UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFAEL MADEIRA

MÉTODO DE UTILIZAÇÃO DOS ALGORITMOS DE APRENDIZAGEM BAYESIANA A PARTIR DA NETICA JAVA API

CRICIÚMA, JULHO DE 2008

RAFAEL MADEIRA

MÉTODO DE UTILIZAÇÃO DOS ALGORITMOS DE APRENDIZAGEM BAYESIANA A PARTIR DA NETICA JAVA API

Trabalho de Conclusão de Curso apresentado para a obtenção do grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientadora: Prof[a]. Msc. Priscyla Waleska Targino de Azevedo Simões

Co-orientador: Prof. Fabrício Giordani

CRICIÚMA, JULHO DE 2008

RAFAEL MADEIRA

Método de Utilização dos Algoritmos de Aprendizagem Bayesiana a Partir da Netica Java API

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

_____
Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

_____
Profª. MSc. Priscyla Waleska T. A. Simões (UNESC)
Orientador

_____
Prof. Fabricio Giordani (UNESC)
Co-Orientador

_____
Profª. MSc. Merisandra Côrtes de Mattos (UNESC)

_____
Profª. MSc. Cristiane Raquel Woszezenki (UNESC)

AGRADECIMENTOS

Agradeço primeiramente a Deus, por me dar luz, saúde e sabedoria para chegar até aqui, iluminando meu caminho nos momentos difíceis.

A minha querida mãe, minha base em tudo que faço, por seu apoio incondicional em todos os momentos de minha vida.

Aos meus amigos pelos momentos de apoio e descontração no decorrer do curso.

Aos meus orientadores Priscyla e Fabricio que forneceram todo o auxílio necessário para a realização e conclusão deste trabalho.

E a todos que de uma forma ou de outra contribuíram para que este sonho se tornasse realidade.

Muito obrigado.

RESUMO

Os sistemas inteligentes, ao processar o raciocínio especialista, muitas vezes têm de tomar decisões com base em informações incertas, incompletas ou até mesmo contraditórias, e para a modelagem da incerteza por aleatoriedade surgiram as redes bayesianas, utilizadas na representação do conhecimento de sistemas especialistas probabilísticos. Assim, a aquisição de conhecimento inerente a esse processo é uma tarefa que costuma ser realizada de forma tradicional a partir de entrevistas entre o engenheiro do conhecimento e o especialista e, a aprendizagem bayesiana então, tema dessa pesquisa busca facilitar esta atividade. Nesse contexto, essa pesquisa teve como objetivo estudar e aplicar os métodos de utilização dos algoritmos de aprendizagem bayesiana disponibilizados pela Netica Java API. A metodologia contou com as seguintes etapas: levantamento bibliográfico; estudo na Netica Java API; desenvolvimento de uma aplicação Java integrando-a por meio da Netica Java API; e, desenvolvimento de um módulo de aprendizagem bayesiana na aplicação. O resultado obtido com essa pesquisa refere-se a uma aplicação em Java desenvolvida no ambiente Netbeans IDE 6.0, que, por meio da Netica Java API, permite o uso dos algoritmos Counting Learning, EM e Gradient no processo de aprendizagem bayesiana, além de possibilitar também realizar inferências a partir da rede bayesiana gerada. Pode-se concluir que os objetivos foram alcançados e os resultados considerados satisfatórios.

Palavras chaves: Redes Bayesianas, *Shell* Netica, Netica Java API, Aprendizagem Bayesiana.

ABSTRACT

The expert systems while processing the specialist reasoning, have, many times, to take decisions based on some uncertain, incomplete, and even contradictory information, and for the modeling of the uncertainty by randomness, bayesian nets used in the representation of knowledge and the expert and, therefore, bayesian learning, which is theme of our study, tries to facilitate this activity. In this context, this research had as its goal to study and to apply the methods of use of algorithm of bayesian learning available by Netica Java API. The methodology used the following steps: bibliography survey, study at Netica Java API; development of a module of bayesian learning in application. The result from this research refers to a application in Java developed at Netbeans IDE 6.0 environment, which through Netica Java API allows the use of algorithm Counting Learning, Expectation Maximization and Gradient Descent in the process of bayesian learning, besides allowing the realization of inferences from the bayesian net generated. The goals were achieved and the results were considered satisfactory.

Key Words: Bayesian Nets, Shell Netica, Netica Java API, Bayesian Learning

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

AC	Aquisição de Conhecimento
API	*Application Programming Interface*
BC	Base de Conhecimento
EM	*Expectation Maximization*
GAO	Grafo Acíclico Orientado
IA	Inteligência Artificial
MD	Mineração de Dados
PC	Probabilidade Condicional
RB	Redes Bayesianas
SE	Sistema Especialista
SEP	Sistema Especialista Probabilístico
TPC	Tabela de Probabilidade Condicional
UFMS	Universidade Federal do Mato Grosso do Sul
UML	*Unified Modeling Language*
UNESC	Universidade do Extremo Sul Catarinense

SUMÁRIO

1 INTRODUÇÃO

Em aplicações baseadas em conhecimento os sistemas especialistas probabilísticos (SEP) se propõem a resolver problemas geralmente de natureza incerta. Nesses sistemas, uma das etapas mais trabalhosas é a aquisição de conhecimento, já que muitas vezes essa atividade pode tornar-se complicada para o engenheiro do conhecimento que busca compreender o raciocínio do especialista e traduzi-lo em probabilidades, consumindo dessa forma muito tempo em reuniões e entrevistas (MANARIN, 2004).

Estes sistemas costumam ter em sua base de conhecimento fatos e regras que representam o conhecimento de um especialista. A estes fatos e regras é associada à incerteza presente no domínio que são explicitadas por meio de valores de probabilidade (TESSARI apud RAMOS; AZEVEDO; LIMA, 2007).

Assim, as redes bayesianas referem-se a um modelo de representação do conhecimento que utiliza o teorema de Bayes para expressar as relações causais e valores de probabilidade de um determinado domínio. Lidam diretamente com probabilidades condicionais (PC), que são valores que se deseja conhecer sobre o domínio estudado (RUSSEL; NORVIG, 2004).

As PC que representam a base de conhecimento costumam ser implementadas em ferramentas como a *shell* Netica (NORSYS, 2006) ou o *Hugin* (HUGIN, 2006) utilizadas no desenvolvimento de redes bayesianas (RB). Observa-se que a base de conhecimento e o motor de inferência necessitam se comunicar com os demais módulos do SEP por meio de uma *Application Programming Interface* (API) e de uma *shell* de RB (LUNA, 2007).

A primeira API da s*hell* Netica foi desenvolvida para ser utilizada na linguagem C, e oferecia recursos para gerenciamento do ambiente Netica, de nós, de PC

e das relações de dependência entre os nós. Posteriormente, os recursos dessa API foram atualizados, e seu fabricante desenvolveu uma nova biblioteca de funções, denominada Netica Java API, que apresenta as funções originais, e novos recursos que visam facilitar sua utilização, além de ser desenvolvida conforme a filosofia e metodologia do Java e do paradigma de programação orientada a objetos (NORSYS, 2007).

Quando uma rede bayesiana é construída, algumas importantes questões são levantadas, como quais são as variáveis e os seus valores, qual sua estrutura gráfica e quais são suas probabilidades, assim, a aprendizagem em uma rede bayesiana significa responder essas questões.

Nesse sentido, o processo de aprendizagem quando feito de forma não automatizada pode se tornar caro e demorado, principalmente em domínios complexos e muito amplos. Assim, a aprendizagem bayesiana visa facilitar essa etapa, onde por meio de um algoritmo é realizada a busca automática das probabilidades da RB a partir de uma base de casos.

Existem vários algoritmos que se propõem a realizar essa aprendizagem, proporcionando um importante ganho de tempo no processo de aquisição e representação do conhecimento, agilizando a comunicação entre o engenheiro do conhecimento e o especialista na construção de sistemas especialistas probabilísticos.

Assim, essa pesquisa propõe o realiza o estudo dos algoritmos disponibilizados pela Netica Java API, além do desenvolvimento de uma aplicação que possibilite a aprendizagem bayesiana a partir de uma base de casos.

1.1 OBJETIVO GERAL

Aplicar métodos de utilização dos algoritmos de aprendizagem bayesiana disponibilizados pela Netica Java API.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos são:

a) compreender a construção de Sistemas Especialistas Probabilísticos e os processos envolvidos;
b) oferecer um mecanismo para aprendizagem bayesiana em Sistemas Especialistas Probabilísticos;
c) disponibilizar a documentação sobre os algoritmos de aprendizagem da Netica Java API.

1.3 JUSTIFICATIVA

No processo de tomada de decisão, especialistas do domínio de conhecimento muitas vezes têm de tomar decisões com base em informações incertas, incompletas ou até mesmo contraditórias. Para que um sistema inteligente seja confiável, este deve tratar com a incerteza com o mesmo detalhamento de um especialista humano. Assim, a necessidade de tratar a incerteza em sistemas especialistas levou a construção de sistemas especialistas probabilísticos (TESSARI, 1998 apud RAMOS; AZEVEDO; LIMA, 2007).

A representação do conhecimento em sistemas especialistas probabilísticos é realizada por meio das redes bayesianas, que são um formalismo para representar a distribuição das probabilidades e a relação de interdependência entre as variáveis de um domínio de dados.

As redes bayesianas permitem ainda, analisar uma grande quantidade de dados, por exemplo, para extração de conhecimentos úteis em tomadas de decisão ou diagnóstico de causas de um fenômeno (WILLIAMSON, 2005).

Pode-se observar que existem poucas aplicações de descoberta de conhecimento aplicadas à construção de redes bayesianas, assim a necessidade de extrair informações de bases de dados vem aumentando a cada ano, devido ao fato do crescimento exponencial dessas bases. Com isso, torna-se cada vez mais difícil a análise dos dados, e necessita-se estudar ferramentas e técnicas que permitam extrair conhecimento a partir dessas grandes bases de dados (MANARIN, 2004).

No Brasil, muitas aplicações que utilizam redes bayesianas para representação da base de conhecimento de SEP, utilizam a *shell* Netica e a Netica C API, por ser utilizada no ensino de redes bayesianas na disciplinas de graduação e pós graduação do cursos de Ciência da Computação, além de se apresentar como um recurso didático, de fácil utilização e que permite boa interação com os especialistas do domínio de aplicação (NASSAR, 2005).

A Netica Java API foi desenvolvida para SEP construídos em Java, e apresenta algumas vantagens em relação a outras linguagens, tais como: portabilidade, possibilita trabalhar e utilizar ferramentas como a *shell* Netica e é uma linguagem orientada a objetos (BILESIMO, 2007).

Os algoritmos existentes para a aquisição de conhecimento de forma automatizada na API da *shell* Netica são *counting*, *Expectation Maximization* (EM), e

Gradient Descent que são utilizados na *shell* em virtude do raciocínio simplificado do *counting,* e os demais afim de solucionar as limitações presentes nesse algoritmo (NORSYS, 2007).

Dessa forma, essa pesquisa justifica-se pelo estudo e o desenvolvimento de métodos de utilização dos algoritmos de aprendizagem bayesiana por meio da Netica Java API.

1.4 ESTRUTURA DO TRABALHO

O trabalho está estruturado em cinco capítulos. O Capítulo 1 apresenta uma visão geral sobre o tema proposto, os objetivos e justificativa dessa pesquisa.

No Capítulo 2 são abordados os sistemas especialistas probabilísticos, incerteza e as redes bayesianas.

A aprendizagem bayesiana é tema do Capítulo 3, sendo apresentada sua definição e algumas características do processo. Os principais algoritmos disponibilizados pela Netica Java API são apresentados no Capítulo 4, juntamente com os respectivos métodos disponibilizados por essa API para a aprendizagem bayesiana.

Alguns trabalhos correlatos são apresentados no Capítulo 5 que abrange algumas aplicações.

Por fim, no Capítulo 6 é descrito o trabalho desenvolvido, as etapas metodológicas seguidas durante a pesquisa bem como os testes e os resultados obtidos na aplicação.

2 SISTEMAS ESPECIALISTAS PROBABILÍSTICOS

Na vida real, especialistas humanos muitas vezes têm de tomar decisões com base em informações incertas, incompletas ou até mesmo contraditórias. Para que um Sistema Especialista (SE) seja confiável, este sistema deve tratar com a incerteza como um especialista humano. Foi a necessidade de tratar com a incerteza em SE que levou a construção dos Sistemas Especialistas Probabilísticos (SEP). A partir da década de 80 a pesquisa sobre raciocínio probabilístico em SE resultou na introdução das Redes Bayesianas (RB), que têm sua origem na teoria da probabilidade e são caracterizadas por um formalismo que representa o conhecimento no domínio e pelas incertezas associadas a este domínio (STEIN, 2000).

Estes sistemas têm em sua base de conhecimento fatos e regras que representam o conhecimento de um especialista. A estes fatos e regras é associada à incerteza presente no domínio e são explicitadas por meio de valores de probabilidade (TESSARI, 1998 apud RAMOS; AZEVEDO; LIMA, 2007).

Os SEP caracterizam-se como sistemas que reproduzem o conhecimento adquirido por um especialista ao longo dos seus anos de trabalho, buscando resolver problemas simulando o comportamento deste. Um SEP deve ser construído com o auxílio de um especialista humano, que fornecerá os dados para a construção da Base de Conhecimento (BC) (FERNANDES, 2003).

Um SEP pode ser aplicado em muitas áreas, destacando-se a manufatura, finanças e serviços (educação, engenharia, meteorologia, medicina, militar, entre outros.). O sucesso de um SEP em qualquer uma dessas áreas depende de fatores ligados à natureza do conhecimento envolvido, como confiabilidade, integridade, ambigüidade e estabilidade. O ideal é que as decisões fossem tomadas inteiramente por

fatores mensuráveis que não se alteraram ao longo do tempo. Mas no mundo real, entretanto, um especialista toma decisões com base em dados imprecisos, incompletos e dinâmicos (BARONE, 2003).

Ainda segundo Barone (2003), um SEP é composto basicamente dos seguintes componentes (Figura 1):

a) **módulo de aquisição de conhecimento**: é por meio deste módulo que é alimentada a base de conhecimento do SEP. São especificadas as probabilidades, a estrutura da rede Bayesiana e as relações de dependência;

b) **base de conhecimento:** armazena todo conhecimento do especialista a ser utilizado pelo mecanismo de inferência;

c) **módulo de explanação:** este módulo tem por fim demonstrar o raciocínio efetuado pelo SEP, auxiliando o usuário na construção da base de conhecimento;

d) **mecanismo de inferência:** é o núcleo do SEP. É responsável pelos cálculos das PC a partir das probabilidades informadas pela base de conhecimento, sendo realizado pela propagação das PC a partir do teorema de Bayes;

e) **interface:** responsável pela interação entre o sistema e o usuário.

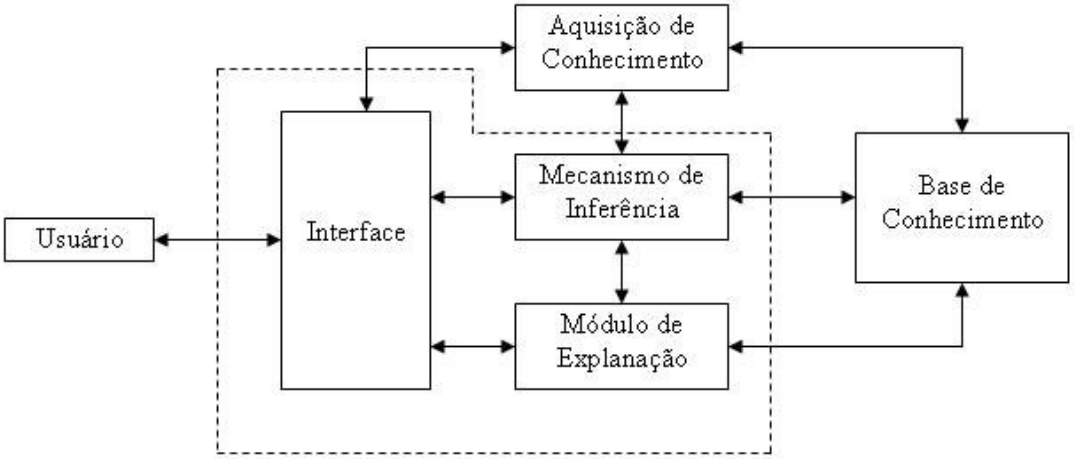


Figura 1. Arquitetura de um sistema especialista probabilístico
Fonte: LUGER, G. (2002).

