo 9

Conclusões

Nesta tese, nós propomos dois sistemas de revisão de modelos probabilísticos: DAHVI (REVOREDO, PAES, et al., 2009), o qual revisa redes Bayesianas e o PFORTE_PI (REVOREDO, PAES, et al., 2006; REVOREDO, PAES, et al., 2007), o qual revisa teorias probabilísticas de primeira-ordem. Em ambos os sistemas nós apresentamos (i) formas de identificar pontos potenciais na estrutura para serem modificados, chamados *pontos de revisão* e (ii) mecanismos para efetuar essas modificações, denominados *operadores de revisão*. A principal contribuição desta tese foi a definição de *operadores de revisão* que estendem a linguagem corrente, introduzindo novas estruturas ao modelo: (i) *variáveis não-observadas* no caso de redes Bayesianas e (ii) *predicados inventados probabilísticos* no caso de teorias probabilísticas de primeira-ordem. Outra contribuição desta tese foi a aplicação de técnicas de revisão de teoria aos programas Problog probabilísticos (DE RAEDT, KERSTING, et al., 2008).

Neste capítulo, concluímos o nosso trabalho e discutimos possíveis pesquisas futuras tanto em relação ao sistema DAHVI, seção 9.1, quanto ao sistema PFORTE_PI, seção 9.2.

9.1 Sistema DAHVI

Uma questão importante, mas desafiadora em aprendizado de redes Bayesianas é como aprender a estrutura da rede na presença de *variáveis não-observadas*. Nesta tese, nós introduzimos uma nova abordagem para aprender redes Bayesianas com *variáveis não-observadas*, baseado em revisão de teoria (WROBEL, 1996). Nós des-

crevemos um algoritmo, chamado DAHVI (**D***iscriminative* **A***pproach for* **H***idden* **V***ariable* **I***ntroduction*), o qual verifica se uma rede Bayesiana pode ser melhorada usando os exemplos para identificar *pontos de revisão*. Em seguida, usamos um novo *operador de revisão*, proposto por nós, para introduzir *variáveis não-observadas* nestes *pontos de revisão*, escolhendo o que proporciona maior ganho a rede Bayesiana de acordo com alguma função de avaliação probabilística. Essas *variáveis não-observadas* são introduzidas intermediando a variável *ponto de revisão* e seus pais. Os resultados experimentais mostraram que o nosso sistema é capaz de melhorar uma rede Bayesiana aprendida a partir de diferentes algoritmos de aprendizado disponíveis na literatura, encontrando os melhores resultados quando revisando a rede Bayesiana aprendida com o algoritmo SEM. Os experimentos também mostraram que um outro benefício do nosso algoritmo é a possibilidade de selecionar o melhor número de *variáveis não-observadas* à ser introduzida durante a execução do algoritmo, diferentemente do sistema SEM, que precisa desta informação a priori.

As *variáveis não-observadas* inseridas pelo DAHVI assumem domínio binário, já que assim aumentamos o número de parâmetros probabilísticos o mínimo possível. Uma extensão do sistema DAHVI é a flexibilização desta restrição, permitindo a determinação de outros domínios, como proposto em (ELIDAN, FRIEDMAN, 2005). Ao permitir a definição de domínios diferentes do binário, aumentamos mais o número de parâmetros probabilísticos. Uma forma de contornar isto é a utilização de outros *operadores de revisão*, como por exemplo, o de exclusão de arestas. Uma possível extensão, então do sistema DAHVI, é a consideração de outros *operadores de revisão*. Uma possibilidade, sem considerar técnicas de revisão, é após a inserção de uma *variável não-observada* executar um algoritmo de aprendizado de estrutura, como por exemplo o SEM, a partir desta rede. Dessa forma o algoritmo de aprendizado vai propor outras modificações em toda a estrutura da rede. Outra possibilidade, agora considerando técnicas de revisão, é fazer de forma similar ao que é feito em BANNER (RAMACHANDRAN, MOONEY, 1998): tentar utilizar os exemplos para selecionar as melhores alterações a serem feitas na rede.

Todos os *datasets* considerados neste trabalho para analisar os benefícios do nosso sistema DAHVI, foram *datasets* cujas as variáveis tinham domínio discreto.

Como trabalho futuro, pretendemos aplicar o algoritmo DAHVI a *datasets* contínuos. Para isto estudaremos métodos de discretização, como os utilizados no trabalho (TEIXEIRA, REVOREDO, et al., 2003)(REVOREDO, ZAVERUCHA, 2004).

Finalmente, uma consideração interessante que pode surgir em aprendendo generativo se não soubermos a priori quais são as possíveis *variáveis de consulta*. Estamos investigando heurísticas para selecionar variáveis neste caso, tais como as variáveis que mais contribuíram para a avaliação total ou as variáveis que foram pior classificadas.

9.2 Sistema PFORTE_PI

Outra contribuição desta tese foi a extensão do nosso sistema de revisão PFORTE, com a introdução de dois novos *operadores de revisão*, *operador de compactação* e *operador de incremento*, baseados em invenção de predicados. O objetivo dessa extensão era encontrar teorias probabilísticas de primeira-ordem mais acuradas, compactas e que representassem informações "escondidas". A base desses dois operadores é o mecanismo de reformulação 3.3, o qual tem por objetivo a simplificação da teoria, com a inclusão de um *predicado probabilístico inventado* intermediário, tornado-a mais compacta. Uma das diferença dos nossos operadores de revisão para o mecanismo de reformulação é que o último, assim como todos os mecanismos de invenção de predicados (ver seção 3.3 para mais detalhes) em ILP percorrem toda a estrutura da teoria para identificar aonde um *predicado probabilístico inventado* poderia ser introduzido, enquanto que os nossos *operadores de revisão* ao combinarem invenção de predicados com técnicas de revisão de teoria permitem que a introdução do *predicado probabilístico inventado* seja feita em pontos específicos da estrutura da teoria, ou seja nos *pontos de revisão*, tornando assim o mecanismo de invenção de predicados mais eficiente. Os resultados experimentais mostraram que tanto o *operador de compactação* quanto o *operador de incremento* são capazes de propor modificações à teoria de forma a encontrar uma teoria mais acurada e compacta. Foi mostrado também que se uma teoria com antecedentes do corpo da cláusula ponto de revisão ocorrerem no corpo de outras cláusulas da teoria, o *operador de compactação* obtém resultados melhores do que os *operadores de revisão*

