Gislaine Maria Nalepa

Detecção de *Drifts* em um Processo de Negociação Bilateral Utilizando Rede Bayesiana e IB3

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Curitiba
2010

Gislaine Maria Nalepa

Detecção de *Drifts* em um Processo de Negociação Bilateral Utilizando Rede Bayesiana e IB3

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Área de Concentração: Agentes de Software

Orientador: Prof. Dr. Edson Emílio Scalabrin

Curitiba
2010

Nalepa, Gislaine Maria
Detecção de *Drifts* em um Processo de Negociação Bilateral Utilizando Rede Bayesiana e IB3. Curitiba, 2010.

Dissertação - Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática.

1. detecção de *drift* 2. rede Bayesiana 3. IB3 I.Pontifícia Universidade Católica do Paraná. Centro de Ciências Exatas e Tecnologia. Programa de Pós-Graduação em Informática II - t

A Deus, aos meus pais Miguel e Emilia,
minha irmã Josiele e
meu noivo Ricardo.

Agradecimentos

Agradeço a Deus por ter me guiado em todos os momentos. À Pontifícia Universidade Católica do Paraná, aos professores do Programa de Pós-Graduação em Informática (PPGIa), em especial ao professor Dr. Edson Emílio Scalabrin, pelas sugestões e apoio constante, aos professores Dr. Bráulio Coelho Ávila e Dr. Fabrício Enembreck por terem me auxiliado neste período.

Aos meus pais Miguel e Emilia, minha irmã Josiele e meu noivo Ricardo pela paciência, compreensão, incentivo e apoio.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Em um processo de negociação automatizada, é comum que uma das partes não revele suas preferências ao seu oponente. Na tentativa de ajustar tal processo, cada agente de negociação pode ser dotado de capacidade de aprendizagem e de detecção de alteração das preferências do seu oponente. Este trabalho apresenta em detalhes técnicas de detecção de *drift* que se mostram úteis neste cenário. Os métodos de conjunto, como DWM e AddExp, permitem que as competências dos especialistas se complementem e que o resultado seja definido por um sistema de votação baseado em pesos, permitindo o alcance de resultados mais satisfatórios do que quando um indivíduo isolado é utilizado. Neste estudo, a capacidade de detecção de *drift* do DWM é testada com algoritmos de aprendizagem de contextos diferentes; trata-se do IB3 (baseado em instâncias) e da rede Bayesiana (probabilístico); sendo assim, os experimentos realizados centram-se em torno das configurações DWM-IB3 e DWM-Rede Bayesiana. O desempenho de cada sistema foi avaliado para mudanças de conceito gradual, moderada e abrupta e os resultados obtidos comprovaram que ambas as configurações são capazes de detectar alterações nas preferências do oponente de forma eficiente.

Palavras-chave: negociação, IB3, rede Bayesiana, DWM, detecção de *drift*.

Abstract

In an automated negotiation process, it is usual for one party not to reveal their preferences to its opponent. In an attempt to adjust itself to this process, each trade agent can be endowed with capabilities of learning and detecting its opponent‘s changing preferences. This paper presents techniques for drift detection that are useful in this scenario. Ensemble methods, like DWM and AddExp, allow the specialists‘ skills to complement each other and prediction is defined by a voting system based on weights, allowing to reach better results than when a single individual is used. In this paper, the capacity of drit detection of DWM is tested with algorithms from different contexts: IB3 (instance-based) and bayesian network (probabilistic). The experiments performed revolve around the settings DWM-IB3 and DWM-Bayesian Network. The performance of each system was evaluated for gradual, moderate and abrubt drifts in concept, and the results showed that both settings are able to efficiently detect drifts in the preferences of the opponent.

Keywords: negotiation; IB3; bayesian network; DWM; drift detection.