Basicamente, a diferença entre um SEP para o SE clássico está no fato da base de conhecimento do SEP ser formada pela distribuição de PC. Essa base de conhecimento contém fatos e regras que representam o conhecimento do especialista, associadas as incertezas presente no domínio (NASSAR, 2007).

2.1 RACIOCÍNIO SOB INCERTEZA

Em afirmações lógicas, as regras booleanas indicam uma estrutura de representação que com tabelas verdade e diagramas de decisões binárias têm uma alta eficiência como algoritmo de inferência. No raciocínio lógico, são usados quatro elementos de lógica conectiva: conjunção, disjunção, implicação e negação. Por exemplo, em afirmações como “se está chovendo, o gramado está molhado”, e “o gramado não está molhado” é possível inferir que não está chovendo (JENSEN, 2001).

A incerteza é algo comum ao raciocínio humano, e em muitas situações práticas é raro encontrar um problema que envolva apenas fatos certos, completos e consistentes, tendo-se algumas formas distintas de incerteza que um sistema inteligente

precisa lidar: ignorância, já que o limite do conhecimento pode gerar incerteza sobre muitas coisas; indeterminismo, mesmo que haja conhecimento sobre tudo que pode importar para investigar sobre uma moeda e for conhecida a rotação quando lance ela, ainda restará um grau de incerteza sobre qual lado irá cair; e por fim, imprecisão em que muitos dos atributos que são empregados podem ser vagos (KORB; NICHOLSON, 2004).

Muitas técnicas foram desenvolvidas para tratar de incerteza no processo de tomada de decisão que foram sendo incorporadas aos sistemas especialistas. A análise das probabilidades aparece como o melhor caminho para tratar a incerteza por aleatoriedade em sistemas especialistas (BADIRU; CHEUNG, 2008).

Assim, uma abordagem geral para tratar com incerteza é o cálculo das probabilidades. Nessa abordagem o raciocínio é baseado na realização de inferência probabilísticas, isto é, no cálculo da PC de um evento, dadas todas as evidências disponíveis a partir do teorema de Bayes.

2.2 TEOREMA DE BAYES

Foi a partir da obra póstuma de Thomas Bayes publicada por Richard Price e intitulada como “Ensaio para resolver um problema na doutrina das probabilidades” que teve início a técnica chamada de estimação bayesiana. Esta técnica calcula a validade da probabilidade de uma proporção na base de uma estimativa prévia de sua probabilidade e nova evidência relevante, tendo como um dos teoremas mais famosos o Teorema de Bayes. Mais tarde, este teorema generalizado por Laplace, foi o ponto de partida para resolver problemas de inferência usando a teoria da probabilidade como lógica (STEIN, 2000).

Aplicada a partir da década de 60, a teoria probabilística de Bayes tratava da incerteza nos sistemas computacionais, mostrando uma maneira de calcular a probabilidade em um evento particular, dado um conjunto de observações que se tenha feito (BARONE, 2003). O teorema de Bayes dispõe a base necessária para o tratamento das incertezas na informação presente nos sistemas baseados em conhecimento.

Sua fórmula é descrita da seguinte forma:

$$P(Hi \mid E) = \frac{P(E \mid HI) * P(Hi)}{\sum P(E \mid Hn) * P(Hn)} \qquad (1)$$

Onde:

a) P(Hi | E) = a probabilidade de a hipótese Hi ser verdadeira, dada a evidência E,
b) P(E | Hi) = a probabilidade de observar a evidência E, dado que a hipótese Hi é verdadeira,
c) P(Hi) = a probabilidade *a priori* de Hi ser verdadeira na ausência de evidências específicas.

A partir do teorema de Bayes que foram criadas as Redes Bayesianas.

2.3 REDES BAYESIANAS

Uma Rede Bayesiana (RB) é um modelo de representação de conhecimento que utiliza o teorema de Bayes para expressar as relações causais e valores de probabilidade de um determinado domínio. As RB lidam diretamente com PC, que são os valores que se deseja conhecer sobre o domínio estudado (RUSSEL; NORVIG, 2004).

As RB's representam um Grafo Acíclico Orientado[1] (GAO) *G*. $G = (V,E)$, onde V e E são respectivamente o conjunto de vértices e arcos direcionados do grafo. O conjunto de vértices V apresenta as variáveis sobre as quais a RB é definida. Na Figura

2 é dado um exemplo de um GAO, e para cada variável $A \in V$, S especifica a distribuição de probabilidade de A condicionada aos arcos adjacentes a ele (WILLIANSON, 2005).

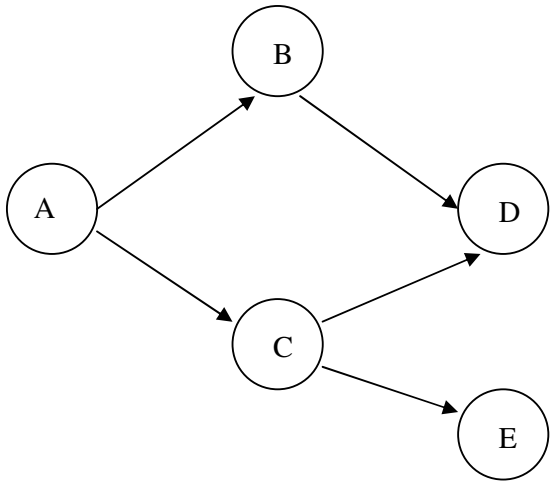


Figura 2. Exemplo de vínculo entre nós de uma RB.
Fonte:WILLIANSON, J. (2005)

Para que se possa verificar se um GAO representa uma RB existe uma condição necessária: cada variável X (nó X) do grafo deve ser condicionalmente independente de todos os nós que não são seus descendentes exceto seus pais, ou seja, se os valores dos nós diretamente conectados a uma variável são conhecidos, todos os outros nós do grafo são irrelevantes na definição do valor da variável em questão (HRUSCHKA, 2007).

Uma estrutura ou topologia de uma RB deve representar o relacionamento entre as variáveis. Dois nós podem se conectar diretamente se um é dependente do outro, onde uma seta indica a direção da dependência. Por exemplo, em um diagnóstico médico, quais são as evidências que podem influenciar na chance do paciente ter câncer? Se as respostas forem poluição e o tabagismo, devem ser traçadas setas ligando estes dois fatores ao câncer. Da mesma forma, se o câncer afetar a respiração do

paciente é adicionada uma seta ligando o câncer à dificuldade de respiração. A estrutura resultante destas relações pode ser vista na Figura 3 (KORB; NICHOLSON, 2004).

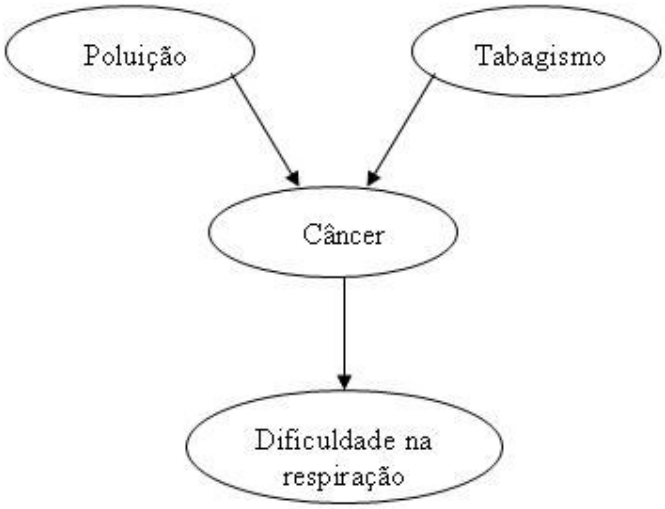


Figura 3. Estrutura de uma RB.
Fonte: Korb e Nicholson (2004)

Entre as topologias existentes para a representação de uma RB estão a serial, divergente e a convergente (Figura 4). Na conexão serial uma evidência em A se propaga a B que influencia a crença em C. Como não há uma canal ligando A a C estes nós tornam-se condicionalmente independentes. Na conexão divergente, uma evidência em um ascendente de A influencia a crença sobre os nós filhos (B, C, ... E). E na conexão convergente os pais do nó A são condicionalmente dependentes, pois uma evidência em A influencia na crença dos nós B e C (LADEIRA; VICARI; COELHO, 2007).

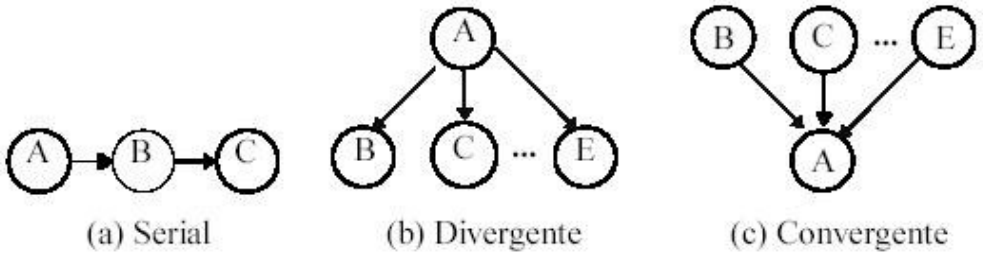


Figura 4. Exemplos de topologia de uma RB
Fonte: LADEIRA; VICARI; COELHO (2007)

A construção de uma RB pode ser dividida em duas fases: a primeira fase é a qualitativa, onde é definida a estrutura gráfica da RB e o relacionamento entre as variáveis, com a relação de dependência entre elas. A segunda fase é conhecida como quantitativa, onde são definidos os conjuntos de PC associadas as variáveis do modelo gráfico e as probabilidades estimadas *a priori* das hipóteses diagnósticas (COWEL; DAWID, 2008).

Em cada variável, são calculadas probabilidades por meio do Teorema de Bayes, permitindo definir hipóteses sobre cada variável e sua influência de acordo com as ligações existentes com as outras variáveis. Como forma de verificar as inferências em uma RB, são feitas suposições de valores para algumas evidências, observando como cada uma infere no valor da hipótese (BORGES; PADILHA, 2008).

Uma RB fornece uma descrição completa de um determinado domínio. Toda entrada na distribuição de probabilidade pode ser calculada a partir das informações armazenadas na rede. Cada distribuição de probabilidade é exibida sob a forma de uma Tabela de Probabilidade Condicional (TPC) (Tabela 1). Cada linha em uma TPC contém a PC de cada valor de nó para um caso de condicionamento que representa uma combinação de valores para os nós superiores (RUSSEL; NORVIG, 2004).

Tabela 1. Exemplo de uma Tabela de Probabilidade Condicional

Resfriado	Gripe	Malaria	Febre Presente	Febre Ausente
Nao	Sim	Sim	98.000	2.000
Nao	Sim	Nao	80.000	20.000
Nao	Nao	Sim	90.000	10.000
Nao	Nao	Nao	0.000	100.00
Sim	Sim	Sim	99.000	1.000
Sim	Sim	Nao	88.000	12.000
Sim	Nao	Sim	94.000	6.000
Sim	Nao	Nao	40.000	60.000

Fonte: RUSSEL, I; NORVIG, I. (2004)

As RB's permitem também analisar uma grande quantidade de dados, como para extração de conhecimentos úteis em tomadas de decisões ou diagnóstico de causas de um fenômeno. Sua utilização ocorre em vários domínios, como saúde (diagnósticos), indústria (controle de autômatos ou robôs) e finanças (análise financeira). Tendo em vista esse vasto campo de aplicações, o desenvolvimento de sistemas inteligentes que utilizam as RB como técnica para lidar com a incerteza tem crescido no meio acadêmico (WILLIAMSON, 2005).

Uma vez que a RB esteja definida, podem-se extrair os conhecimentos nela representados por meio de um processo denominado de inferência bayesiana.

2.4 INFERÊNCIA BAYESIANA

Após a construção da representação probabilística por meio do modelo de redes bayesianas (RB), para as incertezas presentes entre as variáveis de um domínio de dados, passa-se a tarefa de obter estimativas de probabilidades de eventos relacionados aos dados, à medida que novas informações ou evidências sejam conhecidas. Este processo é denominado como inferência bayesiana (LUNA, 2004).

A tarefa básica da inferência bayesiana é calcular a distribuição de probabilidade posterior para um conjunto de variáveis de consulta, dado algum evento observado, ou seja, alguma atribuição de valores a um conjunto de evidência (RUSSEL; NORVIG, 2004).

As PC são a base da inferência bayesiana, sendo que sempre que uma indicação de probabilidade $P(a)$ de um determinado evento a é dado, ele é condicionado a outros fatores conhecidos. Por exemplo, um dentista pode dizer: na minha opinião, eu acredito que a probabilidade de cárie seja de aproximadamente 0,1. $P(A \mid W)$ é o grau

de crença do especialista na veracidade da proposição *A*, dado sua experiência e conhecimento *W*. Nesse sentido, toda probabilidade é condicionada ao conhecimento do agente (JENSEN, 2001).

Em uma aplicação do teorema de Bayes são necessários três termos: uma PC e duas probabilidades incondicionais, tendo como resultado uma outra PC. A probabilidade incondicional ou probabilidade *a priori* associada a uma proposição *a* é o grau de crença acordado para a proposição na ausência de quaisquer outras informações, representada como *P(a).* Uma vez que o sistema obtém alguma evidência relativa às variáveis aleatórias que constituem o domínio, as probabilidades incondicionais não são mais aplicáveis. Em vez disso, usa-se a PC ou posteriores, cuja notação é *P(a | b)* onde *b* são proposições quaisquer (RUSSEL; NORVIG, 2004).

Por exemplo, em um diagnóstico médico, onde há as PC e deseja-se derivar o diagnóstico. O médico sabe que a meningite é uma doença que causa enrijecimento no pescoço do paciente em 50% dos casos. O médico conhece também alguns fatos incondicionais: a probabilidade incondicional de um paciente ter a meningite é 1/50.000, e a probabilidade incondicional de qualquer paciente ter um torcicolo é de 1/20. Sendo *T* a proposição que representa o paciente com torcicolo e *M* a proposição que representa o paciente com meningite, tem-se que (RUSSEL; NORVIG, 2004):

$$P(T \mid M) = 0{,}5 \qquad (2)$$

$$P(M) = \frac{1}{50000} \qquad (3)$$

$$P(T) = \frac{1}{20} \qquad (4)$$

$$P(M \mid T) = \frac{P(T \mid M) * P(M)}{P(T)} = \frac{0{,}5 * 1/50000}{1/20} = 0{,}0002 \qquad (5)$$

Conforme mostra a fórmula 5, isto significa que 0,0002 pacientes ou então, 1 em cada 5000 pacientes com torcicolo tenham meningite. É possível concluir ainda que mesmo que o torcicolo seja um forte indício de meningite, a probabilidade de ser

diagnosticada meningite continuará sendo pequena, uma vez que a probabilidade incondicional de torcicolo *(P (T))* é muito maior que a probabilidade incondicional de meningite *(P (M)).*

A obtenção das probabilidades condicionais pode ser realizada por meio de algoritmos a partir de uma base de casos, denominando-se esse processo de aprendizagem bayesiana.

3 APRENDIZAGEM BAYESIANA

O desempenho humano envolve o uso da aprendizagem, muitas vezes inconsciente, com base nas experiências vivenciadas no dia a dia. Este conhecimento adquirido muitas vezes é formatado como regras práticas que serão usadas em uma situação semelhante no futuro. Essas regras práticas são chamadas de heurísticas, e a obtenção dessas regras por um SEP é chamada de aprendizagem (BARONE, 2003).

A aprendizagem é uma das fases mais importantes na construção de um SEP. Esta etapa não é simplesmente adicionar novas informações à base de conhecimento, mas integrar toda nova informação com a já existente. Outro ponto importante na aquisição de conhecimento é o tratamento de incoerências, pois dependendo da forma como o novo conhecimento é adquirido, pode haver erros de aquisição. Assim, diversas técnicas foram desenvolvidas para evitar essas questões, como, por exemplo, a especificação de regras de aquisição em que o tipo de conhecimento esperado é definido (BITTENCOURT, 2001).