usuais, como adição/exclusão de regras ou antecedentes. Dessa forma, é mais vantajoso revisar a teoria utilizando o sistema PFORTE_PI do que o sistema PFORTE. Além disso, foi mostrado também que o *operador de incremento*, mesmo quando a teoria não contém cláusulas com corpos comuns, o que não permitiria encontrar uma teoria mais compacta, pode ser mais vantagens do que os *operadores de revisão* usuais quando a revisão necessária for a especialização da cláusula *ponto de revisão*.

Em 8.2 vimos que o *operador de incremento* mostra-se vantajoso quando a revisão alternativa é a *adição de antecedentes*, de forma que com a aplicação do *operador de incremento* o número de parâmetros probabilísticos é reduzido. Nesses casos, a cláusula que define o *predicado probabilístico inventado* pode ser uma cláusula específica, podendo ter um corpo que não é encontrado em outras cláusulas, inviabilizando a substituição de antecedentes pelo *predicado probabilístico inventado*, o que acarretaria a diminuição, ainda maior, no número de parâmetros. Na tentativa de revolver esse problema, pretendemos analisar os benefícios de um novos operador: assim como o *operador de incremento* ele especializa a cláusula que define o *predicado probabilístico inventado*, mas antes ele busca por cláusulas da teoria que contenham o antecedente que é substituído na cláusula *ponto de revisão*, para então substitui-lo pelo *predicado probabilístico inventado*. Dessa forma, a especialização que a cláusula que define o *predicado probabilístico inventado* sofrer será refletida para outras cláusulas na teoria. Esse novo operador permite tanto a redução do número de parâmetros quanto a possibilidade de especialização de várias cláusulas ao mesmo tempo. Suponha, por exemplo, a teoria da Tabela 9.1, extraída do domínio da Família. Em todas as cláusulas está faltando o predicado $genero(A)$, o que impediria, por exemplo, que o exemplo $esposa(joao, maria) = verdadeiro$ fosse provado.

Tabela 9.1: Exemplo de teoria onde todas as cláusulas precisam ser modificadas para incluir o predicado $genero(A)$

1	$esposa(A, B) : -casado(A, B).$
2	$esposo(A, B) : -casado(A, B).$
3	$tio(A, B) : -casado(A, C), irma(C, D), pais(D, B).$
4	$tia(A, B) : -casado(A, C), irmao(C, D), pais(D, B).$

Assumindo a cláusula 1. como a cláusula *ponto de revisão*, o novo *operador de revisão* proporia a cláusula $new1(A, B) : -casado(A, B)$ e antes de especializar, diferente do *operador de incremento*, substituiria o antecedente $casado(A, B)$ por $new1(A, B)$ nas outras cláusula da teoria. A teoria proposta seria como a exibida na Tabela 9.2. Se todas as cláusulas fossem especializadas com a adição do antecedente $genero(A)$, teríamos 160 parâmetros probabilísticos (considerando domínio binário para todos os predicados Bayesianos). A teoria exibida na Tabela 9.2 tem 56 parâmetros probabilísticos. Além da redução no número de parâmetros probabilísticos, esse operador traz o benefício de reduzir o espaço de busca, já que enquanto o PFORTE necessitaria de 4 iterações para especializar todas as cláusulas, esse operador em uma única iteração já especializa.

Tabela 9.2: Teoria proposta considerando o novo *operador de revisão*

5	$new1(A, B) : -casado(A, B), genero(A).$
1	$esposa(A, B) : -new1(A, B).$
2	$esposo(A, B) : -new1(A, B).$
3	$tio(A, B) : -new1(A, C), irma(C, D), pais(D, B).$
4	$tia(A, B) : -new1(A, C), irmao(C, D), pais(D, B).$

Um outro operador que pode ser considerado é um operador baseado na abordagem *Bias Shift* vista na seção 3.3.2. Neste caso não introduziríamos um *predicado probabilístico inventado* intermediário, alteraríamos o conhecimento preliminar.

Uma possível alteração para o *operador de incremento* e para o *operador de compactação* é permitir que um subconjunto das cláusulas da teoria que contém os antecedentes que definem o *predicado probabilístico inventado* sejam modificadas, para que esses antecedentes sejam substituídos pelo *predicado probabilístico inventado*. A escolha de quais dessas cláusulas seriam modificadas seria feita de acordo com a função de avaliação.

Experimentos alterando a CPD da cláusula que define o *predicado probabilístico inventado* e da cláusula *ponto de revisão*, que teve antecedentes substituídos pelo *predicado probabilístico inventado*, mostraram a relevância de uma boa inicialização para estas CPDs, já que a avaliação das propostas é afetada. Dessa forma, um estudo aprofundado da melhor inicialização considerando que um novo predicado

foi introduzido na linguagem é importante e pode refletir em melhores resultados para os *operadores de revisão* que consideram invenção de predicados. Além disso, como estamos acrescentando um novo predicado a linguagem, um estudo de outras funções de avaliação que levem isso em consideração é interessante de ser estudado. A função de avaliação *Minimum Description Length*(MDL), por exemplo, penaliza teorias considerando a quantidade de parâmetros probabilísticos. Sendo que no primeiro momento a proposta dos nosso *operadores de revisão* sempre aumentam o número de parâmetros probabilísticos, já que eles criam uma nova cláusula. A redução ocorre posteriormente quando o *predicado probabilístico inventado* substitui antecedentes do corpo de outras cláusulas da teoria. Dessa forma, a função MDL no primeiro momento penalizaria esses operadores. Será que ela poderia ser alterada de forma que levasse em consideração que o aumento do número de parâmetro se deu porque foi introduzido um novo predicado?