Capítulo 1

Introdução

Um agente de software é uma unidade computacional autônoma que pode agir no lugar de um ser humano em ambiente aberto e evolutivo (e.g. Internet e WWW), empregando técnicas de inteligência artificial para executar certas tarefas relativas à manipulação de informações, reconhecer o ambiente em que está inserido, comunicar-se com outros agentes, interagir com o usuário, criar um perfil do usuário a partir de um modelo de suas atividades e comportar-se de forma tal que seus objetivos sejam alcançados, como por exemplo, na resolução de conflitos por meio de um processo de negociação. Tal negociação pode ser definida como uma forma de tomada de decisão envolvendo dois ou mais agentes que, por não poderem tomar decisões de modo independente, são levados a fazerem concessões para obterem um acordo (Hart, 1968). A automatização de tal processo de decisão é uma forma de instigar melhorias ao processo de negociação e pode trazer benefícios significativos, como a redução dos ciclos, dos custos das transações e dos erros causados por perdas de informação; por exemplo, em negócios realizados em um ambiente virtual do tipo B2B. Tal sofisticação, no modo de realizar transações comerciais, tornou-se possível graças aos avanços nas tecnologias de redes abertas e da inteligência artificial.

A negociação realizada por agentes autônomos em um ambiente aberto traz diversos desafios no nível de aplicação, que são:

- a tomada decisão no tocante a escolha do parceiro de transação deve ser realizada por meio de um conjunto de critérios baseado na reputação do parceiro [Botelho et al 2010];
- o número de trocas de ofertas e contraofertas em uma negociação deve ser limitado sem negligenciar a possibilidade de um resultado favorável a ambos os lados (estilo *win/win*) [Romanhuki et al 2008];
- a presente alternância de interesse local por parte de cada negociador, expressa na

forma de uma mudança local de conceito, precisa ser detectada de modo eficiente para elevar as chances de sucesso da interação [Enembreck et al 2007].

A alternância de interesse local por parte de cada negociador pode ser tratada eficientemente por meio de técnicas de *Drift Detection*, à medida que ela detecta rapidamente a alteração de um conceito. A rápida detecção é relevante, de um lado, para reduzir o efeito da propagação de equívocos de alterações de conceitos e, de outro lado, para elevar o efeito da propagação de reais alterações de conceitos. A dificuldade é a distinção de entre uma falsa e uma real mudança de conceitos.

1.1 Objetivo

O objetivo deste trabalho é apresentar um sistema de aprendizagem baseado em agentes para a detecção de *drift* em negociação bilateral. São consideradas duas formas de aprendizagem: baseada em instância (IB3) e em rede Bayesiana (RB). A coordenação e a integração das decisões dos agentes baseiam-se na estratégia *Dynamic Weighted Majority (DWM)*, pois espera-se verificar a eficiência deste método de conjunto com a utilização de diferentes técnicas de aprendizagem. As configurações avaliadas são DWM-IB3 e DWM-RB.

O desempenho de cada configuração deve ser avaliado em situações de variações de conceitos, simulando mudanças nas preferências de uma das partes negociadoras. Para testar a capacidade de adaptação automática a essas mudanças, os conceitos serão alterados de forma gradual, moderada e abrupta.

Como resultado, espera-se que o DWM consiga manter um desempenho satisfatório para ambas as configurações e que elas sejam capazes de detectar alterações nas preferências do oponente de forma eficiente.

1.2 Motivação

A globalização e a necessidade de fortalecer relacionamentos mostram o atual contexto da negociação com pessoas interagindo, cada vez mais, apoiadas em recursos tecnológicos e não face a face como em épocas anteriores. Os negócios estão se tornando ilimitados, exigindo que organizações e pessoas adotem diferentes estratégias para melhorar o relacionamento e aumentar a eficiência de seus processos de negociação. A negociação é um processo em que tanto pessoas como sistemas podem interagir. Diversos serviços conhecidos como *e-commerce* ou comércio eletrônico estão disponíveis na internet, entre

eles lojas virtuais e leilões eletrônicos, onde as pessoas podem comprar ou vender produtos por um preço fixo. O comércio eletrônico atual requer uma nova estrutura de negócios e a negociação potencialmente faz parte deste contexto.

A utilização de agentes de software para automatizar processos de negociação torna a arquitetura simples, pois as atividades são distribuídas e o problema decomposto, influenciando positivamente na tratabilidade do problema. A existência de diversos métodos de aprendizagem, torna os agentes capazes de identificar fatores importantes para a negociação, tais como as preferências do oponente. A utilização de um método de conjunto possibilita uma baixa complexidade para a coordenação e cooperação entre os agentes.