A construção manual de uma RB pode ser um processo trabalhoso e caro para grandes aplicações e em domínios complexos sua especificação além de consumir bastante tempo está propensa a erros. Por esse motivo os esforços dirigidos para o desenvolvimento de métodos que possam construir RB's diretamente de um banco de dados, ao invés do discernimento de especialista humanos vem crescendo constantemente (MATSUURA, 2008).

Quando uma RB é construída algumas importantes questões são levantadas, como quais são as variáveis e os seus valores, qual sua estrutura gráfica e quais os seus parâmetros (probabilidades). Algumas questões são adicionadas quando trata-se de redes de decisão, quais são as ações/decisões disponíveis, qual a utilidade dos nós e suas

dependências, e quais suas utilidades. A aprendizagem em uma RB significa responder a estas questões (KORB; NICHOLSON, 2004).

A aprendizagem bayesiana simplesmente calcula a probabilidade de cada hipótese, considerando-se os dados e faz previsões de acordo com ela. Desse modo, a aprendizagem é reduzida à inferência probabilística. Seja *D* a representação de todos os dados, com valor observado *d*, então a probabilidade de cada hipótese é obtida pela regra de Bayes, conforme Fórmula 6 (RUSSEL; NORVIG, 2004):

$$P(h_i \mid d) = \alpha P(d \mid h_i) P(h_i) \qquad (6)$$

O processo de aprendizagem determina automaticamente as probabilidades da RB a partir de uma base de casos. Cada caso dessa base representa um exemplo, dado, objeto ou situação real e fornecem os dados necessários para um conjunto de variáveis que descrevem esses eventos. Cada variável irá se tornar um nó da rede formada e os valores de cada variável serão os estados destes nós (NORSYS, 2007).

A rede formada pode ser usada para analisar novos casos que venham a ser adicionados a base. Normalmente esse novo dado irá fornecer valores para apenas algumas variáveis. Esses dados são adicionados como resultados, e depois é realizada uma inferência probabilística para determinar as crenças para o resto dos valores das variáveis deste caso (NORSYS, 2007).

As variáveis de uma base de casos podem ser de dois tipos: discreta ou contínua. Uma variável discreta é aquela que assume apenas valores enumeráveis dentro de uma faixa de variação, enquanto a variável contínua pode assumir qualquer valor real dentro do seu limite de variação. Os nós de uma RB podem representar tanto uma variável discreta, que assumem valores de um conjunto finito, quanto uma variável contínua, que assume valores de um conjunto infinito (MATSUURA, 2008).

A base de casos pode conter dados incompletos, ocultos ou faltantes. Nesse caso o algoritmo de aprendizagem necessita estimar esse dado para a construção da RB. Quando a variável é do tipo contínua não é possível realizar essa estimativa, sendo necessário dessa forma transformar a variável contínua em uma variável discreta, processo conhecido como discretização.

3.1 DISCRETIZAÇÃO

Muitos problemas reais envolvem dados continuos, como altura, massa, dinheiro e temperatura, e considera-se que grande parte da estatística lida com variáveis aleatórias cujos domínios são contínuos. Por definição, variáveis contínuas têm um número infinito de valores possíveis, e assim é impossível especificar explicitamente PC para cada valor. Um modo possível de manipular variáveis contínuas é evitá-las usando a discretização (RUSSEL; NORVIG, 2004).

A discretização pode ser definida como o processo de transformação de uma variável contínua discreta. Essa técnica consiste em unir valores adjacentes em intervalos. Basicamente cria-se uma variável discreta a partir de uma continua, e cada valor da variável discreta corresponde a um intervalo de valores da contínua. A variável discreta resultante desse processo é usada no lugar da contínua (MATSUURA, 2008).

A discretização divide os valores contínuos de uma variável em pequenas listas de intervalos. Efetivamente, converte atributos contínuos em categóricos, ou seja, cada intervalo resultante é considerado como um valor discreto do atributo. Na Figura 5 é ilustrada a discretização do atributo idade. Na parte inferior da Figura há uma lista ordenada de valores contínuos do atributo idade que foram discretizados em cinco intervalos (FREITAS; LAVINGTON, 1998 apud PAULA, 2008).

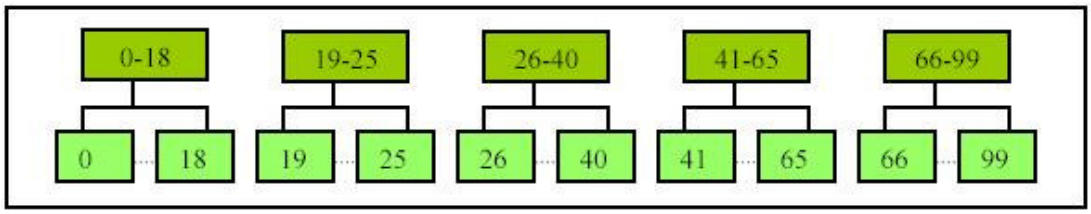


Figura 5. Discretização da variável idade.
Fonte: FREITAS; LAVINGTON, 1998 apud PAULA, 2008

A discretização torna o processo de aprendizagem mais simples e eficiente, pois diminui a necessidade de poder computacional para o processamento, armazenagem e tempo necessário para a construção de uma RB. Em muitos casos, o uso de variáveis discretas pode ainda resultar em uma RB mais adequada ao domínio do problema (MATSUURA, 2008).

Terminada a discretização os dados estão preparados para a fase seguinte que é a aprendizagem propriamente dita.

4 APRENDIZAGEM BAYESIANA NA NETICA JAVA API

Atualmente são disponibilizadas no mercado muitas ferramentas para a construção e gerenciamento de RB's. Dentre essas ferramentas destaca-se a *shell* Netica desenvolvida pela *Norsys Software Corporation* com sede em Vancouver, Canadá (BILÉSIMO, 2007).

A primeira API da s*hell* Netica foi desenvolvida para ser utilizada na linguagem C, com recursos para gerenciamento do ambiente Netica, de nós, de PC e das relações de dependência entre os nós. Posteriormente os recursos dessa API foram atualizados, e seu fabricante desenvolveu uma nova biblioteca de funções, denominada Netica Java API, que apresenta as funções originais, e novos recursos que visam facilitar sua utilização, além de ser desenvolvida conforme a filosofia e metodologia do Java e do paradigma de orientação a objetos (NORSYS, 2007).

A API da *shell* Netica é representada por uma biblioteca de funções que permitem o desenvolvimento de aplicações que utilizem RB's. Entre as funções disponíveis atualmente estão as de criação, modificação, gerenciamento, aprendizagem, teste e inferência de RB's (NORSYS, 2007).

Entre os arquivos presentes na Netica Java API pode-se citar alguns conforme tabelas abaixo:

Tabela 2. Conteúdo da pasta *doc* do pacote da Netica Java API

Diretório	Arquivo/site	Descrição
doc	NeticaJ_Man.pdf	Manual de utilização da Netica Java API
	NeticaAPIMan_C.pdf	Manual da API do Netica C que é base para a Netica Java API
	Javadocs http://www.norsys.com/netica-j/docs/javadocs/index.html	Documentação da Netica Java API
	LicAgree.txt	Documento sobre as licenças de utilização da Netica Java API

Fonte: BILESIMO, V (2007)

A pasta *doc* (Tabela 2) contém o manual de utilização da Netica Java API, com alguns exemplos e instruções de instalação; o manual da API do Netica em C, que possui uma visão geral de todas as funções disponibilizadas pela API; os *Javadocs*, que apresentam a documentação da Netica Java API; e o arquivo *LicAgree* que é um documento sobre as licenças de utilização da Netica Java API (NORSYS, 2007).

Tabela 3. Conteúdo da pasta *bin* do pacote da Netica Java API

Diretório	Arquivo	Descrição
bin	NeticaJ.jar	Classe com as bibliotecas de definições da Netica Java API
	Neticaj.dll	Biblioteca Nativa da Interface Windows
	Netica.dll	Biblioteca Nativa da Netica Java API

Fonte: Bilesimo, V (2007)

Na pasta *bin* (Tabela 3) encontram-se as bibliotecas necessárias para a utilização da Netica Java API. Essas bibliotecas são necessárias para a conFiguração do ambiente que utiliza a tecnologia Java e o ambiente de variáveis do sistema operacional *Windows* (BILÉSIMO, 2007).

Tabela 4. Conteúdo da pasta *examples* da Netica Java API

Diretório	*Arquivo*	*Descrição*
exemples / Data Files	ChestClinic.dne	Um exemplo utilizado pela classe ou arquivo SimulateCases / LearnCPTs / TestNet.java
	BreastCancer.dne	Um exemplo utilizado pela classe ClassifyData.java
	ChestClinic.cas	Um arquivo criado por SimulateCases.java e utilizado pelo TestNet.java
	ChestClinic_WithVisuals.cas	chestClinic.dne mas incluindo todas as dimensões / posição / cor para exibir as informações
	LearnLatent.cas	um processo utilizado pela LearnLatent.java
	BreastCancer.cas	um processo utilizado pela ClassifyData.java

Fonte: Bilesimo (2007)

No diretório *examples* (tabela 4) encontram-se alguns exemplos de RB's, destacando-se os arquivos com extensão *.cas* que se tratam de exemplos de bases de casos, utilizadas para a aprendizagem bayesiana.

Uma das limitações da Netica Java API é que ela não possibilita a aprendizagem da estrutura de uma RB, ou seja, as ligações de dependências entre os nós. Os algoritmos disponibilizados pela *Shell* permitem apenas a aprendizagem dos parâmetros da RB, isto é, as probabilidades da rede.

4.1 ALGORITMOS PARA APRENDIZAGEM BAYESIANA NA NETICA JAVA API

Existem três algoritmos disponibilizados pela *shell* Netica que permitem realizar a aprendizagem bayesiana a partir de uma base de casos: *Counting*, *Expectation Maximization* (EM) e *Gradient Descent*. Segundo o manual de referência da *shell*

Netica, não há um algoritmo que possa ser considerado melhor, sendo que cada um possui uma aplicação mais adequada em alguma situação (NORSYS, 2007).

4.1.1 Algoritmo *Counting Learning*

O algoritmo *counting learning* busca durante o processo de aprendizagem obter a máxima verossimilhança da RB, ou seja, a rede mais provável com os dados que foram passados. Se N é a rede e D são os dados, o algoritmo busca por N que apresente o maior P(N | D), usando a regra de Bayes, P(N | D) = P(D | N) P(N) / P(D). Sendo P(N | D) igual para todas as possíveis redes, é procurado maximizar P(D | N) P(N), por meio do logaritmo log (P(D | N)) + log (P(N)) (NORSYS, 2007).

Há diferentes abordagens para tratar com o segundo termo, log (P(N)), que é a probabilidade prévia de cada rede. Uma abordagem é de tratar cada rede como igualmente provável, sendo o termo simplesmente ignorado, pois irá contribuir da mesma forma para todas as rede prováveis, sendo esta abordagem a utilizada pelo algoritmo *counting* disponibilizado pela s*hell* Netica. A outra abordagem é a de ignorar as redes mais complexas tratando-as como menos prováveis. Se os dados D consistem em *n* casos independentes $d_1, d_2, ..., d_n$, então o termo log (P(D | N)) é: log (P(D | N)) = log ($P(d_1 | N) P(d_2 | N) \ldots P(d_n | N)$) (NORSYS, 2007).

Dos algoritmos apresentados pela *shell* Netica, o algoritmo *counting* é o mais simples e rápido, entretanto, possui como deficiência o fato de não trabalhar com base de dados que apresentem dados faltantes ou ocultos, sendo nesses casos necessária a aplicação dos algoritmos EM ou *Gradient Descent* (NORSYS, 2007).

4.1.2 Algoritmo Expectation Maximization *(EM)*

O algoritmo EM é um método para estimar funções de máxima verossimilhança a partir de dados incompletos. Em geral os parâmetros descrevem as características de determinado domínio de dados. O algoritmo estima os dados faltantes a partir dos parâmetros que sejam os mais consistentes (LUNA, 2007).

Cada iteração do algoritmo EM consiste em dois passos: o passo E onde são encontrados os valores esperados das estatísticas suficientes para os dados completos, com base nos dados incompletos e as estimativas atuais dos parâmetros; o passo M utiliza essas estatísticas suficientes para fazer uma estimativa de máxima verossimilhança como é usual (BORMAN, 2008).

Sendo x o conjunto de todos os valores observados, Z o conjuntos de todas as variáveis ocultas, e θ o conjunto de todos os parâmetros para o modelo de probabilidade, então tem-se a fórmula que representa o algoritmo EM (Fórmula 7) (RUSSEL; NORVIG, 2004):

$$\theta^{(I+1)} = \arg\max_{\theta} \sum_{z} P(Z = z \mid x, \theta^{(i)}) L(x, Z = z \mid \theta) \quad (7)$$

O passo E é o cálculo do somatório, que corresponde à probabilidade logarítmica dos dados completados com relação à distribuição $P(Z = z \mid x, \theta^{(i)})$, que é posterior sobre as variáveis ocultas, considerando-se os dados. O passo M é a maximização da probabilidade logarítmica esperada com relação aos parâmetros. Nas RB's as variáveis ocultas são os valores das variáveis não observadas (RUSSEL; NORVIG, 2004).

O algoritmo EM é indicado principalmente em casos onde nem todos os dados são conhecidos. É um algoritmo robusto porém pode se tornar lento. Uma alternativa mais rápida é o algoritmo *gradient descent* (NORSYS, 2007).

4.1.3 Algoritmo *Gradient Descent*

O algoritmo g*radient descent* ou gradiente descendente tomam por base um estado inicial e, incrementando e decrementando cada variável de certo valor determina por comparação ao desempenho do estado inicial a direção da próxima busca. O algoritmo continua até que a situação atual seja melhor que os cenários com incremento e decremento do passo. A *shell* Netica permite que esse passo seja alterada para que o processo seja agilizado. Este tipo de algoritmo garante a solução ótima para funções unimodais, ou seja, que não apresentam máximos ou mínimos locais (NORSYS, 2007).

Nesse algoritmo um vetor x é inicializado com um valor aleatório e, e posteriormente com um passo de cada vez, x é atualizado de acordo com a fórmula 8.

$$y = x - \alpha \nabla_x f(x) \quad (8)$$

A função $f(x)$ tende a diminuir ao longo do tempo. Ela pode, eventualmente, chegar próximo de um mínimo local, e, em seguida, começar a oscilar em torno de um mesmo valor. Para fazer $f(x)$ convergir, geralmente é necessário reduzir o ritmo da aprendizagem ao longo do tempo. Se o passo da aprendizagem for muito rápido, então x pode convergir para um ponto em que não é o mínimo local (BAIRD, 2008).

A partir da Equação 8 do algoritmo gradiente descendente, observa-se que existem dois parâmetros que podem gerar problemas no uso deste: o parâmetro de aprendizagem α e a função de $f(x)$. Ambos os parâmetros permitem um grau de escolha muito grande por serem parâmetros livres (podendo a princípio assumir qualquer valor)

e podem definir a velocidade de convergência do processo tomando o incremento *x* de maior ou menor intensidade (BURGES et al, 2008).

O parâmetro de aprendizado é o peso que o gradiente da função de custo terá na atualização do parâmetro x e esta escolha deve ser sempre levada em consideração. Se α é muito pequeno, a convergência de y é desnecessariamente lenta pelo fato do incremento de x ser muito pequeno, caso contrário, se α é muito grande, o termo x recebe um incremento também de grande valor e o processo pode divergir (BURGES et al, 2008).

A principal característica do algoritmo de gradiente descendente é sua velocidade, já que o passo pode ser alterado tornando-o mais ou menos veloz.

Os métodos para a utilização dos algoritmos de aprendizagem são disponibilizados na classe *learner* da API da *shell* Netica, sendo que esta classe, junto a classe *Caseset*, também foram objeto de estudo dessa pesquisa.

4.2 CLASSE *LEARNER*

Essa classe é responsável por disponibilizar as funções e métodos para realizar a aprendizagem bayesiana. Essa classe encontra-se no pacote *norsys.netica* como mostra a Figura 6.

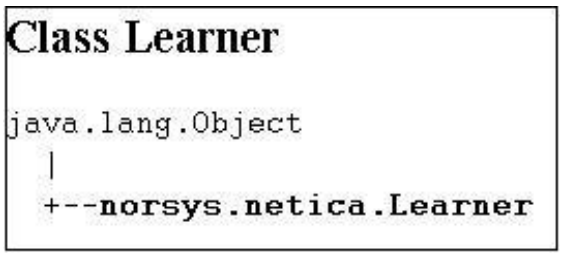


Figura 6. Classe *Learner*
Fonte: NORSYS (2007).