Tanto o *dataset* Mutagenesis quanto o Amine são *datasets* totalmente observados. Um experimento artificial com parte do *dataset* da Família foi feito, onde retiramos valores de alguns predicados, transformando o *dataset* em um *dataset* parcialmente observado. Esse experimento fez com que a proposta de modificação retornada pelo *operador de compactação* em uma determinada iteração fosse a escolhida, o que não ocorria antes com o *dataset* completo. Dessa forma, é interessante analisar os resultados obtidos com a aplicação do PFORTE_PI em *datasets* incompletos.

Verificamos pelos resultados exibidos em 8, que os benefícios dos nossos *operadores de revisão* baseados em invenção de predicados são alcançados quando a teoria inicial tem cláusulas com antecedentes comuns no corpo. Logo se uma teoria inicial com essa característica for recebida, pode-se optar pelo sistema PFORTE_PI invés do sistema de revisão PFORTE. Além disso se for conhecida a necessidade de especialização de alguma cláusula, o sistema PFORTE_PI também poderá ser mais vantajoso. Dessa forma, pretendemos aplicar o sistema PFORTE_PI em outros *datasets*, onde a teoria inicial detenha essas características, para que assim verificaremos todas os benefícios do nosso sistema PFORTE_PI.

Uma possível extensão do sistema PFORTE_PI é a consideração de outras aridades do *predicate utility lattice* (MUGGLETON, 1993) para o *predicado proba-*

bilístico inventado.

Apesar de ter sido aplicado somente em datasets onde deseja-se aprender um único predicado (*class* para o *dataset* Mutagenesis e *great_ne* para o *dataset* Amine), os *operadores de revisão* podem ser aplicados a *datasets* como o da *Família*, onde existem diferentes predicados a serem aprendidos. Como um trabalho futuro pretendemos aplicar o PFORTE_PI a *datasets* reais com mais de um predicado.

Referências Bibliográficas

AHA, D., LAPOINTE, S., LING, C., , MATWIN, S., 1994, "Learning Recursive Relations with Randomly Selected Small Training Sets", In:Cohen, W., Hirsh, H., , editors, *Proceedings of the 11th International Conference on Machine Learning*, pp. 12–18.

ALLEN, T., GREINER, R., 2000, "Model selection criteria for learning belief nets: An empirical comparison", In:*Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, pp. 1047–1054.

BAIAO, F., MATTOSO, M., SHAVLIK, J., , ZAVERUCHA, G., 2003, "Applying Theory Revision to the Design of Distributed Databases", In:*Proceedings of the the 13th Int. Conference on Inductive Logic Programming,LNAI 2835, Springer Verlag*, pp. 57–74.

BAIN, M., MUGGLETON, S., 1992, "Non-monotonic learning", *Inductive Logic Programming.*

BEINLICH, I., SUERMONDT, G., CHAVEZ, R., , COOPER, G., 1989, "The ALARM monitoring system:A Case Study with Two Probablistic Inference Techniques for Belief Networks", In:*Proc. 2nd European Conference on AI and Medicine.*

BINDER, J., KOLLER, D., RUSSELL, S., , KANAZAWA, K., 1997, "Adaptive probabilistic networks with hidden variables", *Machine Learning*, v. 29, pp. 213–244.

BLOCKEEL, H., DE RAEDT, L., 1998, "ISIDD: An Interactive System for Inductive Database Design", *Applied Artificial Intelligence*, v. 5, n. 12, pp. 385–421.

BOSTROM, H., 1999, "Induction of Recursive Transfer Rules", pp. 52–62, Bled, Slovenia.

BRATKO, I., 1986, *PROLOG: Programming for Artificial Intelligence*, Addison-Wesley.

BREESE, J., GOLDMAN, R., WELLMAN, M., 1997, "Introduction to the special section on knowledge-based construction of probabilistic and decision models", *Cybernetics*, v. 24, n. 11, pp. 1577–1579.

BUNTINE, W., 1991, "Theory Refinement on Bayesian Networks", In:*Proceedings Seventeenth Conference Uncertainty in Artificial Intelligence*, pp. 52–60, San Mateo, CA.

CASANOVA, M., GIORNO, F. A. C., FURTADO, A. L., 1987, *Programação em Lógica e a Linguagem Prolog*, EDGARD BLUCHER.

CHICKERING, M., HECKERMAN, D., 1997, "Efficient approximations for the marginal likelihood of Bayesian Networks with hidden variables", *Machine Learning*, v. 29, pp. 181–212.

COOPER, G., 1990, "Computational complexity of probabilistic inference using Bayesian belief networks (research note)", *Artificial Intelligence*, v. 42, pp. 393–405.

COSTA, V. S., PAGE, D., QAZI, M., , CUSSENS., J., 2003, "CLP(BN): Constraint Logic Programming for Probabilistic Knowledge", In:*Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*, pp. 517–524.

COWELL, R. G., DAWID, P. P., LAURITZEN, S. L., et al., 1999, *Probabilistic Networks and Expert Systems*, Springer Verlag, New York.

DAGUM, P., LUBY, M., 1993, "Approximating probabilistic reasoning in Bayesian belief networks is NP-hard", *Artificial Intelligence*, v. 60, n. 1, pp. 141–153.

DAVIS, J., BURNSIDE, E., DUTRA, I., PAGE, D., RAMAKRISHNAN, R., COSTA, V. S., , SHAVLIK, J., 2005, "View Learning for Statistical Relational Learning: With an Application to Mammography", In:*the Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

DAVIS, J., ONG, I., STRUYF, J., BURNSIDE, E., PAGE, D., , COSTA, V. S., 2007, "Change of Representation for Statistical Relational Learning", In:*the Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI)*.