A escolha do método DWM em meio a outras técnicas de conjunto, é devida a este ser um método *online*, que atribui peso aos especialistas e realiza predições por meio do voto da maioria. Optou-se também por utilizar técnicas de aprendizagem distintas, uma baseada em instâncias e outra probabilística, pois acredita-se que as características e as capacidades de coordenação do DWM possibilitam o alcance de um desempenho satisfatório na detecção de *drifts* para ambas as configurações, contribuindo assim, com o desenvolvimento e apresentação de resultados de mais um estudo nesta área em desenvolvimento ascendente, a negociação automatizada.

1.3 Organização

Cinco capítulos compõe o presente trabalho. O primeiro capítulo introduz e contextualiza o estudo desenvolvido. O capítulo 2 apresenta fundamentos de um processo de negociação automatizada. O capítulo 3 apresenta alguns métodos de conjunto que são capazes de trabalhar com *drift detection*, técnicas de aprendizagem e também alguns trabalhos previamente realizados sobre o tema. O capítulo 4 detalha a metodologia, a arquitetura do ambiente de negociação, os cenários de testes e os resultados obtidos. O capítulo 5 expõem as considerações finais e trabalhos futuros.

Capítulo 2

Negociação

A negociação é um processo de tomada de decisão que acontece entre dois ou mais agentes, humanos ou virtuais, com objetivos contraditórios, onde eles buscam resolver as diferenças na tentativa de estabelecer um acordo (Pruitt, 1981)(Raiffa, 1982). O termo agente será utilizado para denominar agente virtual ou de software. Um processo de negociação é aplicável na resolução de situações de conflitos por meio de diversas interações entre os agentes e tem como objetivo encontrar um estado que estabiliza a satisfação das partes envolvidas. A quantidade de partes envolvidas permite que as negociações sejam classificadas como bilateral ou multilateral.

2.1 Negociação Bilateral

A negociação bilateral envolve dois agentes (Figura 2.1), um comprador e um vendedor, cujos desejos são conflitantes. Eles estabelecem um processo de negociação, por meio de trocas sucessivas de ofertas e contraofertas, na tentativa de atingirem um acordo comum [Kersten et al 1991].

Raiffa (1982) considera que o processo de negociação deve ser colaborativo para que o mesmo seja factível, onde cada parte deve raciocinar, mesmo que inconscientemente, sobre funções de utilidade. Tais funções requerem que a formalização do conhecimento para tomada de decisão seja anterior à delegação da negociação. Isto significa que para ofertar um produto, o vendedor deve ter conhecimentos iniciais sobre os produtos disponíveis e sobre o perfil inicial do comprador.

Assim, cada agente possui uma lista de preferências que define o domínio de uma negociação, incluindo o limite inicial e o limite final de cada atributo. De acordo com tal lista, um agente comprador envia uma proposta para um agente vendedor com valores próximos aos limites iniciais. O agente vendedor pode aceitar a proposta ou devolver

uma contraproposta ao agente comprador. Este processo repete até que ambos os agentes estabeleçam um comum acordo ou até atingir um número de iterações predefinidas. O agente comprador e o agente vendedor relaxam os valores das ofertas e contraofertas na direção do seu próprio limite final, conforme ilustra a Figura 2.2.

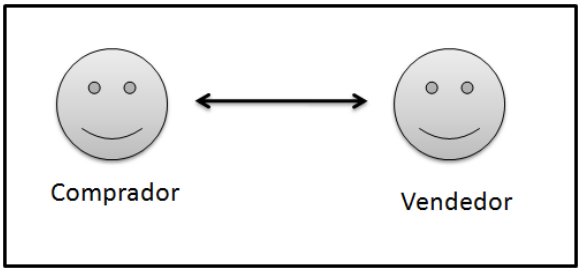

Figura 2.1: Ilustração de negociação do tipo 1:1 (bilateral).