Dentre os métodos mais importantes desta classe estão os métodos *learner*, *setMaxIterations* e *learnCPTs*.

O método *learner* cria e retorna um objeto para uso na aprendizagem bayesiana que interage com o ambiente da *shell* Netica. Tem como parâmetros *int method* que define o algoritmo de aprendizagem que será utilizado, podendo ser os algoritmos EM *Learning, Gradient Descent Learning* e *Counting Learning*. O segundo parâmetro, *String Info*, será utilizado apenas em futuras expansões em versões futuras da *shell* Netica, devendo ser passado *null* na versão atual. O último parâmetro, *Environ env* é o ambiente da *shell* Netica.

O método *learnCPT's* é o responsável por realizar a aprendizagem. Ele recebe como parâmetro o atributo *NodeList*, que é a lista de nós e PC que serão atualizados após a aprendizagem; *CaseSet* que é o conjunto de casos que serão usados como base para a aprendizagem, e *degree* que é o grau de freqüência aplicado para cada conjunto de casos.

```
1       Learner learner = new Learner (Learner.EM_LEARNING, null, env);
2       learner.learnCPTs (nodes, cs, 1.0);
3       learner.finalize();
```

Figura 7. Exemplo de código utilizando a classe *Learner*

Na Figura 7 é apresentado um exemplo de código utilizando a classe *Learner*. Na linha 1 é criado um objeto do tipo *Learner*, sendo passado o algoritmo EM para a aprendizagem e o ambiente da shell Netica denominado env. Na linha 2 é utilizado o método *LearnCPTs*, sendo passado a lista de nós *nodes*, o conjunto de casos *cs* e o grau de freqüência igual a 1 para aplicação do conjunto de casos. Por fim a classe é finalizado na linha 3 com o método *finalize*.

4.3 CLASSE *CASESET*

Essa é uma classe utilizada pela classe *Learner*, que cria e manipula um conjunto de casos, e é encontrada no pacote *norsys.netica* como mostra a Figura 8. Tem como principais métodos *Caseset* e *addCases*.

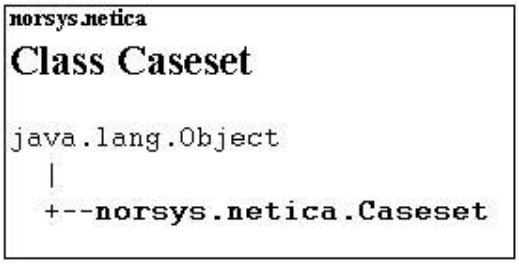


Figura 8. Classe *Caseset*
Fonte: NORSYS (2007)

O método *Caseset* cria e retorna um novo conjunto de casos, e recebe como parâmetros uma variável do tipo *string* que define o nome para esse conjunto, e o ambiente da *shell* Netica do tipo *Environ*.

A integração entre o arquivo de casos e o ambiente da *shell* Netica é realizado por meio do método *addCases.* Esse método recebe como parâmetros uma variável do tipo *streamer* com o endereço do arquivo *.cas* utilizado como base de casos, uma variável do tipo *double* que indica quantas vezes o arquivo deve ser utilizado e por fim uma variável para controle, que será utilizada em versões futuras, sendo passada como *null* na versão atual.

```
1     Caseset cs = new Caseset();
2     cs.addCases ("C:\\Aprendizagem\\Cases.cas", 1.0, null);
```

Figura 9. Exemplo de código utilizando a classe *Caseset*

Na Figura 9 temos um exemplo de utilização da classe *Caseset*. Na linha 1 é criado um novo conjunto de casos, enquanto na linha 2 esse conjunto é utilizado para a adição dos nós a partir do arquivo de casos, no exemplo o arquivo Cases *.cas*.

Outros estudos referentes a aprendizagem bayesiana e algoritmos de aprendizagem serão relatados no próximo capítulo.

5. TRABALHO CORRELATOS

Como já visto anteriormente as RB's destacam-se como um formalismo para a representação do conhecimento em domínios onde há a presença de incerteza. É possível construir uma RB a partir do conhecimento de um especialista, porém, dependendo do domínio a ser modelado, este pode ser um processo difícil e demorado. Definir cada uma das probabilidades condicionais para uma RB, cujo número de variáveis seja vasto, pode se tornar uma tarefa exaustiva se realizada a técnica tradicional de entrevista com especialista.

Assim, pode-se observar um crescente interesse no aperfeiçoamento e desenvolvimento de métodos para aprender estruturas e probabilidades a partir de uma base de dados. Nesse sentido, são apresentados alguns trabalhos correlatos desenvolvidos a partir do objeto de estudo dessa pesquisa.

5.1 MÉTODO PARA INTEGRAÇÃO DE UMA BASE DE CONHECIMENTOS DE UM SISTEMA ESPECIALISTA PROBABILÍSTICO UTILIZANDO A NETICA JAVA API

Esta pesquisa foi desenvolvida como Trabalho de Conclusão de Curso realizado na UNESC no ano de 2007, e teve como principal objetivo utilizar a Netica Java API como meio de comunicação entre a base de conhecimento e a interface com o usuário no desenvolvimento de um protótipo de um Sistema Especialista Probabilístico (BILÉSIMO, 2007).

Como resultados dessa pesquisa foram obtidos novas versões do Sistema Especialista para o Apoio ao Diagnóstico de Doenças Exantemáticas Maculopapulosas

com Erupção Obrigatória, Sistema Especialista Probabilístico para Prognóstico de Doenças Bucais, Biowoman – Base de Conhecimento Dinâmica para Sistemas Especialistas Probabilístico que haviam sido desenvolvidos anteriormente na UNESC (ANTUNES,2002; ROSA,2002; RODRIGUES,2002). A Figura 10 ilustra a interface obtida em um desses sistemas (BILÉSIMO, 2007).

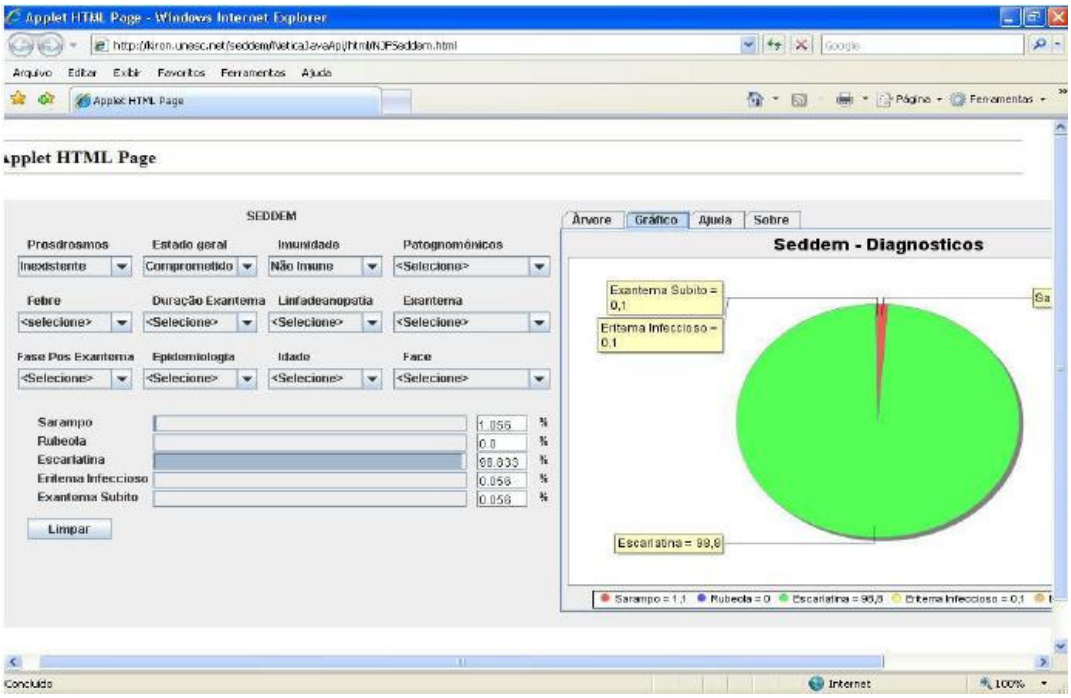


Figura 10. Interface inicial da applet do SEP SEDDEM
Fonte: BILÉSIMO, V. (2007)

5.2 AQUISIÇÃO DE CONHECIMENTO EM SISTEMAS ESPECIALISTAS PROBABILÍSTICOS POR MEIO DA DESCOBERTA DO CONHECIMENTO EM BASES DE DADOS PARA CONSTRUÇÃO DE REDES BAYESIANAS

Esse Trabalho de Conclusão de Curso desenvolvido no ano de 2004 na UNESC teve por objetivo realizar a aquisição do conhecimento por meio da descoberta do conhecimento em base de dados, para a construção de redes bayesianas (MANARIN, 2004).

A pesquisa consistiu na análise de versões *free* das ferramentas de aprendizagem em redes bayesianas *Belief Network Power Constructor* (BNPC), *Bayesian Knowledge Discoverer* (BKD) e *Hugin Expert.*

Após a análise comparativa, optou-se pela utilização da ferramenta BNPC, por oferecer uma interface intuitiva e por apresentar variadas opções de bases de dados. A partir de uma base de dados sobre Diabetes Mellitus tipo 2, foi gerada uma rede bayesiana que posteriormente, avaliada por uma especialista, foi considerada adequada, permitindo o relacionamento dos fatores de risco e complicações, sendo a interface principal da ferramenta ilustrada na Figura 11 (MANARIN, 2004).

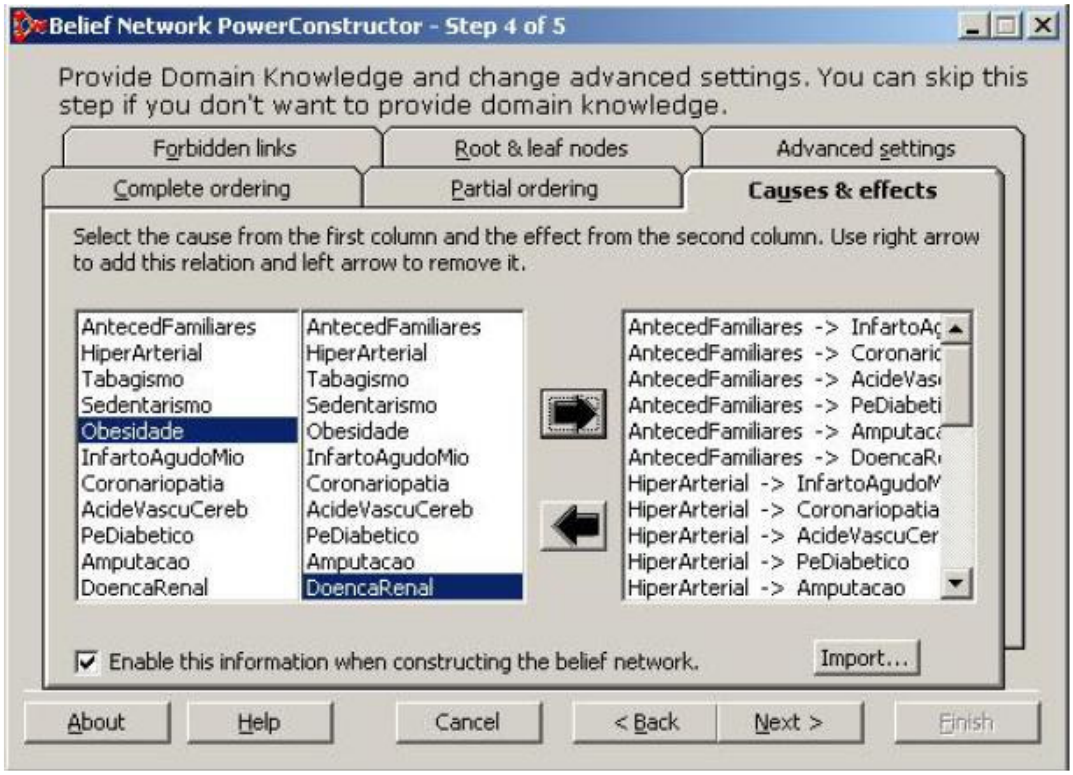


Figura 11. Especificação das relações de causa e efeito no BNPC
Fonte: MANARIN, D (2004)

5.3 MINERAÇÃO DE DADOS EM REDES BAYESIANAS UTILIZANDO A API DA *SHELL BELIEF NETWORK POWER CONSTRUCTOR* (BNPC)

Trabalho de Conclusão de Curso desenvolvido no ano de 2006 na UNESC, e teve como objetivo desenvolver o protótipo de um sistema para aquisição de conhecimento em sistemas especialistas probabilísticos que integre um ambiente de desenvolvimento com a API de uma *shell* de mineração de dados em redes bayesianas (GUINZANI, 2006).

Foram analisadas algumas API's das *shells* de mineração de dados em RB's, sendo elas Hugin Lite, UnBBayes e BNPC, escolhendo-se a última para realizar a integração com o ambiente de desenvolvimento *Visual Basic* (GUINZANI, 2006).

A pesquisa resultou em um protótipo denominado VisionBayes que utilizou os recursos dessa API para a aprendizagem automatizada em RB's, conforme ilustra sua interface principal na Figura 12(GUINZANI, 2006).

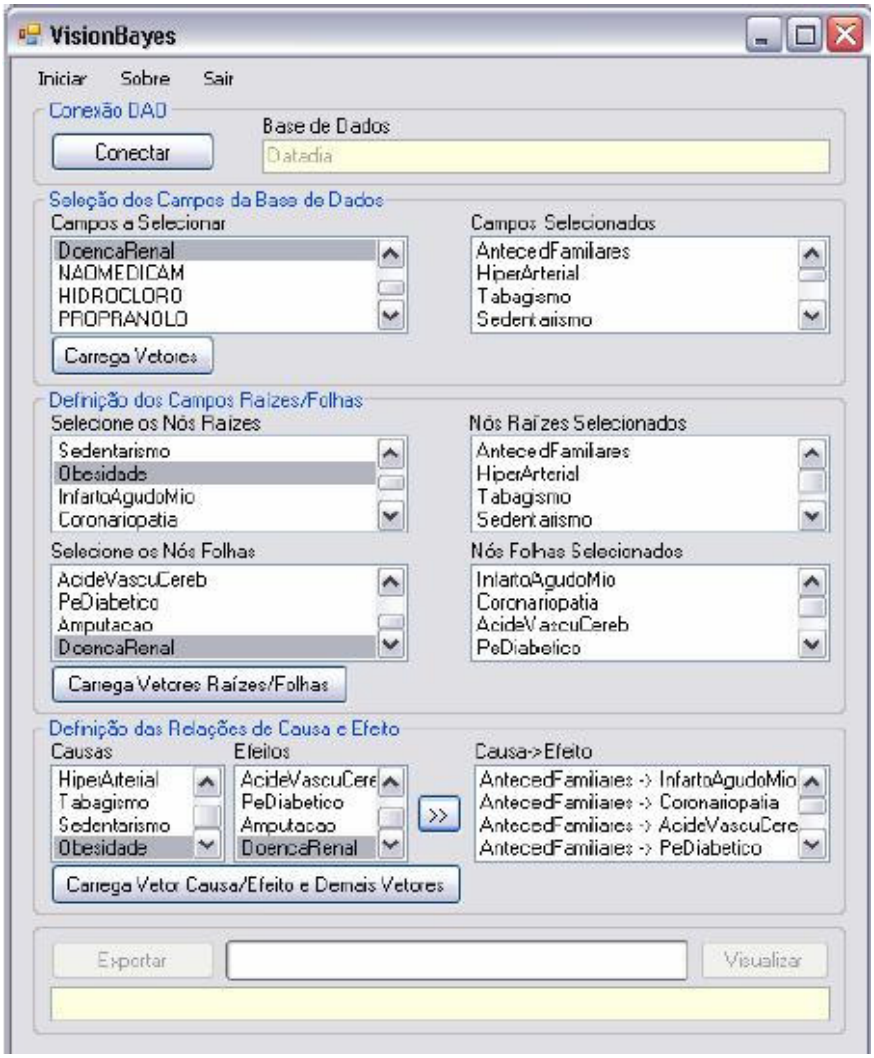


Figura 12. Interface principal do sistema VisionBayes
Fonte: GUINZANI, J. (2006)

5.4 ALGORITMOS EM PARA APRENDIZAGEM DE REDES BAYESIANAS A PARTIR DE DADOS INCOMPLETOS

Esta pesquisa foi tema de Dissertação de Mestrado apresentado ao Departamento de Computação e Estatística do Centro de Ciências Exatas e Tecnologia da Universidade Federal de Mato Grosso do Sul (UFMS) no ano de 2004, e teve como objetivo implementar algoritmos de aprendizagem para redes bayesianas a partir de dados incompletos baseando-se no algoritmo EM (LUNA, 2008).