DE RAEDT, L., 1997, "Logical Settings for Concept-Learning", *Artificial Intelligence*, v. 95, n. 1, pp. 187–201.

DE RAEDT, L., BRUYNOOGHE, M., 1993, "A theory of clausal discovery", In:*Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pp. 1058–1063.

DE RAEDT, L., DEHASPE, I., 1997, "Clausal discovery", *Machine Learning*, v. 2.

DE RAEDT, L., KERSTING, K., KIMMIG, A., REVOREDO, K., , TOIVONEN, H., 2008, "Compressing probabilistic Prolog programs", *Machine Learning*, v. 70, n. 2-3, pp. 151–168.

DE RAEDT, L., KIMMIG, A., TOIVONEN, H., 2007, "ProbLog: A probabilistic Prolog and its application in link discovery", In:*Proc. IJCAI-2007*, pp. 2462–2467.

DE RAEDT, L., LAVRAC, N., 1993, "The many faces of inductive logic programming", In:Komorowski, J., Raś, Z., , editors, *Methodologies for Intelligent Systems*, volume 689 of *LNAI*, pp. 435–449, (Invited paper).

DE RAEDT, L., LAVRAC, N., 1996, "Multiple Predicate Learning in two Inductive Logic Programming Settings", *Journal on Pure and Applied Logic*, v. 4(2), pp. 227–254.

DE RAEDT, L., LAVRAC, N., DZEROSKI, S., 1993, "Multiple Predicate Learning", In:Bajcsy, R., , editor, *IJCAI93*, pp. 1037–1043.

DEGROOT, M. H., 1987, *Probability and Statistics*, Addison-Wesley.

DEMPSTER, A. P., LAIRD, N. M., RUBIN, D. B., 1977, "Maximum likelihood from incomplete data via the EM algorithm", *Royal Stat Soc*, v. 39, pp. 1–39.

ELIDAN, G., FRIEDMAN, N., 2005, "Learning Hidden Variable Networks: The Information Bottleneck Approach", *Journal of Machine Learning Research*, v. 6, pp. 81–127.

ELIDAN, G., LOTNER, N., FRIEDMAN, N., , KOLLER, D., 2001, "Discovering hidden variables: a structure-based approach", In:*Neural Information Processing Systems*, volume 13, pp. 479–485.

ELIDAN, G., NINIO, M., FRIEDMAN, N., , SCHUURMANS, D., 2002, "Data perturbation for escaping local maxima in learning", In:*National Conference on Artificial Intelligence*, pp. 132–139.

ESPOSITO, F., MALERBA, D., LISI, F., 2000, "Induction of Recursive Theories in the Normal ILP Setting: Issues and Solutions", In:Cussens, J., Frisch, A., , editors, *ILP00*, volume 1866 of *LNAI*, pp. 93–111.

FABIAN, I., LAMBERT, D. A., 1998, "First-Order Bayesian Reasoning", In:*Proceedings Eleventh Australian Joint Conference on Artificial Intelligence*, number 1502, pp. 131–142.

FLENER, P., YILMAZ, S., 1999, "Inductive Synthesis of Recursive Logic Programs: Achievements and Prospects", *Journal of Logic Programming*, v. 41, n. 2-3, pp. 141–195.

FOGEL, L., ZAVERUCHA, G., 1998, "Normal Programs and Multiple Predicate Learning", In:Page, D., , editor, *ILP98*, volume 1446 of *LNAI*, pp. 175–184.

FRIEDMAN, N., 1997, "Learning belief networks in the presence of missing values and hidden variables", In:*14th International Conference on Machine Learning*, pp. 1252–133, San Francisco.

FRIEDMAN, N., 1998, "The Bayesian structural EM algorithm", In:*UAI*, pp. 129–138.

FRIEDMAN, N., GEIGER, D., GOLDSZMIDT, M., 1997, "Bayesian network classifiers", *Machine Learning*, v. 29, pp. 131–163.

FRIEDMAN, N., GETOOR, L., KOLLER, D., et al., 1999, "Learning Probabilistic Relational Models", In:*Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pp. 1300–1309, Stockholm, Sweeden.

FRIEDMAN, N., GOLDSZMIDT., M., 1996, "Building classifiers using Bayesian networks", In:*National Conference on Artificial Intelligence*, pp. 1277–1284.

FUHR, N., 2000, "Probabilistic datalog: Implementing logical information retrieval for advanced applications", *American Society for Information Science*, v. 51, pp. 95–110.

GARCEZ, A., ZAVERUCHA, G., 1999, "The Connectionist Inductive Learning and Logic Programming System", *Applied Intelligence*, v. 11, pp. 59–77.

GHAHRAMANI, Z., 1998, "Learning Dynamic Bayesian Networks", *Lecture Notes in Computer Science*, v. 1387, pp. 168–197.

GIORDANA, A., SAITTA, L., BAROGLIO, C., 1993, "Learning Simple Recursive Theories", In:*ISMIS '93: Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems*, pp. 425–434, London, UK Springer-Verlag.

GLOVER, F., LAGUNA, M., 1993, "Tabu search", *In.Reeves,editor, Modern Heuristic Techniques for Combinatorial Problems.*

GäRDENFORS, P., 1992, *Belif Revision*, Cambridge University Press, Cambridge,England.

GROSSMAN, D., DOMINGOS, P., 2004, "Learning Bayesian Network Classifiers by Maximizing Conditional Likelihood", In:*Proceedings of the 21th International Conference on Machine Learning (ICML-04)*, pp. 361–368.

HADDAWY, P., 1999, "An overview of some recent developments on Bayesian problem solving techniques", v. 20, n. 2, pp. 11–29.