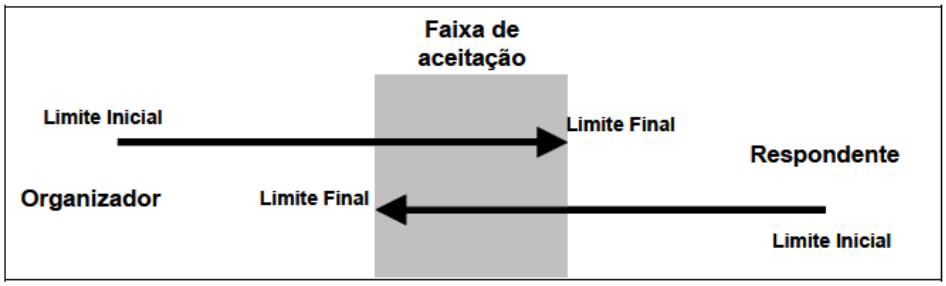

Figura 2.2: Processo de negociação bilateral.

O acordo pode ser concluído se os valores propostos por ambos os agentes estiverem dentro da faixa de aceitação. Uma negociação pode ser encerrada sem que haja acordo, caso a data pré-estabelecida para conclusão da negociação extrapole e valores propostos por ambos os agentes não estejam dentro da faixa de aceitação.

Seja $i(i \in \{a, b\})$ os agentes participantes de uma negociação, $j(j \in \{1, ..., n\})$ os atributos de cada agente, $x_j \in [min_j, max_j]$ o valor do atributo j onde $[min_j, max_j]$ define o domínio do atributo j em termos de valor inicial e final do mesmo. Cada agente i tem um valor de utilidade para cada atributo j, V_j^i: $[min_j, max_j]$, que são definidos no intervalo de $[0, 1]$. Outro item relevante é o peso de cada atributo, onde w_j^i determina a importância do atributo j para o agente i. Por convenção, o somatório dos pesos de cada atributo é igual a 1 para todo i em $\{a, b\}$.

$$V^i(x) = \sum_{1 \leq j \leq n} w_j^i V_j^i(x_j) \tag{2.1}$$

Quando o agente comprador a recebe uma oferta do agente vendedor b no tempo t, a importância da oferta $x_b^t \rightarrow a$ é determinada por meio do cálculo da utilidade.

$$I^a(t', x^t_{b\to a}) = \begin{cases} aceitar, se\, t' < t^a_{max} \\ rejeitar, se\, V^a(x^t_{b\to a}) \geq V^a(x^{t'}_{a\to b}) \\ x^{t'}_{a\to b}\, caso\, contrário \end{cases} \quad (2.2)$$

Onde:

- $V^a(x^t_{b\to a})$ é a avaliação que o agente a faz da oferta recebida do agente b no instante de tempo t;
- $V^a(x^{t'}_{b\to a})$ é a próxima oferta a ser proposta pelo agente;
- t^a_{max} é o tempo máximo da negociação;
- t' é o tempo atual da negociação.

Se o valor da utilidade da oferta de $b \to a$ for maior que a contraoferta do agente a preparada para o agente b, então o agente a aceita a oferta e encerra a negociação. Se a data pré-estabelecida para conclusão da negociação não extrapolou e a negociação ainda não foi estabelecida, então uma nova oferta é submetida [Raiffa 82].

2.2 Negociação Multilateral

As negociações multilaterais são tipicamente processos de leilão e caracterizam-se pelo envolvimento de mais de dois agentes. Uma negociação multilateral pode ser do tipo 1:N, onde um único vendedor recebe oferta de vários compradores (Figura 2.3), ou do tipo N:M, onde vários vendedores e compradores participam do processo (Figura 2.4).

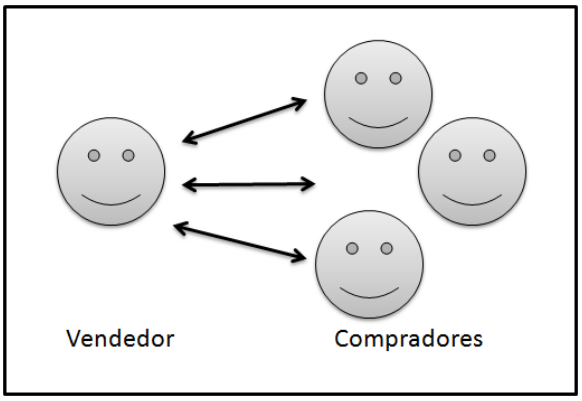

Figura 2.3: Ilustração de negociação do tipo 1:N (multilateral).