O trabalho aplicou o algoritmo EM paramétrico de Lauritzen[2] para completar os dados em casos onde houvesse informações faltantes, além de realizar a aprendizagem das probabilidades para o caso de uma estrutura qualquer. O processo começa a partir de uma estrutura aleatória, completa os dados a cada estrutura intermediária e aprende suas probabilidades usando o algoritmo EM (LUNA, 2008).

Com o estudo dos trabalhos correlatos teve início então o desenvolvimento do trabalho proposto.

6 MÉTODO DE UTILIZAÇÃO DOS ALGORITMOS DE APRENDIZAGEM BAYESIANA A PARTIR DA NETICA JAVA API

Conforme mencionado anteriormente, a obtenção dos dados de uma RB de forma automatizada tem se mostrado de grande valia, pois agiliza sua construção. Assim, o objetivo principal dessa pesquisa foi estudar os algoritmos disponibilizados pela Netica Java API sobre aprendizagem bayesiana e utilizá-los em uma aplicação desenvolvida na linguagem Java que permitisse a construção de uma RB a partir de uma base de dados.

Como forma de alcançar os objetivos definidos, algumas etapas metodológicas foram seguidas, sendo a primeira o levantamento bibliográfico. Entre as demais estão o estudo da Netica Java API e de seus algoritmos de aprendizagem. A etapa de desenvolvimento de uma aplicação em Java, integrada por meio da Netica Java API com um módulo de aprendizagem bayesiana no aplicativo, e a etapa de testes concluem a metodologia desta pesquisa.

6.1 LEVANTAMENTO BIBLIOGRÁFICO

O levantamento bibliográfico foi a etapa inicial dessa pesquisa científica e buscou nortear seu desenvolvimento. Neste estudo foram pesquisados os conceitos dos Sistemas Especialistas Probabilísticos, Redes Bayesianas, Aprendizagem Bayesiana e Algoritmos de Aprendizagem Bayesiana, baseando-se em bibliografias nacionais e internacionais como livros, artigos, trabalhos de conclusão de curso, teses e dissertações, entre outros.

Finalizado o levantamento bibliográfico passou-se a tarefa da escolha da base de casos, para o posterior desenvolvimento da aplicação proposta.

6.2 BASE DE DADOS

A base de casos escolhida para o estudo da ferramenta foi a mesma utilizada por Manarin (2004) e Guinzani (2006) em suas pesquisas, que relaciona os fatores de risco a determinadas complicações em pacientes com Diabete Mellitus tipo dois nos bairros de Rio Maina, São Sebastião, Mineira Velha, Renascer, Laranjinha, São Luiz e Boa Vista, da cidade de Criciúma localizada em Santa Catarina (MANARIN, 2004).

A escolha dessa base deve-se ao fato de se tratar de uma base médica, propícia a análise por meio de RB, e conseqüentemente, por representar um domínio de conhecimento com certo grau de incerteza por aleatoriedade.

A base conta com 379 registros e 40 campos com informações dos pacientes cadastrados, apresentando dados gerais, como nome, idade, sexo, nascimento; fatores de risco, como antecedentes familiares, hipertensão arterial, tabagismo, sedentarismo, classificação da obesidade; complicações, como infarto agudo do miocárdio, coronariopatias, acidente vascular cerebral, pé diabético, amputação, doença renal; e medicamentos antidiabéticos utilizados (MANARIN, 2004).

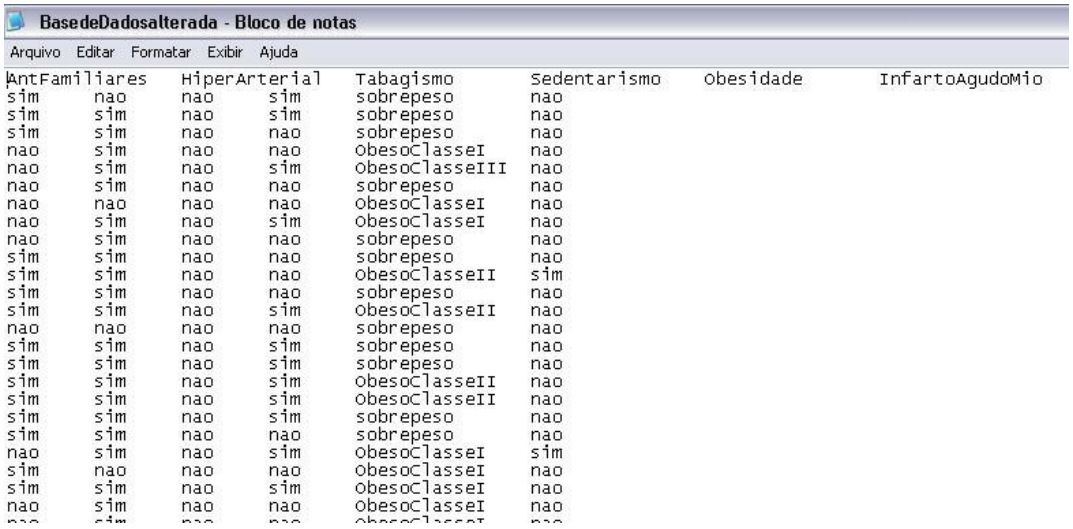
Outro fator decisivo na escolha dessa base deve-se ao fato desta já estar pré-processada por Manarin (2004), que realizou a limpeza, integração, seleção e transformação dos dados. Com relação aos dados ausentes, a base de dados possuía muitos campos não preenchidos que foram tratados, adicionando-se o valor não informado nesses campos. Como a *shell* Netica não permite nomes compostos (com

espaços entre eles), acentuados ou com caracteres especiais, também foram realizadas alterações para resolver essas questões (MANARIN, 2004).

Em sua pesquisa, Manarin (2004) selecionou os seguintes campos para o aprendizado da RB conforme sugestão de uma médica especialista: antecedentes familiares, hipertensão arterial, tabagismo, sedentarismo, obesidade, infarto agudo miocárdio, coronariopatias, acidente vascular cerebral, pé diabético, amputação e doença renal (MANARIN, 2004).

A API da *shell* Netica permite que a base de casos possa ser criada por meio de bancos de dados, planilhas do Microsoft Excel® ou um arquivo de texto. Nesta pesquisa, optou-se pela construção da base de casos em um arquivo de texto, pela facilidade de criação e manipulação da base por parte da aplicação desenvolvida, uma vez que dispensa o uso de funções adicionais para conexão com o banco de dados.

A formatação de uma base de casos consiste basicamente nas colunas representando os nós e nas linhas representando os casos (Figura 13). Os valores presentes na primeira linha são usados para identificação dos nós e nas linhas seguintes apresentam os estados possíveis para cada nó.


BasedeDadosalterada - Bloco de notas

Arquivo Editar Formatar Exibir Ajuda

AntFamiliares	HiperArterial	Tabagismo	Sedentarismo	Obesidade	InfartoAgudoMio
sim	nao	nao	sim	sobrepeso	nao
sim	sim	nao	sim	sobrepeso	nao
sim	sim	nao	nao	sobrepeso	nao
nao	sim	nao	nao	ObesoClasseI	nao
nao	sim	nao	sim	ObesoClasseIII	nao
nao	sim	nao	nao	sobrepeso	nao
nao	nao	nao	nao	ObesoClasseI	nao
nao	sim	nao	sim	ObesoClasseI	nao
nao	sim	nao	nao	sobrepeso	nao
sim	sim	nao	nao	sobrepeso	nao
sim	sim	nao	nao	ObesoClasseII	sim
sim	sim	nao	nao	sobrepeso	nao
sim	sim	nao	sim	ObesoClasseII	nao
nao	nao	nao	nao	sobrepeso	nao
sim	sim	nao	sim	sobrepeso	nao
sim	sim	nao	sim	sobrepeso	nao
sim	sim	nao	sim	ObesoClasseII	nao
sim	sim	nao	sim	ObesoClasseII	nao
sim	sim	nao	sim	sobrepeso	nao
sim	sim	nao	nao	sobrepeso	nao
nao	sim	nao	sim	ObesoClasseI	sim
sim	nao	nao	nao	ObesoClasseI	nao
sim	sim	nao	sim	ObesoClasseI	nao
nao	sim	nao	nao	ObesoClasseI	nao

Figura 13. Exemplo de formatação de uma base de casos

Na utilização dessa base a partir da Netica Java API procedeu-se na conversão do formato do *Microsoft Excel*® (.xls) para arquivo de texto com a extensão padrão utilizada pela *shell* Netica (*.cas).*

No desenvolvimento da aplicação optou-se por realizar a aprendizagem com apenas um nó raiz em virtude de limitações de criação de *links* da *shell* Netica. Dessa forma para os testes com a aplicação foi selecionado somente o infarto agudo do miocárdio como o nó raiz. Os demais nós que Manarin (2004) selecionou como raiz foram excluídos da base.

Com a base definida e pré-processada, passou-se a integração com a Netica Java API.

6.3 ESTUDO DA NETICA JAVA API

Conforme descrito anteriormente, a Netica Java API disponibiliza funções que possibilitam a uma aplicação desenvolvida em Java o manuseio de RB. Entre as funções disponíveis pela API estão as de criação e gerenciamento das redes, aprendizagem bayesiana, interação dos nós, entre outras.

Além das classes utilizadas na aprendizagem bayesiana já citadas anteriormente, outras que merecem destaque por serem posteriormente referenciadas no aprendizado bayesiano, mas que não se destinam exclusivamente a esse objetivo, são as classes *Environ*, *Net* e *Node* (BILÉSIMO, 2007).

6.3.1 Classes Pré Requisitos

A classe *Environ* possui métodos destinados a criação e utilização de um novo ambiente onde a RB será construída, podendo-se ilustrar o *Finalize* (Figura 14), que tem por objetivo liberar todos os recursos alocados na memória durante a execução da RB. Com o ambiente finalizado, nenhum método da Netica Java API poderá ser executado (BILÉSIMO, 2007).

```
1  protected void finalize () trows Throwable {
2      try {
3          env.finalize ();
4          }catch (NeticaException ne){
5              ne.printStackTrace ();
6          }
7  }
```

Figura 14. Método *finalize* da classe *Environ*
Fonte: BILESIMO, V (2007)

A criação de uma nova RB é realizada utilizando os métodos da classe *Net*, que buscam criar e executar a RB, destacando-se o construtor *Net* (*Streamer*) utilizado para a leitura de um arquivo desta rede. Na execução da leitura do arquivo, é retornada a referência da nova rede do arquivo ou uma exceção com a informação de que não foi possível ler o arquivo (Figura 15).

```
net = new Net (new Streamer (this.getClass().getResourceAsStream<"/arquivo.dne">, "<nome da RB>", <nome do ambiente>));
```

Figura 15. Construtor *Net* (*Streamer*)
Fonte: BILESIMO, V (2007)

Outro método importante desta classe é o *compile()*, responsável pela compilação e atualização das probabilidades na RB (Figura 16).

```
1   import norsys.netica.*;
2       public class DoInference {
3         public static void main (String[ ] args){
4          try {
5               Environ env = new Environ (null);
6               Net net = new Net (new Streamer ("DataFiles/ChestClinicBuilt.dne"));
7               Node visitAsia = net.getNode("VisitAsia");
8               Node tuberculosis = net.getNode("Tuberculosis");
9               Node xRay = net.getNode("XRay");
10              net.compile();
11              net.finalize();
12         }
13          catch (Exception e){
14            e.printStackTrace();
15           }
16      }
17  }
```

Figura 16. Método *Compile*
Fonte: NORSYS (2007)

A criação e manipulação dos nós é realizada utilizando os métodos da classe *Node*, destacando-se *getBelief* e *enterFinding*.

O método *getBelief* é utilizado para retornar a probabilidade de cada estado de um determinado nó da RB. Na Figura 17 é ilustrado um exemplo de aplicação desse método, onde uma variável *belief* do tipo *double* recebe a probabilidade do estado de *present* para o nó *tuberculosis* (BILESIMO, 2007).

```
1       Double belief = tuberculosis.getBelief ("present");
```

Figura 17. Método *GetBelief*
Fonte: NORSYS (2007)

A propagação das probabilidades condicionais na RB, dada uma nova evidência, é realizada com a aplicação do método *enterFinding*, conforme ilustra a Figura 18. Considerada a evidência informada no método, denominada Cariogênica do nó dieta, a inferência bayesiana é realizada para os demais nós da RB.

```
1    Dieta.enterFinding("Cariogenica");
```

Figura 18. Método enterFinding
Fonte: NORSYS (2007)

6.3.2 Classes de Aprendizado

Os métodos utilizados para a aprendizagem bayesiana já foram apresentados no Capítulo 4, e representam o objeto de estudo desta pesquisa. Por meio destes métodos a API da *shell* Netica permite que, a partir de uma base de casos, seja criada a rede bayesiana e a distribuição das probabilidades de acordo com a freqüência de valores encontrados nos estados do nó em cada um dos casos.

A classe *Learner* da Netica Java API disponibiliza os métodos, funções e construtores necessários para a realização da aprendizagem, destacando-se *learner* e *learnCPTs*. O construtor *Learner* cria e retorna um objeto para uso na aprendizagem bayesiana que interage com o ambiente da *shell* Netica, enquanto método *learnCPTs* é responsável por realizar a aprendizagem.

Outra classe utilizada no processo de aprendizagem é a classe *Caseset* que cria e manipula um conjunto de casos, e tem como principais métodos, *addCases* que cria o conjunto a partir da base de dados definida, e *o* construtor *Caseset*, que cria e retorna um novo conjunto de casos

Durante esse estudo verificou-se uma das limitações da *shell* e conseqüentemente da API, que indica a impossibilidade da aprendizagem automática da ligação entre os nós, sendo necessário realizá-la manualmente.

A partir do estudo da Netica Java API prosseguiu-se na metodologia com a modelagem do sistema desenvolvido.

6.4 MODELAGEM DO SISTEMA

A modelagem do sistema foi realizada por meio da *Unified Modeling Language* (UML)[3] pela ferramenta Pacestar[4], e teve por objetivo descrever a funcionalidade do sistema proposto, para melhor representação dos processos envolvidos. Na aplicação a interação entre usuário e interface ocorre como demonstra o seguinte diagrama de caso de uso (Figura 19).

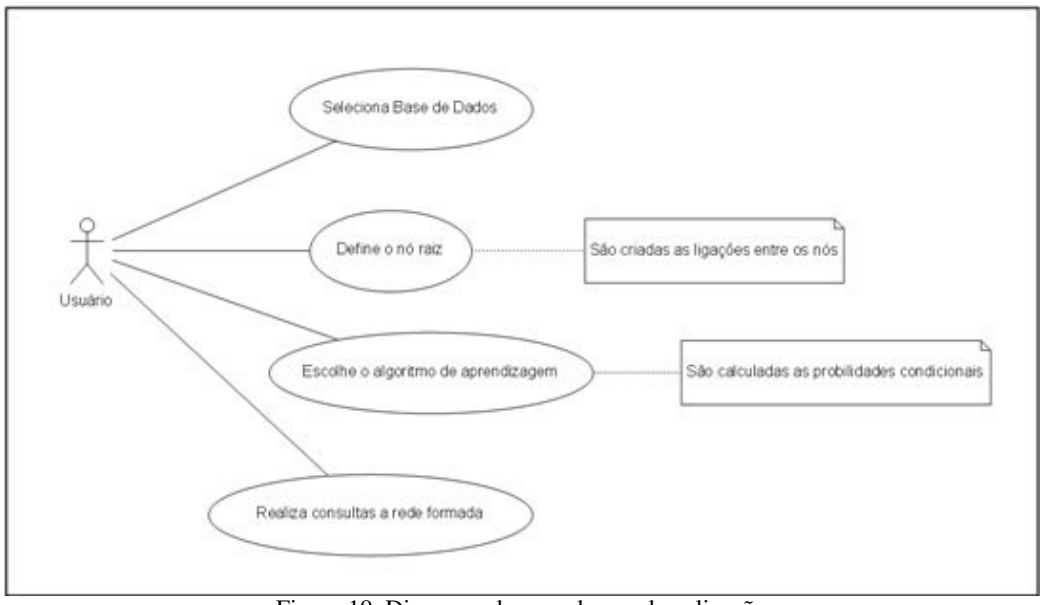


Figura 19. Diagrama de caso de uso da aplicação

O usuário primeiramente deve selecionar qual a base de casos que deseja utilizar no processo de aprendizagem. A rede então é carregada na aplicação e entre os nós que foram criados na rede, o usuário deve escolher qual é o nó raiz. As ligações são então criadas partindo dos nós filhos para o nó raiz.