HECKERMAN, D., 1995, "A tutorial on learning Bayesian networks.", Technical report, Microsoft Research.

HECKERMAN, D., 1998, "A tutorial on learning Bayesian networks.".

HINTON, G. E., 1986, "Learning distributed representations of concepts", In:*Proceedings of the Eigth Annual Conference of the Cognitive Science Society*, pp. 1–12.

IDESTAM-ALMQUIST, P., 1995, "Efficient Induction of Recursive Definitions by Structural Analysis of Saturations", In:De Raedt, L., , editor, *ILP95*, pp. 77–94 DEPTCW.

JAEGER, M., 1997, "Relational Bayesian Networks", In:*Proceedings of the Thirteenth Conference on Uncertainty in AI*, pp. 266–273.

JENSEN, F. V., 1996, *An Introduction to Bayesian Networks*, Springer Verlag, New York.

KERSTING, K., DE RAEDT, L., 2001a, "Adaptive Bayesian Logic Programs", pp. 104–117, Strasbourg, France.

KERSTING, K., DE RAEDT, L., 2001b, "Bayesian Logic Programs", Technical Report 151, University of Freiburg, Institute for Computer Science, Freiburg, German, , April.

KERSTING, K., DE RAEDT, L., 2001c, "Towards Combining Inductive Logic Programming with Bayesian Networks", In:*Proceedings of the 12th Int. Conference on Inductive Logic Programming, LNAI 2157 Springer Verlag*, pp. 118–131, Strasbourg, France.

KERSTING, K., DE RAEDT, L., 2002, "Basic Principles of Learning Bayesian Logic Programs", Technical Report 174, University of Freiburg, Institute for Computer Science, Freiburg, German.

KERSTING, K., DE RAEDT, L., 2005, *Bayesian Logic Programming: Theory and Tool*, 1 ed., Chapter to appear in L. Getoor and B. Taskar, editors, An Introduction to Statistical Relational Learning, MIT Press.

KHARDON, R., 2000, "Learning horn expressions with LOGAN-H", In:*Proceedings of the 17th International Conference on Machine Learning*.

KING, R. D., STERNBERG, M. J. E., SRINIVASAN, A., 1995, "Relating Chemical Activity to Structure: An Examination of ILP Successes.", *New Generation Computing*, v. 13, n. 3-4, pp. 411–433.

KOHAVI, R., 1995, "A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection", In:*Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI)*, pp. 1137–1145.

KOK, S., DOMINGOS, P., 2007, "Statistical Predicate Invention", In:*the Proceedings of the 24th International Conference on Machine Learning (ICML-2007)*.

KOLLER, D., 1999, "Probabilistic Relational Models", In:*Proceedings of the 9th Int. Conference on Inductive Logic Programming, LNAI 1634, Springer Verlag*, pp. 3–13.

KOLLER, D., PFEFFER, A., 1997, "Learning probabilities for noisy first-order rules", In:*Proceedings of the Fifteenth International Joint Conference on Artficial Intelligence (IJCAI-97)*, pp. 1316–1323.

KRAMER, S., 1995, "Predicate Invention: A Comprehensive View", Technical Report ÖFAI-TR-95-32, Austrian Research Institute for Artificial Intelligence.

LAM, W., BACCHUS, F., 1994, "Learning Bayesian Belief Networks: an approach based on the MDL principle", *Computational Intelligence*, v. 10, n. 4, pp. 269–293.

LANGLEY, P., 1995, *Elements of Machine Learning*, Morgan Kaufman, San Francisco.

LAPOINTE, S., MATWIN, S., 1992, "Sub-unification: A Tool for Efficient Induction of Recursive Programs", In:Sleeman, D., Edwards, P., , editors, *ML92*, pp. 273–281.

LAURITZEN, S., 1995, "The EM algorithm for graphical association models with missing data", *Computational Statistics and Data Analysis*, v. 19, pp. 191–201.

LLOYD, J., 1989, *Foundations of Logic Programming*, 2 ed., Springer Verlag.

MALERBA, D., 2003, "Learning Recursive Theories in the Normal ILP Setting", *Fundamenta Informaticae*, v. 57, pp. 39–77.

MCLACHLAN, G. J., KRISHNAN, T., 1997, *The EM algorithm and Extensions*, 1 ed., Wiley Interscience, New York.

MITCHELL, T., 1997, *Machine Learning*, McGraw-Hill, New York.

MUGGLETON, S., 1992, *Inductive logic programming*, McGraw-Hill, New York.

MUGGLETON, S., 1993, "Predicate invention and utility", *Experimental and Theoritical AI*, v. 1.

MUGGLETON, S., 1995, "Inverse Entailment and Progol", *New Generation Computing, Special issue on Inductive Logic Programming*, v. 13, n. 3-4, pp. 245–286.

MUGGLETON, S., 2002, "Learning structure and parameters of stochastic logic programs", In:*Proceedings of the 12th Int. Conference on Inductive Logic Prgramming,LNAI 2583, Springer Verlag*, pp. 198–206.

MUGGLETON, S., BRYANT, C., 2000, "Theory completion using Inverse Entailment", volume 1866, pp. 130–146.

MUGGLETON, S., BUNTINE, W., 1988, "Machine invention of first-order predicates by inverting resolution", In:*5th International COnference on Machine Learning* Morgan Kaufmann.

MUGGLETON, S., FENG, C., 1990, "Efficient induction of logic programs", In:*Proceedings of the 1st Conference on Algorithmic Learning Theory*, pp. 368–381 Ohmsma, Tokyo, Japan.

MURPHY, *Bayes Net Toolbox for Matlab. U.C. Berkeley.*, `http://www.cs.berkley.edu/murphyk/Bayes/bnt.html`.

MURPHY, K., 2001, "The Bayes Net Toolbox for Matlab", *Computing Science and Statistics*, v. 33.