A seguir são apresentados os tipos de leilões mais comuns:

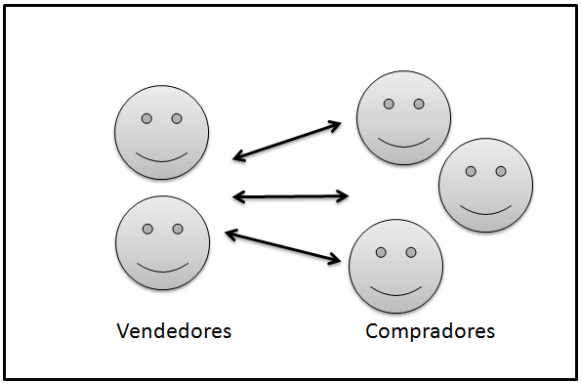

Figura 2.4: Ilustração de negociação do tipo N:M (multilateral).

- Leilão Inglês

  Neste tipo de leilão, o leiloeiro inicia sugerindo um preço pelo bem. Se não houver proposta da parte dos agentes, o bem é alocado ao leiloeiro por esse valor. Em cada lance, os agentes realizam ofertas com um valor maior que a última oferta realizada, até que nenhum agente volte a realizar um lance. O bem é então alocado ao agente que realizou a maior oferta. Este é um tipo de leilão de primeiro preço, lance aberto e ascendente.

- Leilão Holandês

  Neste leilão, o leiloeiro inicia ofertando o bem por um valor alto e então diminui o valor continuamente, até que um agente realize uma oferta com valor igual ao ofertado. O bem é alocado para o agente que realizou a oferta. Este é um tipo de leilão de primeiro preço, lance aberto e descendente.

- Leilão de primeiro preço e lance fechado

  São leilões de apenas uma rodada onde os participantes realizam somente um lance e o bem é alocado ao agente que realizou a maior oferta.

- Leilão Vickrey

  São leilões de uma rodada, lance fechado e de segundo preço. Os participantes realizam somente um lance e o bem é alocado ao agente que realizou a maior oferta. A diferença para o tipo de leilão anterior é que o valor a ser pago pelo ganhador é o valor da segunda maior oferta.

Visando aperfeiçoar um processo de negociação, táticas podem ser empregadas para programar as trocas em uma negociação. A próxima seção apresenta algumas táticas de negociação utilizadas com frequência.

2.3 Táticas de Negociação

As táticas de negociação podem ser expressas como dependentes do tempo, de recurso e de comportamento. Cada tática é definida por um conjunto de funções que determinam como será calculado o valor de cada atributo usado na negociação. Cada função define um comportamento próprio (Faratin et al., 1998). Cada agente implementa sua própria tática. Todavia, um agente pode ter, por exemplo, duas táticas, T1 e T2, e neste caso, cada tática com um peso diferente. Essas táticas são descritas a seguir com mais detalhes.

2.3.1 Tática Dependente do Tempo

Neste tipo de tática, a variação do valor do atributo depende do tempo restante da negociação. A oferta inicial é instanciada como sendo um ponto no domínio dos valores do atributo j da negociação. Para calcular este ponto, uma constante K_j^a é definida e multiplicada pelo tamanho do intervalo para determinar o valor do atributo a ser ofertado. Para calcular o valor de uma oferta, é necessário definir a variação das ofertas em um determinado tempo. Essa variação é calculada por uma função α(t), que pode ser dividida em:

1. Polinomial: tende a relaxar os valores da negociação de forma constante.

$$\alpha_t^a(t) = k_j^a + (1 - k_j^a)\left(\frac{min(t, t_{max}^a)}{t_{max}^a}\right)^{\frac{1}{\beta}} \tag{2.3}$$

2. Exponencial: tende a relaxar os valores de forma mais lenta no inicio da negociação e de forma mais rápida quando o processo se aproxima do fim.

$$\alpha_t^a(t) = e^{\left(1-\frac{min(t,t_{max}^a)}{t_{max}^a}\right)^{\beta} ln\, k_j^a} \tag{2.4}$$

Onde:

- K_j^a: constante que determina o valor do critério j que o agente oferecerá na primeira proposta;
- β : constante que determina o comportamento da negociação;
- t: tempo da negociação.