Para a criação da TPC, o usuário deve escolher qual o algoritmo será usado na aprendizagem, a qual é então criada e a rede formada. O usuário pode então realizar

---

[3] Linguagem para visualização, especificação, construção e documentação de artefatos de um software em desenvolvimento (BEZERRA, 2003)
[4] Disponível em www.pacestar.com/

as inferências, selecionando as evidências enquanto o sistema propaga as probabilidades a cada seleção do usuário.

Com a modelagem do sistema concluída passou-se ao desenvolvimento da aplicação Java.

6.5 DESENVOLVIMENTO DE UMA APLICAÇÃO JAVA A PARTIR DA NETICA JAVA API

O aplicativo foi desenvolvido na linguagem Java, devido ao fato da API da *shell* Netica, que é objeto de estudo dessa pesquisa, ser adequada para aplicações implementadas na linguagem. Optou-se ainda pela utilização do ambiente de programação NetBeans IDE 6.0, em virtude de apresentar versão gratuita e por ser utilizada em algumas disciplinas do curso de Ciência da Computação da UNESC.

Ao inicializar a aplicação, o usuário deve selecionar a base de casos (Figura 20) que será utilizada no processo de aprendizagem. Em seguida, a ferramenta lê a primeira linha do arquivo de casos (nome dos campos) e define esses valores como os nós os quais serão apresentados ao usuário para escolha da raiz.

Figura 20. Leitura da base de dados

Após a escolha da raiz, é selecionado o algoritmo e a aprendizagem é realizada, sendo detalhada no próximo item.

6.6 DESENVOLVIMENTO DA APRENDIZAGEM BAYESIANA EM JAVA

Conforme apresentado anteriormente, a Netica Java API permite realizar a aprendizagem com três algoritmos diferentes que são utilizados pela API de forma semelhante. A fim de facilitar o entendimento das etapas envolvidas no processo de aprendizagem, foi elaborado um algoritmo ilustrado na Figura 21.

Passo 1 – Abrir o arquivo de casos

Passo 2 – Ler o arquivo de casos

Passo 3 – Criar os nós a partir do arquivo de casos

Passo 4 – Criar os estados dos nós

Passo 5 – Criar o nó raiz

Passo 6 – Criar as ligações entre os nós

Passo 7 – Escolher o algoritmo de aprendizagem

Passo 8 – Realizar a aprendizagem

Passo 9 – Salvar a rede

Figura 21. Algoritmo do processo de aprendizagem

6.6.1 Abertura e Leitura do Arquivo de Casos

O primeiro passo para a utilização da aprendizagem bayesiana em Java, a partir da Netica Java API, é a criação do ambiente Netica na aplicação e, dentro desse ambiente, criar a rede lógica que será usada para a aprendizagem (Figura 22).

```
1       try {
2               env = new Environ(null);
3               net = new Net (env);
4               net.setName("Infarto_MioCardio")
5       }catch (NeticaException ex) {
6               Logger.getLogger (NeticaEngine.class.getName()).log(Level.SEVERE, null, ex);
7       }
```

Figura 22. Criação do ambiente Netica

Na linha 2 o ambiente é criado, declarando uma variável da classe *Environ* e passando como parâmetro uma variável do tipo *string* com a licença de utilização da API. Esta licença é necessária apenas em RB's como mais de 16 nós, caso contrário o parâmetro *null* é informado. É criada então uma nova RB (linha 3), passando por parâmetro o ambiente criado anteriormente, e então a rede é nomeada como "Infarto_Miocardio" (linha 4). A rede lógica é criada vazia, ainda sem referência aos nós, que devem ser criados em seguida (Figura 23).

```
1       FileReader reader = new FileReader(arquivo);
2       BufferedReader leitor = new BufferedReader(reader);
3       int i=0;
4       String linha;
5
6       while ((linha = leitor.readLine()) != null) {
7
8               StringTokenizer st = new StringTokenizer(linha, "\t");
9               int coluna=0;
10
11              while (st.hasMoreTokens()) {
12
13                      String dados = st.nextToken();
14
15                      if (i == 0){
16
17                              cabecalho.add(dados);
18                              valores.put(coluna, new HashSet());
19                      } else {
20
21                              Set<String> valoresColuna = valores.get(coluna);
22                              valoresColuna.add(dados);
23                      }
24                      coluna++;
25              }
26              i++;
27      }
```

Figura 23. Carregar o arquivo de texto na aplicação

Conforme ilustra a Figura 23, para a criação dos nós primeiramente o arquivo da base de casos é carregado pela ferramenta (linha 1). Esse arquivo é associado a uma variável no ambiente (linha 2) e, em seguida, na linha 8, o arquivo é lido em blocos, sendo separado por tabulação. Os blocos da primeira linha do arquivo (linha 15), são gravados como sendo o nome dos nós da RB (linhas 17 e 18). No caso da base de texto utilizada nessa aplicação, é realizado um *looping* para associar todos os nós da rede, AntFamiliares, HiperArterial, Tabagismo, Sedentarismo, Obesidade e InfatoAgudoMio.

Os blocos seguintes (linha 19 - else) são representados como os estados de cada nó, ou seja, os valores “sim”, “sim”, “não”, “sobrepeso” e “não” são carregados, respectivamente, como estados dos nós citados acima. A próxima linha do arquivo de casos é lida e o processo torna-se iterativo, sendo que os casos repetidos, caso existam, não são gravados. Isso é realizado por meio da estrutura *set* (linha 21) que não permite que valores repetidos sejam gravados.

6.6.2 Adição dos Nós e Estados

Para a tarefa de adição dos nós, optou-se por utilizar uma estrutura de dados denominada mapa. Essa estrutura que possui a organização de um dicionário – chave, valor – foi escolhida devido à necessidade de ler cada bloco de texto, e atribuir esse valor como o nome da variável, ou seja, realizar a criação das variáveis e seus nomes de forma dinâmica. Com os dados gravados no mapa, os nós são criados na RB (Figura 24).

```
1       for (int j=0;j<cabecalho.size();j++){
2
3               String nomeNo = cabecalho.get(j);
4               Set<String> valoresPossiveis = valores.get(j);
5               int n = 0;
6
7               for (Iterator it = valoresPossiveis.iterator(); it.hasNext();) {
8                       String valor = (String) it.next();
9                       n++;
10              }
11
12              Node no = new Node(nomeNo, n, net);
13              nos.put(nomeNo, no);
14
15              int l = 0;
16
17              for (Iterator it = valoresPossiveis.iterator(); it.hasNext();) {
18                      String valor = (String) it.next();
19                      no.state(l).setName(valor);
20                      l++;
21              }
22      }
```

Figura 24. Código fonte da criação dos nós na RB

Os blocos de texto que foram gravados no mapa são lidos pela aplicação (linhas 3 e 4). O nome do nó é gravado em uma variável do tipo *String,* nomeada “nomeNo” (linha 3). Por exemplo, na base utilizada, o nome do primeiro nó é AntFamiliare, assim, nas linhas 7, 8 e 9 é realizada uma repetição para a contagem do número de estados deste nó, que é gravado em uma variável do tipo *int*, no caso 3 estados (por exemplo “sim”, “não” e “NaoInformado”).

O nó é criado na linha 12, passando como parâmetro o nome do nó, seus estados e a rede onde esse nó será criado, no caso a rede “net”. Os estados desse nó são lidos a partir da estrutura *Set* (linha 18) e gravados como estados do nó (linha 19).

Na *shell* Netica, esse processo é feito por meio da opção "*Add Case File Nodes"* presente no menu "*Cases"* (Figura 25).

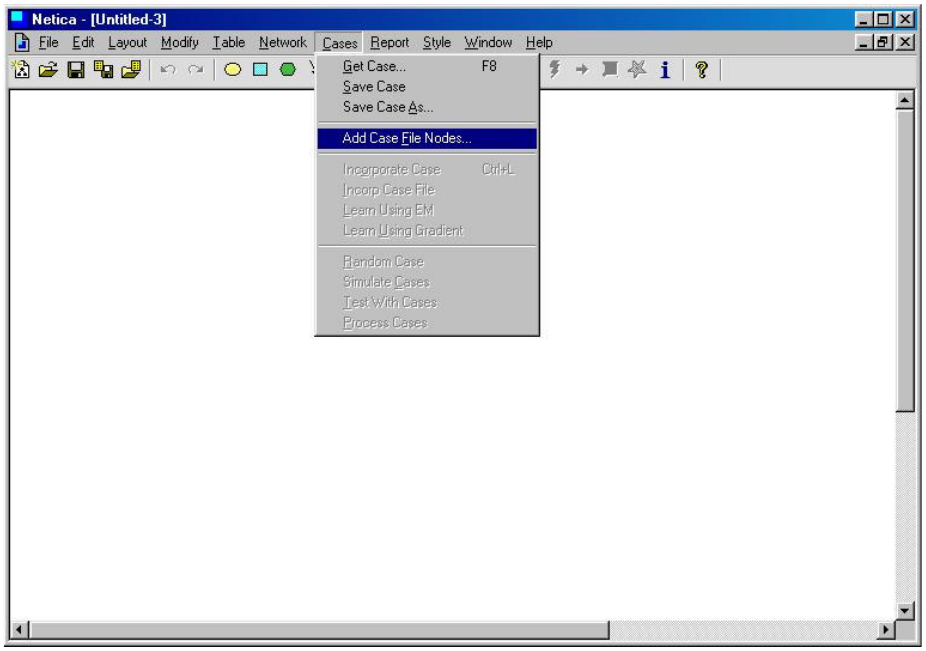


Figura 25. Criação dos nós na *shell* Netica

Em seguida, deve ser selecionada a base de casos que será usada na aprendizagem, por exemplo a base “Base InfartAgudoMio.cas” (Figura 26)

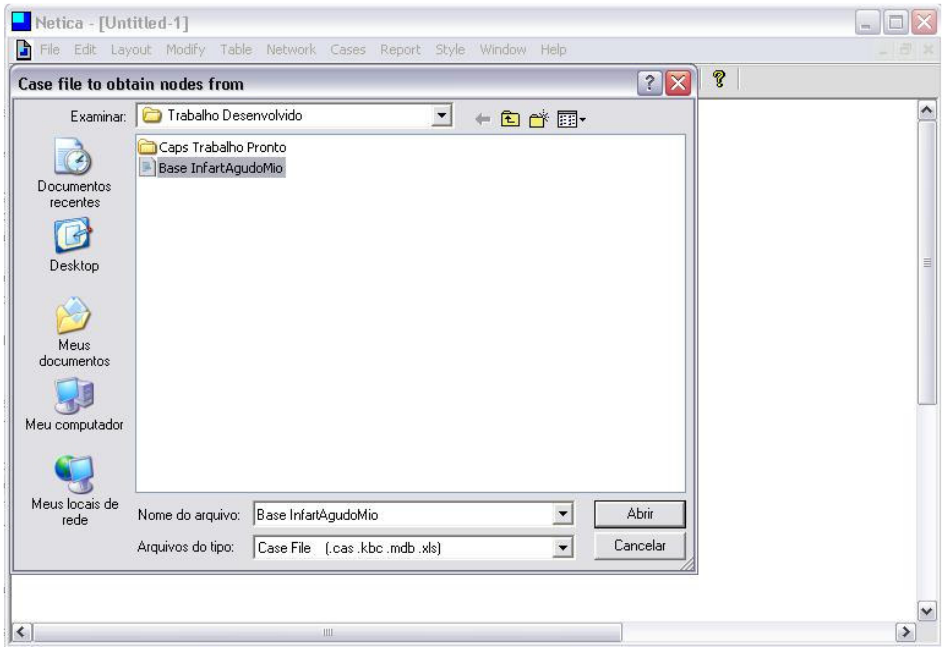


Figura 26. Seleção da base de casos

Os nós então são criados, porém, ainda sem as PC e as ligações entre eles (Figura 27).

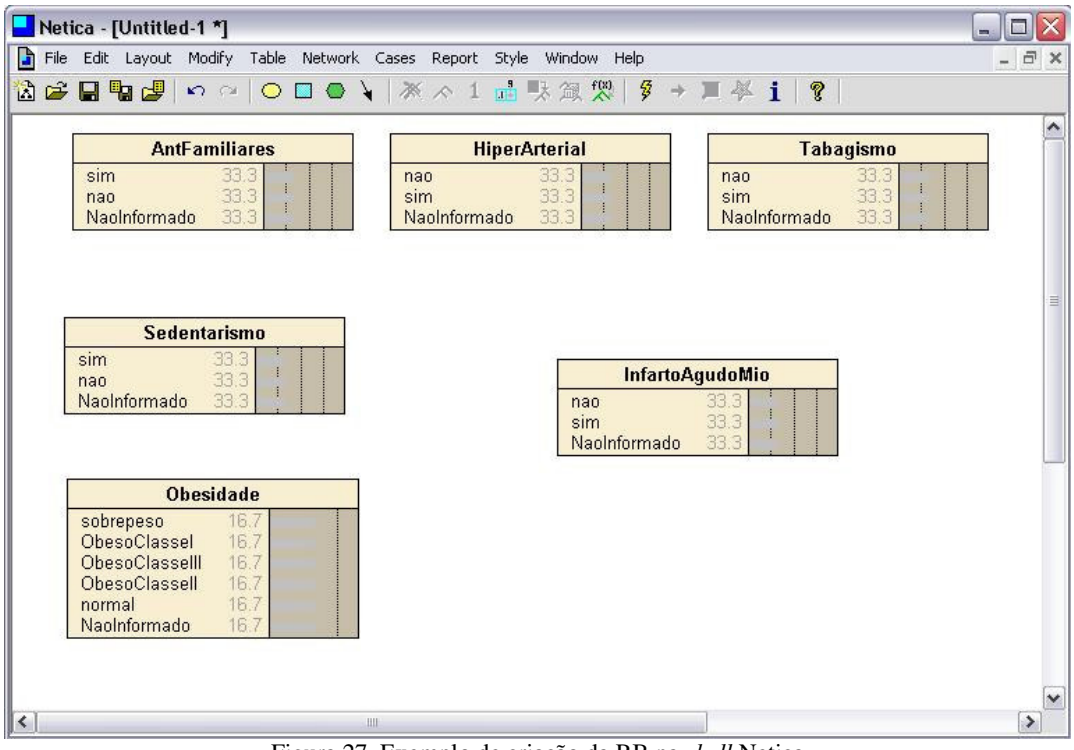


Figura 27. Exemplo de criação da RB na *shell* Netica

6.6.3 Criação dos links e definição do nó raiz

Para a criação das ligações entre os nós é necessário que o usuário especifique qual o nó raiz da RB, em virtude da limitação da API, que não oferece algoritmos de aprendizado da estrutura. O nó raiz é aquele que corresponde ao nó diagnóstico da RB, ao qual todos os outros são ligados. Na aplicação em Java as ligações são criadas conforme ilustra a Figura 28.

```
1       Node noRaiz = nos.get(nomeNoRaiz);
2
3       List<Node> nosFilhos = new LinkedList(nos.values());
4       nosFilhos.remove(nos.get(nomeNoRaiz));
5
6       for (int j = 0; j < nosFilhos.size(); j++) {
7               Node noFilho = nosFilhos.get(j);
8               noRaiz.addLink(noFilho);
9       }
```

Figura 28. Código fonte da criação das ligações entre os nós

A linha 1 recebe o nome do nó raiz, “InfartoAgudoMio”, que é informado pelo usuário na interface principal. Em seguida, na linha 3, a aplicação cria uma lista com todos os nós da rede, e exclui o nó que foi definido como nó raiz (linha 4). Nas linhas 7 e 8 então, os nós da lista são lidos um a um tendo sua respectiva ligação adicionada ao nó raiz. Por exemplo, é realizado uma ligação do nó “Obesidade” (evidência) para o nó “InfartoAgudoMio” (raiz).