NGO, L., HADDAWY, P., 1995, "Probabilistic logic programming and Bayesian networks", In:*In Algorithms, Concurrency and Knowledge: Proceedings of the Asian Computing Science Conference*, pp. 286–300, Pathumthai, Thailand, December.

NGO, L., HADDAWY, P., 1997, "Answering queries from context-sensitive probabilistic knowledge bases", *Theoretical Computer Science*, v. 171, pp. 147–177.

NIENHUYS-CHENG, S.-H., WOLF, R. D., 1997, *Foundations of Inductive Logic Programming*, Springer-Verlag, Germany.

PAES, A., REVOREDO, K., ZAVERUCHA, G., , COSTA, V. S., 2005a, "Further Experimental Results of Probabilistic First-order Revision of Theories from Exampes", In:*4th Workshop on Multi-Relational Data Mining / The Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 59, Chicago, Illinois ACM Press.

PAES, A., REVOREDO, K., ZAVERUCHA, G., , COSTA, V. S., 2005b, "Probabilistic First-Order Theory Revision from Examples", In:*Proceedings of the 15th International Conference on Inductive Logic Programming (ILP-05)*, volume 3625 of *LNAI*, pp. 295–311 Springer.

PAES, A., REVOREDO, K., ZAVERUCHA, G., , COSTA, V. S., 2006, "PFORTE: Revising Probabilistic FOL Theories", In:*Proceedings of the 18th Brazilian AI Symposium (SBIA-06)*, volume 4140 of *LNAI*, pp. 441–450 Springer.

PEARL, J., 1988, "Probabilistic Reasoning in Intelligent Systems", *Morgan Kaufmann*.

POOLE, D., 1993, "Probabilistic Horn abduction and Bayesian networks", *Artificial Intelligence*, v. 64, n. 1, pp. 81–129.

POOLE, D., 1998, "Learning, Bayesian Probability, Graphical Models, and Abduction", *In: Abduction and Induction: essays on their relation and integration*.

POOLE, D., 2000, "Abducing Through Negation as Failure: Stable models within the independent choice logic", *Journal of Logic Programming*, v. 44, pp. 5–35.

PRADHAM, M., DAGUM, P., 1996, "Optimal Monte Carlo estimation of belief network inference", In:*Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pp. 446–453.

QUINLAN, J., 1990, "Learning logical definitions from relations", *Machine Learning*, v. 5, pp. 239–266.

RAMACHANDRAN, S., MOONEY, R., 1998, "Theory Refinement of Bayesian Networds with Hidden Variables", In:*Proceedings of the fifteenth International Conference on Machine Learning (ICML)*, pp. 454–462.

REPOSITORIO UCI, *UCI Irvine Machine Learning Repository*, `http://archive.ics.uci.edu/ml/`.

REVOREDO, K., PAES, A., ZAVERUCHA, G., , COSTA, V. S., 2006, "Combining Predicate Invention and Revision of Probabilistic FOL theories", In:*Short paper proceedings of 16th International Conference on Inductive Logic Programming (ILP-07)*, pp. 176–178.

REVOREDO, K., PAES, A., ZAVERUCHA, G., , COSTA, V. S., 2007, "Combinando Invenção de predicados e revisão de teorias de primeira-ordem probabilísticas", In:*Anais do VI Encontro Nacional de Inteligência Artificial*.

REVOREDO, K., PAES, A., ZAVERUCHA, G., , COSTA, V. S., 2009, "A Discriminative Approach for Hidden Variable Introduction", In:*submitted*.

REVOREDO, K., ZAVERUCHA, G., 2002, "Revision of First-Order Bayesian Classifiers", In:*Proceedings of the 12th Int. Conference on Inductive Logic Programming, LNAI 2583,Springer Verlag*, pp. 223–237.

REVOREDO, K., ZAVERUCHA, G., 2004, "Search-based Class Fiscretization for Hidden Markov Model for Regression", In:*XVII Brazilian Symposium on Artificial Intelligence, 2004, São Luís. Lecture Notes in Artificial Intelligence*, volume 3171, pp. 317–325.

RICHARDS, B. L., MOONEY, R. J., 1995, "Automated Refinement of First-Order Horn-Clause Domain Theories", *Machine Learning*, v. 19, pp. 95–131.

RICHARDSON, M., DOMINGOS, P., 2006, "Markov logic networks", *Machine Learning*, v. 62, n. 1-2, pp. 107–136.

RUSSELL, S., NORVIG, P., 2002, *Artificial intelligence: A modern approach*, Englewook Cliffs, NJ: Prentice-Hall.

SATO, T., KAMEYA, Y., 2001, "Parameter learning of logic programs for symbolicstatistical modeling", *Journal of AI Research*, v. 15, pp. 391–454.

SCHWARZ, G., 1978, "Estimating the dimension of a model", *Annals of Statistics*, v. 6, pp. 461–464.

SRINIVASAN, A., 2001, *The Aleph Manual*.

SRINIVASAN, A., MUGGLETON, S., STERNBERG, M. J. E., , KING, R. D., 1996, "Theories for Mutagenicity: A Study in First-Order and Feature-Based Induction", *Artificial Intelligence*, v. 85, n. 1-2, pp. 277–299.

STAHL, I., 1996, "Predicate Invention in Inductive Logic Programming", In:Raedt, L. D., , editor, *Advances in Inductive Logic Programming*, pp. 34–47, Amsterdam IOS Press.

STAHL, I., WEBER, I., 1994, "The arguments of newly invented predicates in ILP", In:*Proc. of ILP-94*.

STERLING, L., SHAPIRO, E., 1986, *The Art of Prolog: Advanced programming Techniques*, The MIT Press.

STONE, M., 1977, "An asymptotic equivalence of choice of model by cross-validation and Akaike´s criterion", *Journal of the Royal Statistical Society series B*, v. 39, pp. 44–47.