Na *shell* Netica esse processo também é realizado de forma manual, conforme ilustra a Figura 29.

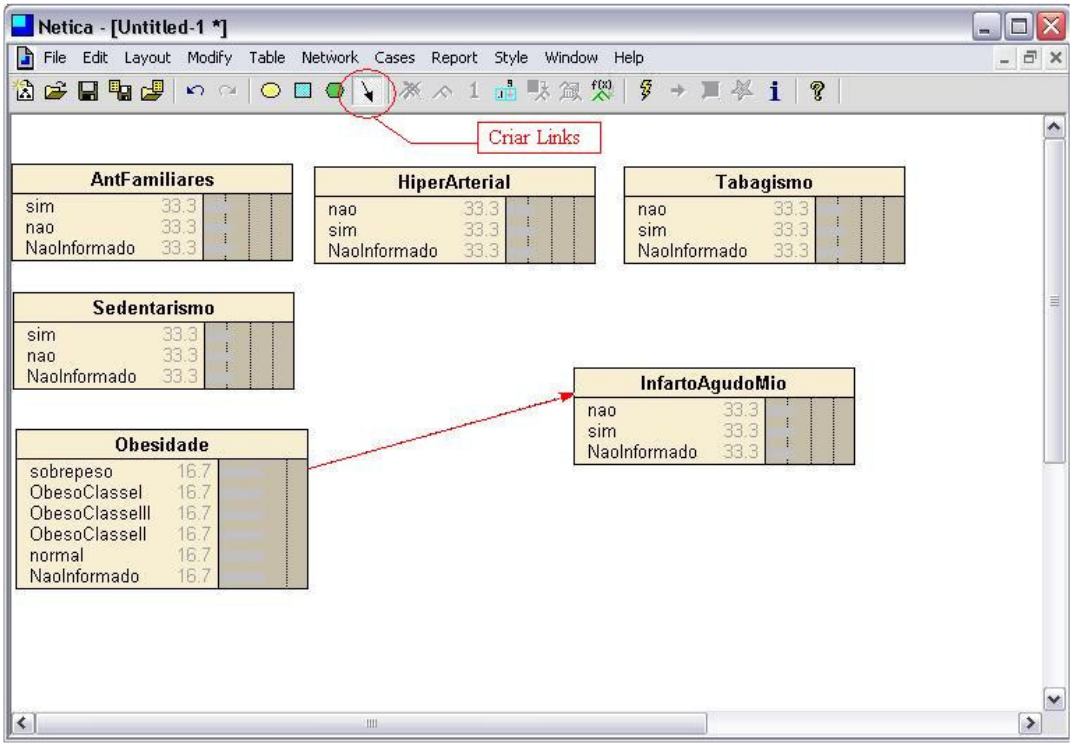


Figura 29. Criação da ligação entre os nós

6.6.4 Escolha do algoritmo, aprendizagem e gravação da RB

O último passo para finalizar a aprendizagem refere-se a escolha do algoritmo. Como já abordado no capítulo 4, a *shell* Netica oferece três diferentes opções (*Counting Learning*, EM e *Gradient Descent*).

Na aplicação em Java a aprendizagem é realizada conforme ilustra Figura 30.

```
1       Caseset cases = new Caseset();
2       Streamer caseFile = new Streamer(arquivo);
3       cases.addCases( caseFile, 1.0, null);
4       Learner learner = new Learner(algoritmo);
5       learner.setMaxIterations(200);
6       NodeList nodes    = net.getNodes();
7       learner.learnCPTs( nodes, cases, 1.0);
8       net.write(new Streamer("C:\\InfartAgudoMio.dne"));
```

Figura 30. Código fonte da aprendizagem na aplicação

O primeiro passo é a criação de um arquivo de casos (linha 1) que será relacionado ao arquivo da base de casos “Base InfartAgudoMio.cas” (linha 2). O

arquivo então é lido utilizando a função *addCases* (linha 3) conforme já foi apresentado pelo Capítulo 4. O usuário então, escolhe qual o algoritmo que deseja usar no processo de aprendizagem, que é passado como argumento do construtor *Learner* (linha 4).

A estrutura da RB, anteriormente definida é lida (linha 6) e a aprendizagem é realizada, criando as PC para cada nó (linha 7). Por fim, a RB é salva na extensão .dne, formato padrão da *shell* Netica, como “InfartAgudoMio.dne”.

Na *shell* Netica, a escolha desses algoritmos é realizada por meio do menu "*Cases"* (Figura 31) na qual a opção "*Incorp Case File"* representa a aprendizagem pelo algoritmo *Counting*, a opção "*Learn Using EM"* indica o algoritmo EM e a opção "*Learn Using Gardient"* o algoritmo *Gradient Descent*.

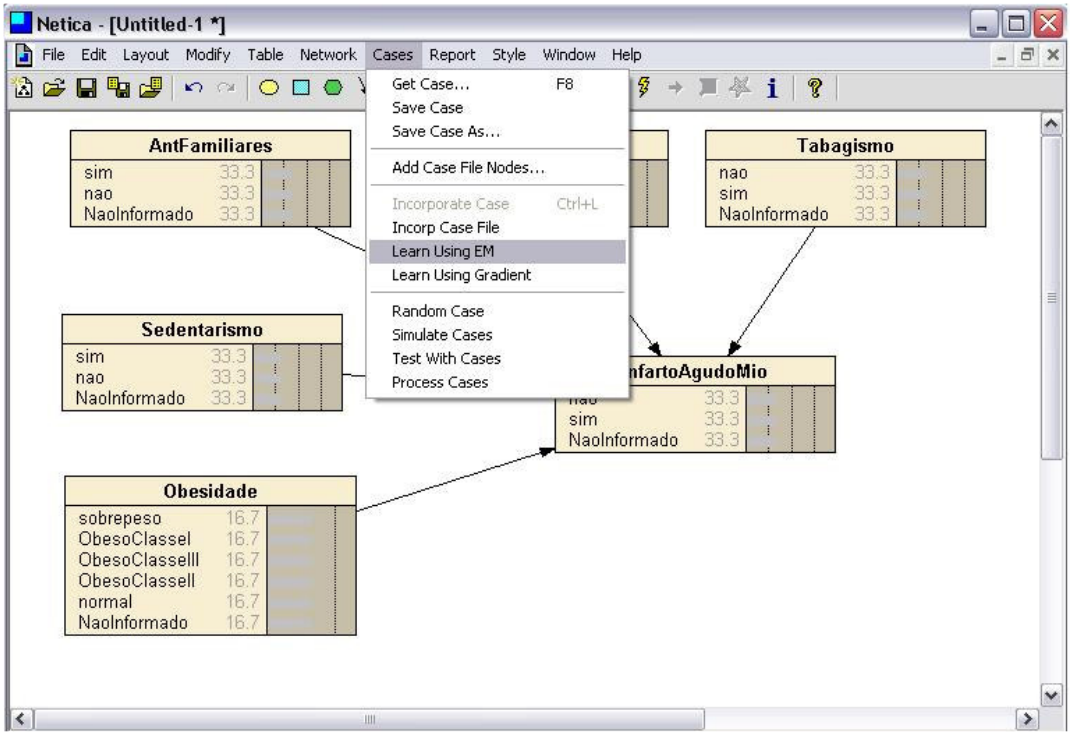


Figura 31. Aprendizagem na *shell* Netica

Finalizada a fase de desenvolvimento da aplicação, passou-se à análise dos resultados obtidos e testes.

6.7 RESULTADOS OBTIDOS E TESTES

Essa pesquisa, que buscou utilizar os algoritmos de aprendizagem bayesiana disponibilizados pela Netica Java API, resultou em uma aplicação desenvolvida em Java a partir do ambiente NetBeans IDE 6.0, que permite, a partir de uma base de casos definida pelo usuário, realizar a aprendizagem bayesiana.

Como visto anteriormente, a base escolhida foi modificada para que houvesse apenas um nó raiz.

Ao inicializar o sistema (Figura 32) o usuário deve selecionar a base de casos que deseja utilizar na aprendizagem, para que então o sistema possa carregar e criar os nós da RB (Figuras 32 e 33).

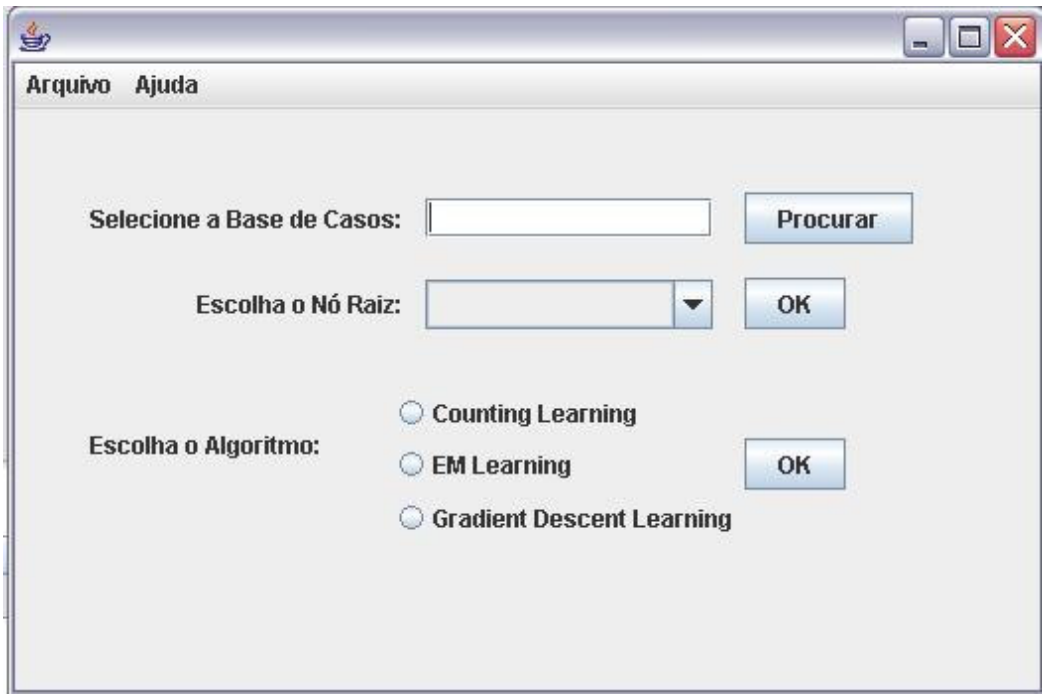


Figura 32. Tela Principal

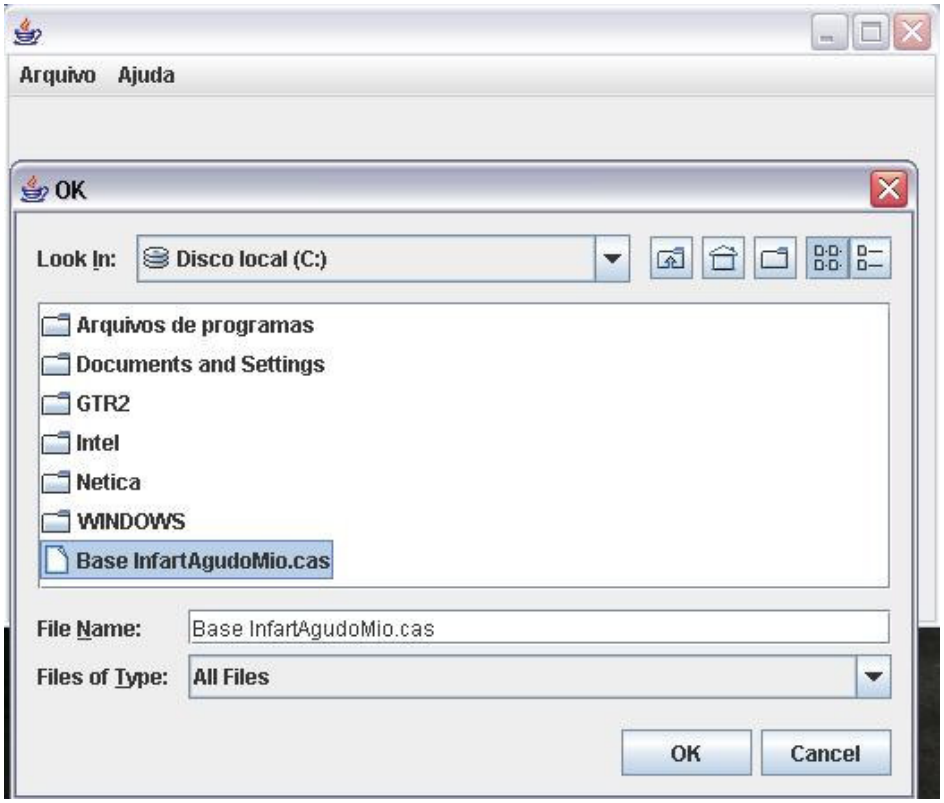


Figura 33. Seleção da base de casos

Com a base de casos definida (Base InfartAgudoMio.cas), a aplicação carrega os nós e cria uma lista com todos, para que o usuário escolha o nó raiz da RB (Figura 34).

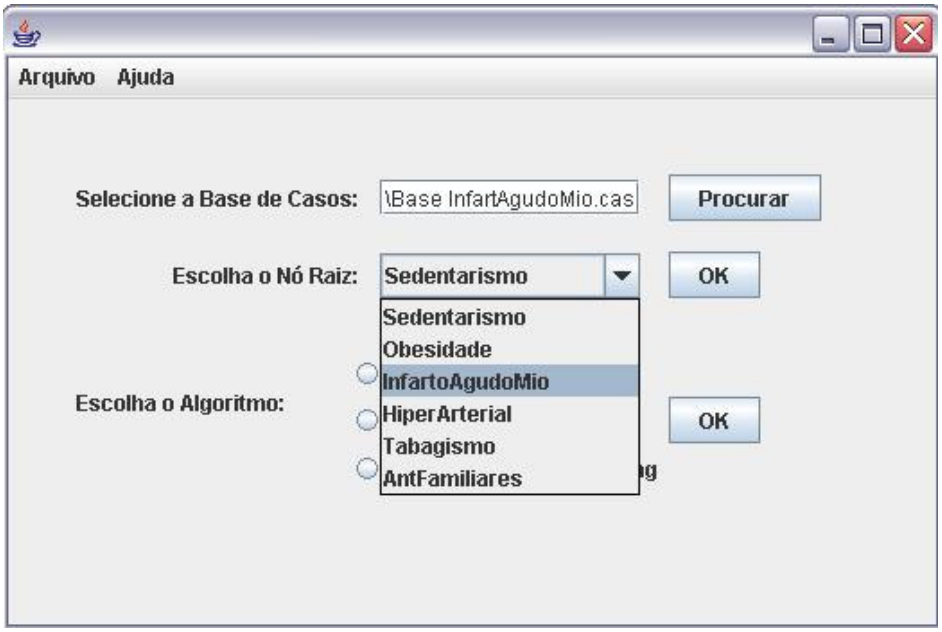


Figura 34. Escolha do nó raiz na ferramenta

Conforme ilustra a Figura 34, o nó InfartoAgudoMio é selecionado como nó raiz da rede e, dessa forma, é criada a ligação dos outros nós para este. Com essa escolha, a RB é formada.

Então, o último passo para a criação da TPC é a escolha do algoritmo que fará a aprendizagem, sendo que o usuário deve selecionar o campo relacionado a opção desejada (Figura 35).

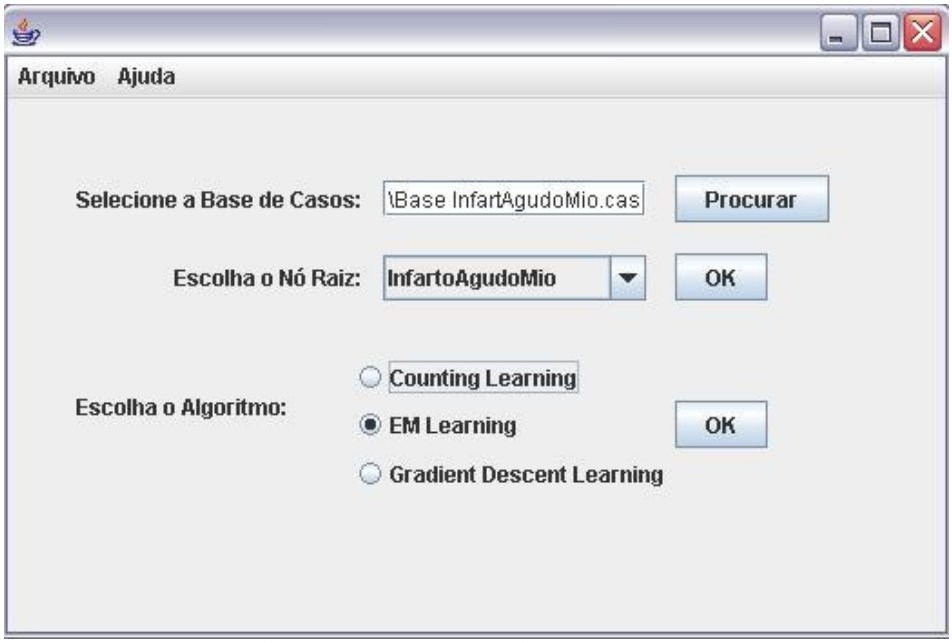


Figura 35. Escolha do algoritmo utilizado na aprendizagem

No exemplo ilustrado na Figura 35, o algoritmo EM é escolhido para realizar a aprendizagem, e com base em sua lógica, a RB é criada e salva no formato do Netica (.dne) (Figura 36).

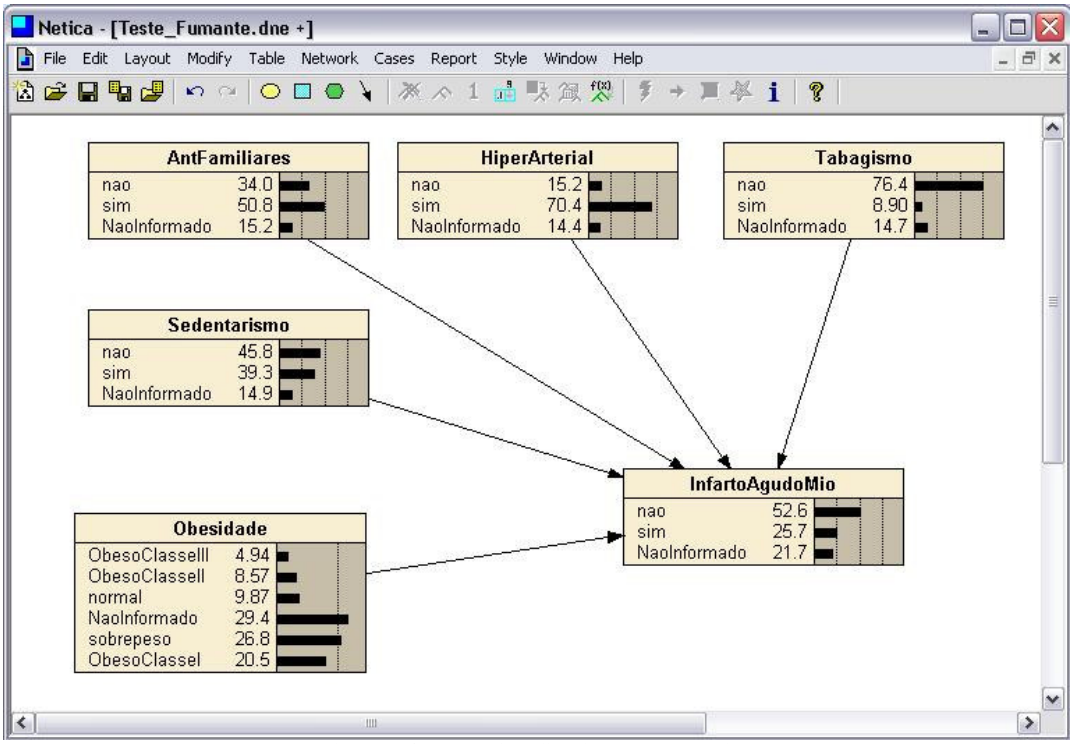


Figura 36. Rede visualizada na *shell* Netica

Com a rede criada, a ferramenta possibilita ao usuário a realização de inferências. Para isso, em uma segunda tela, foram criados dinamicamente os componentes visuais para que o usuário escolha as evidências para cada nó, propagando as PC para o nó raiz, que podem ser visualizadas conforme ilustra a Figura 37.

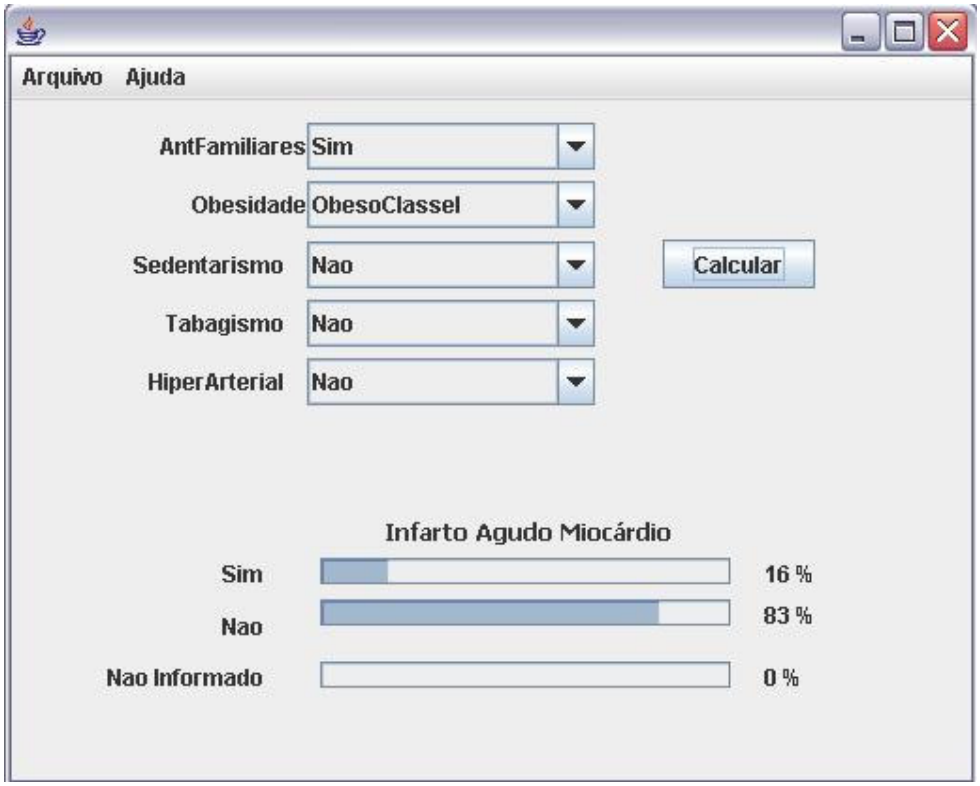


Figura 37. Exemplo de consultas na RB

Os testes foram realizados e comparados com os obtidos pela aprendizagem realizada na ferramenta BNPC desenvolvida por Manarin (2004), e pode-se considerar os resultados semelhantes em ambas as ferramentas, e adequados também conforme opinião da médica especialista que atuou na pesquisa de Manarin (2004).

CONCLUSÃO

Conforme apresentado neste trabalho a aprendizagem bayesiana vem se tornando um tema cada vez mais discutido e explorado em eventos científicos, bibliografias de base, fóruns de discussão, entre outros, em virtude do potencial existente de facilitar o processo de aquisição de conhecimento em SEP que buscam realizar o tratamento de incerteza por aleatoriedade.

Nesse contexto, o objetivo principal dessa pesquisa foi alcançado, sendo construída uma aplicação que possibilita a aprendizagem bayesiana a partir de uma base de casos, utilizando para isso a integração com da Netica Java API em sua versão para o ambiente de programação Java a partir da ferramenta Netbeans IDE 6.0..

Essa pesquisa apresentou então, a utilização dos algoritmos de aprendizagem bayesiana disponibilizados pela Netica Java API (*counting learning*, EM e *gradient descent*) no processo da construção da base de conhecimento de SEP.

Assim, compreendeu-se o processo de construção dos sistemas especialistas probabilísticos e todos os processos envolvidos, etapa fundamental para o desenvolvimento da aplicação proposta.

O aplicativo resultante oferece além das opções de aprendizagem, a possibilidade de inferências e visualização das consultas podendo ser utilizado inclusive por especialistas de domínios de aplicação em suas atividades práticas diárias.

A fim de facilitar futuras pesquisas nessa área, foi disponibilizada a documentação sobre os algoritmos e métodos inerentes a utilização da Netica Java API na aprendizagem bayesiana.

Então, a partir das limitações apresentadas sugere-se como trabalhos futuros implementar no aplicativo desenvolvido alguns algoritmos de aprendizagem de estrutura, a fim de resolver a questão da criação dos *links*.

Recomenda-se ainda, o estudo detalhado dos algoritmos de aprendizagem bayesiana apresentados na Netica Java API (*Expectation Maximization*, *Counting Learning* e *Gradient Descent Learning)*, abrangendo a análise de complexidade e eficiência de cada um.

Pode-se ainda utilizar o aplicativo desenvolvido na resolução de problemas que envolvam o tratamento da incerteza por aleatoriedade.

REFERÊNCIAS

BADIRU, Adedeji B.; CHEUNG, John Y. **Fuzzy** engineering expert systems with neural network applications. New York: J. Wiley, 2002.

BAIRD, Leemon; MOORE, Andrew. **Gradient Descent for General Reinforcement Learning**. Disponível em: <http://www.leemon.com/papers/nips98/graddesc.pdf>. Acesso em: 22 mai. 2008.

BARONE, Dante A. C. **Sociedades artificiais:** a nova fronteira da inteligência nas máquinas. Porto Alegre: Bookman, 2003.

BARRETO, Jorge Muniz, **Inteligência Artificial no Limiar do Século XXI**. Florianópolis: J.M. Barreto PPP Edições

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro: Campus, c2003. 286 p.

BILESIMO, Vanessa Felisberto. **Método para Integração de Uma Base de Conhecimento de Um Sistema Especialista Probabilístico Utilizando a Netica Java API**. 2007. 139 f. Monografia (Bacharel em Ciência da Computação) – Universidade do Extremo Sul Catarinense, Criciúma, 2007.

BITTENCOURT, Guilherme. **Inteligência artificial:** ferramentas e teorias. Florianópolis: UFSC, 1998.

BORMAN, Sean. **The Expectation Maximization Algorithm**: A Short Tutorial. Disponível em: <http://www.seanborman.com/publications/EM_algorithm.pdf>. Acesso em: 22 maio 2008.

BURGES, Chris et al. **Learning to Rank Using Gradient Descent**. Disponível em: <http://research.microsoft.com/~cburges/papers/ICML_ranking.pdf>. Acesso em: 22 maio 2008.

COWEL, R. G.; DAWID, A. P.; et al. **Probabilistic Networks and Expert Systems**. New York: Springer, 1999.

GUINZANI, Jhonas Bonfante. **Mineração De Dados Em Redes Bayesianas Utilizando A API da *Shell Belief Network Power Constructor* (BNPC)**. 2006. 84 f. Monografia (Bacharel em Ciência da Computação) – Universidade do Extremo Sul Catarinense, Criciúma, 2006.

HRUSCHKA JR, E. R. **Imputação Bayesiana no Contexto da Mineração de Dados**. Disponível em: <http://www.coc.ufrj.br/teses/doutorado/inter/2003/teses/HRUSCHKA%20JUNIOR_ER_03_t_D_int.pdf>. Acesso em: 18 nov. 2007

HUGIN EXPERT. Disponível em: http://www.hugin.com/. Acesso: 07 dez. 2007.

KOEHLER, C.; VICARI, R. M.; FLORES, C. D.; et al. **Mineração de Redes Bayesianas a partir da base de dados médicos:** proposta de algoritmo. Disponível em: <www.sbis.org.br/cbis9/arquivos/7.pdf> Acesso em: 7 nov. 2007

KORB, Kevin B.; NICHOLSON, Ann E. **Bayesian artificial intelligence**. Florida: Chapman & Hall/CRC, 2004.

JENSEN, Finn V. **Bayesian Networks and Decisions Graphs**. New York: Springer. 2001

LADEIRA, Marcelo; VICARI, Rosa Maria; COELHO, Helder. **Redes Bayesianas Multiagentes**. In. ENCONTRO NACIONAL DE INTELIGENCIA ARTIFICIAL – ENIA 99. 1999. Disponível em: <http://www.sbc.org.br/reic/edicoes/2002e1/tutoriais/RedesBayesianasMultiagentes.pdf>. Acesso em: 14 out. 2007

LUGER, George F. **Inteligência artificial** : estruturas e estratégias para a resolução de problemas complexos. 4. ed. Porto Alegre, 2002

LUNA, José E. O. **Algoritmos EM Para Aprendizagem de Redes Bayesianas a Partir de Dados Incompletos**. Disponível em: <www.dct.ufms.br/mestrado/dissertacoes/jose_eduardo.pdf> Acesso em: 11 out. 2007

MACHADO, Marcos Paulo. **DinBayes – Componentes Dinâmicos em Sistemas Especialistas Probabilísticos**. 2006. 113 f. Monografia (Bacharel em Ciência da Computação) – Universidade do Extremo Sul Catarinense, Criciúma, 2006.

MANARIN, Daiane de Nez. **Aquisição De Conhecimento Em Sistemas Especialistas Probabilísticos Por Meio Da Descoberta Do Conhecimento Em Base De Dados Para Construção De Redes Bayesianas**. 2004. 112 f. Monografia (Bacharel em Ciência da Computação) – Universidade do Extremo Sul Catarinense, Criciúma, 2004.

MARQUES, Roberto L. DUTRA, Inês. **Redes Bayesianas:** o que são, para que servem, algoritmos e exemplos de aplicações. Disponível em: <www.cos.ufrj.br/~ines/courses/cos740/leila/cos740/Bayesianas.pdf> Acesso em: 07 out. 2007

MATSUURA, Jackson Paul. **Discretização para Aprendizagem Bayesiana:** Aplicação no Auxílio à Validação de Dados em Proteção ao Vôo. Disponível em: <http://www.ele.ita.br/~jackson/files/msc.pdf>. Acesso em: 22 mai. 2008.

MATTOS, Nádia P. **Sistema de Apoio à Decisão para Planejamento em Saúde**. Disponível em: < http://www.ppgia.pucpr.br/ensino/defesas/Nadia_Mattos_2002.PDF>. Acesso em: 21 out. 2007

NASSAR, Sílvia Modesto. **Tratamento de Incerteza:** Sistemas Especialistas Probabilísticos. 2005. Disponível em: <http://www.inf.ufsc.br/~silvia/disciplinas/sep/MaterialDidatico.pdf>. Acesso em: 10 set. 2007.

NORSYS. **Netica-J Manual.** Disponível em: <www.norsys.com>. Acesso em: 15 fev. 2007.

PASSOS, Emmanuel; GOLDSCHIMIDT, Ronaldo. **Data Mining:** um guia prático. Rio de Janeiro: Elsevier, 2005.

PEARL, Judea. **Probabilistic Reasoning in Intelligent Systems:** networks of plausible inference. California, San Mateo: Morgan Kaufmann Publishers, 1988.

RAMOS, Breno Y. K.; AZEVEDO, Fábio R. A.; LIMA, Maurício V. S. **Técnica de Tratamento de Incerteza:** Uma Abordagem Bayesiana para Sistema Especialista Probabilístico de Diagnóstico de LER - DIAGLER. Disponível em: <http://www.cci.unama.br/margalho/portaltcc/tcc2003/d2616.pdf>. Acesso em: 04 nov. 2007

REZENDE, Solange Oliveira**. Sistemas Inteligentes:** fundamentos e aplicações. São Paulo: Manole, 2003.

RUSSELL, Stuart; NORVIG, Peter. **Inteligência Artificial**. Tradução da 2ª edição. Rio de Janeiro: Campus, 2004.

STEIN, Carlos Efrain. **Sistema Especialista Probabilístico:** base de conhecimento dinâmica. 2000. 91f. Dissertação (Pós-Graduação em Ciência da Computação) – Centro Tecnológico, Universidade Federal de Santa Catarina, Florianópolis.

WILLIAMSON, Jon. **Bayesian nets and causality:** philosophical and computational foundations. New York: Oxford University Press, 2005. 239 p.