TEIXEIRA, M. A., REVOREDO, K., ZAVERUCHA, G., 2003, "Hidden Markov Model for Regression in Eletric Load Forecasting", In:*10th International Conference on Neural Information Processing, 2003, Istanbul. ICANN/ICONIP-2003*, volume 1, pp. 374–377.

TOWELL, G., SHAVLIK, J., 1994, "Knowledge-Based Artificial Neural Networks", *Artificial Intelligence*, v. 70, n. 1–2, pp. 119–165.

WIRTH, R., O'RORKE, P., 1991, "Constraints on Predicate Invention", In:Birnbaum, L., Collins, G., , editors, *ML91*, pp. 457–461.

WOGULIS, J., PAZZANI, M., 1993, "A methodology for evaluationg theory revision systems: Results with Audrey II", In:*Proceedings of the Thirteenth International Join Conference on Artificial Intelligence*, pp. 1128–1134, Chambery, France.

WROBEL, S., 1994, "Concept Formation during Interactive Theory Revision", *Machine Learning*, v. 14, pp. 169–191.

WROBEL, S., 1996, "First-order Theory Refinement", *Advances in Inductive Logic Programming*, pp. 14–33.

Apêndice A

Tabelas com os Resultados Obtidos com o Sistema DAHVI

Neste anexo exibimos todos os resultados encontrados com o sistema DAHVI. Esses resultados foram os utilizados para gerar os gráficos de dispersão exibidos na Seção 6.3.

A.1 Resultados para o *Dataset* Stock

Nesta seção, são exibidos os resultados obtidos quando considerando como *variável de consulta* cada uma das 20 variáveis do *dataset* Stock. A Tabela A.1 exibe a log-verossimilhança da rede Bayesiana retornada por cada algoritmo de aprendizado (Naive Bayes (NB), *hill-climbing* (HC) e SEM) e da rede Bayesiana retornada pelo nosso algoritmo DAHVI quando aplicado em cada uma das redes aprendidas ($DAHVI_{NB}, DAHVI_{HC}, DAHVI_{SEM}$ respectivamente) utilizando log-verossimilhança como função de avaliação. Os valores em negrito indicam diferenças estatisticamente significante utilizando *corrected t-test* com 95%.

Variável	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	$DAHVI_{SEM}$
1	-1.4113	-1.4113		-1.3330		-1.4395
2	-1.4817	-1.4817		-1.3293		-1.4510
3	-1.4266	-1.4267		-1.3324		-1.4338
4	-1.5613	$\mathbf{-1.4751}$		-1.3369		$\mathbf{-1.4357}$
5	-1.4316	$\mathbf{-1.4055}$		$\mathbf{-1,3247}$		$\mathbf{-1.4246}$
6	-1.5508	$\mathbf{-1.4348}$		$\mathbf{-1,3207}$		$\mathbf{-1.4208}$
7	-1.5524	$\mathbf{-1.4317}$		$\mathbf{-1,2875}$		$\mathbf{-1.4175}$
8	-1.5448	$\mathbf{-1.4289}$		-1.3360		$\mathbf{-1,4153}$
9	-1.4847	$\mathbf{-1.4128}$		$\mathbf{-1.3251}$		$\mathbf{-1.4139}$
10	-1.5434	$\mathbf{-1.4317}$	-1.3304	$\mathbf{-1.3211}$	-1.4431	$\mathbf{-1.4213}$
11	-1.5515	$\mathbf{-1.4374}$		$\mathbf{-1.3051}$		$\mathbf{-1.4008}$
12	-1.5790	$\mathbf{-1.4454}$		-1.3461		$\mathbf{-1.4112}$
13	-1.5642	$\mathbf{-1.4418}$		-1.3508		$\mathbf{-1.4312}$
14	-1.5586	$\mathbf{-1.4465}$		-1.3479		$\mathbf{-1.4256}$
15	-1.5683	$\mathbf{-1.4419}$		$\mathbf{-1.3040}$		$\mathbf{-1.4245}$
16	-1.5742	$\mathbf{-1.4529}$		$\mathbf{-1.3145}$		$\mathbf{-1.4067}$
17	-1.4943	$\mathbf{-1.4484}$		-1.3383		$\mathbf{-1.4057}$
18	-1.5563	$\mathbf{-1.4283}$		-1.3327		$\mathbf{-1.4288}$
19	-1.5670	$\mathbf{-1.4294}$		$\mathbf{-1.3251}$		$\mathbf{-1.4112}$
20	-1.5657	$\mathbf{-1,4507}$		$\mathbf{-1.3125}$		$\mathbf{-1.4135}$

Tabela A.1: Comparando a log-verossimilhança dos sistemas: NB X $DAHVI_{NB}$, HC X $DAHVI_{HC}$ e SEM X $DAHVI_{SEM}$

A.2 Resultados para todos os *Datasets*

Nesta seção são exibidos os resultados encontrados considerando todos os *datasets*.

As tabelas A.6, A.2 e A.4 exibem a média da acurácia (ACC) de teste encontrada para cada uma das redes Bayesianas aprendidas considerando os algoritmos de aprendizado Naive Bayes (NB), *hill-climbing* (HC) e SEM e para a rede Bayesiana retornada pelo nosso algoritmo DAHVI ($DAHVI_{NB}$, $DAHVI_{HC}$ e $DAHVI_{SEM}$ respectivamente) quando revisando cada uma dessas redes considerando acurácia, log-verossimilhança e log-verossimilhança condicional como função de avaliação respectivamente. A tabela A.3 exibe a log-verossimilhança quando usando esta função como função de avaliação. A tabela A.5 exibe a log-verossimilhança condicional quando usando esta função como função de avaliação.

DB	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	SEM_h	$DAHVI_{SEM}$
Audio	71.30	94.19	-	-	27,06	25,58	**37, 63**
BC	71,28	71,28	70,57	70.57	70,92	70,21	**74, 79**
BCW	96,57	95,29	68.71	68.71	80,43	65,43	**83, 00**
Car	**81, 18**	69,99	53,93	**70.04**	69,81	70,04	70,04
Lymph	80,96	81,43	57.53	59.16	53,25	55,39	**71, 04**
Nursery	89.71	79.00	89,84	89,95	39.50	41,59	39,77
Poper	64,44	70,00	71,11	72.22	71,11	71,11	71,11
Ptumor	44,55	48,16	15,97	**27, 84**	24,57	24,87	25,48
Tic	71,01	65,36	69.36	68.94	65,99	68,42	66,42
Zoo	98,00	99,00	97.17	99,00	49,33	54,83	**61, 83**

Tabela A.2: Comparando a acurácia dos sistemas, considerando log-verossimilhança como função de avaliação: NB X $DAHVI_{NB}$, HC X $DAHVI_{HC}$ e SEM X $DAHVI_{SEM}$

Dataset	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	$DAHVI_{SEM}$
Audio	-2,1526	-2,1750	-	-	-1,9650	-1,9558
BC	-1,0846	-1,0846	-1,0168	-1,0168	-1,2669	-1,2763
BCW	-1,2127	-1,2520	-1,4627	-1,4627	-1,6012	-1,4947
Car	-0,7848	$-\mathbf{0,7634}$	-0,8122	-0,8150	-0,8277	-0,8202
Lymph	-1,6923	-1,6923	-1,6629	-1,6629	-1,7235	-1,7219
Nursery	-0,9791	-0,9812	-0,9705	-0,9686	-1,0492	-1,0492
Poper	-0,8829	-0,8243	-0,9104	-0,9101	-0,8030	-0,8014
Ptumor	-1,3941	-1,3936	-1,1224	-1,1090	-1,1301	-1,1349
Tic	-1,0174	$-\mathbf{0,9581}$	-1,0018	-1,0033	-1,0213	-1,0147
Zoo	-1,1901	$-\mathbf{0,8950}$	-1,3347	-0,9581	-1,2429	-1,0815

Tabela A.3: Comparando a log-verossimilhança dos sistemas, considerando log-verossimilhança como função de avaliação: NB X $DAHVI_{NB}$, HC X $DAHVI_{HC}$ e SEM X $DAHVI_{SEM}$

DB	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	$DAHVI_{SEM}$
Audio	71.30	38.93	-	-	27,06	**40, 45**
BC	71,28	73,49	70,57	70,57	70,92	74,14
BCW	96,57	95,71	68,71	68,71	80,43	**87, 71**
Car	81,18	70,61	53,93	55,85	69,81	70,05
Lymph	80,97	84,29	57,53	59,16	53,25	**61, 30**
Nursery	**89, 71**	65,22	89,87	90,06	39,50	**60, 84**
Poper	64,44	**72, 22**	71,11	71,11	71,11	71,11
Ptumor	44,55	44,48	15,97	**27, 84**	24,57	25,17
Tic	71,01	65,05	69,36	71,29	65,99	67,02
Zoo	98,00	99,00	97,17	99,00	49,33	**62, 88**

Tabela A.4: Comparando a acurácia dos sistemas, considerando log-verossimilhança condicional como função de avaliação: NB X $DAHVI_{NB}$, HC X $DAHVI_{HC}$ e SEM X $DAHVI_{SEM}$

Dataset	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	$DAHVI_{SEM}$
Audio	-0,1083	-0,1792	-	-	-0,2382	$\mathbf{-0,1762}$
BC	-0,0572	-0,0586	-0,0602	-0,0602	-0,0607	$\mathbf{-0,0537}$
BCW	-0,0641	$\mathbf{-0,0157}$	-0,1518	-0,1518	-0,0374	$\mathbf{-0,0283}$
Car	-0,0838	$\mathbf{-0,0595}$	-0,0819	-0,0686	-0,0740	-0,0737
Lymph	-0,0608	-0,0666	-0,0596	-0,0653	-0,0880	-0,0676
Nursery	-0,1099	-0,1099	-0,0458	-0,0187	-0,0943	$\mathbf{-0,0590}$
Poper	-0,0564	-0,0701	-0,0600	-0,0600	-0,0684	-0,0604
Ptumor	-0,2044	-0,2073	-0,2263	-0,2493	-0,2557	-0,2497
Tic	-0,0647	-0,0637	-0,0584	-0,0582	-0,0622	-0,0627
Zoo	-0,1150	-0,1143	-0,0388	-0,0388	-0,1161	-0,1023

Tabela A.5: Comparando a log-verossimilhança condicional dos sistemas, considerando log-verossimilhança condicional como função de avaliação: NB X $DAHVI_{NB}$, HC X $DAHVI_{HC}$ e SEM X $DAHVI_{SEM}$

Dataset	NB	$DAHVI_{NB}$	HC	$DAHVI_{HC}$	SEM	$DAHVI_{SEM}$
Audio	71.30	**80.52**	-	-	24.57	25.78
BC	73.42	72.29	70.57	70.57	70.92	72.16
BCW	96.14	96.29	68.71	68.71	80.43	**92.29**
Car	81.18	82.42	53.93	**71.73**	69.81	70.04
Lymph	80.97	78.83	57.53	**68.96**	53.25	59.61
Nursery	89.71	82.85	89.84	88.83	39.50	**82.18**
Poper	64.44	68.89	71.11	72.22	27.06	**39.55**
Ptumor	43.14	43.74	15.97	**28.74**	71.11	71.11
Tic	71.01	70.49	69.36	68.10	65.99	65.36
Zoo	98	99	97.17	**99**	49.33	**67.83**

Tabela A.6: Comparando a acurácia dos sistemas, considerando acurácia como função de avaliação: NB X $DAHVI_{NB}$, HC X $DAHVI_{HC}$ e SEM X $DAHVI_{SEM}$