Tendo o valor α(t) que está entre 0 e 1, é possível calcular o valor de uma oferta utilizando a seguinte fórmula:

$$x^t_{a\to b}[j] = \begin{cases} min^a_j + \alpha^a_j(t)(max^a_j - min^a_j)\ \ V^a_j \ \ decrescente \\ min^a_j + (1 - \alpha^a_j(t))(max^a_j - min^a_j)\ \ V^a_j \ \ crescente \end{cases} \tag{2.5}$$

Onde:

- min^a_j: valor mínimo do critério j para o agente a;
- max^a_j: valor máximo do critério j para o agente a.

Foi descrito acima que a constante β determina o comportamento da negociação. Existem três comportamentos possíveis:

- *Boulware*: neste comportamento, a constante assume um valor menor que 1 ($\beta < 1$). Uma oferta inicial é determinada e mantida até que o tempo da negociação esteja próximo do fim. A partir deste momento, os valores são relaxados até os valores limites das ofertas;
- Linear: neste comportamento, a constante assume o valor de 1 ($\beta = 1$). Neste caso, as ofertas são calculadas linearmente do inicio até o fim da negociação;
- Conceder: neste caso, a constante recebe um valor maior que 1 ($\beta > 1$). Uma oferta inicial é gerada com os valores próximos ao limite final da negociação e a partir daí os valores são relaxados em pequenas proporções.

2.3.2 Tática Dependente do Recurso

Podem ser considerados recursos: os valores trocados entre os agentes, o número de agentes interessados em uma negociação e o próprio tempo. Neste tipo de tática, as ofertas são geradas dependendo de como um recurso está sendo utilizado, sendo que um agente se torna mais conciliador a medida que a quantidade de recursos diminui. Antes de calcular o valor da tática é necessário determinar a quantidade de recursos em um dado tempo t para um determinado agente a. Este tipo de cálculo varia conforme o tipo de recurso utilizado. Quando o recurso é o número de agentes participantes na negociação, utiliza-se a fórmula 2.6. Para o recurso tempo, utiliza-se a fórmula 2.7 e quando o tempo diminui de uma forma linear deve-se utilizar a fórmula 2.8.

$$resource^a(t) = |N^a(t)| \tag{2.6}$$

$$resource^a(t) = \mu^a \frac{|N^a(t_c)|^2}{\sum_i |x^{t_c}_{i\leftrightarrow a}|} \tag{2.7}$$

$$resource^a(t) = min(0, t - t^a max) \tag{2.8}$$

O valor da tática pode ser adquirido pela fórmula 2.9. Mais detalhes sobre esta tática podem ser encontrados em Faratin et al. (1998).

$$\alpha_j^a(t) = k_j^a + (1 - k_j^a)e^{-resource(t)} \tag{2.9}$$

2.3.3 Tática Imitativa

Neste tipo de tática, um agente tende a imitar o comportamento do seu oponente na geração de suas contraofertas. É importante ressaltar que existem diferentes tipos de imitações:

RELATIVE TIT-FOR-TAT: o agente imita o comportamento do oponente a $\delta \geq 1$ passos anteriores. Esta reprodução de comportamento é expressa em percentual.

$$x_{a\to b}^{t_n+1}[j] = min(max(\frac{x_{b\to a}^{t_n-2\delta}[j]}{x_{b\to a}^{t_n-2\delta+2}[j]} x_{a\to b}^{t_n-1}[j], min_j^a), max_j^a) \tag{2.10}$$

Onde:

- $x_{a\to b}^t$: oferta do agente a para o agente b no tempo t;
- min_j^a : valor mínimo do critério j para o agente a;
- max_j^a : valor máximo do critério j para o agente a;
- δ : número de passos anteriores que serão analisados.

RANDOM ABSOLUTE TIT-FOR-TAT : relaxa os valores em termos absolutos, ou seja, se um agente relaxar a oferta em 5 o outro agente também vai relaxar em 5.

$$x_{a\to b}^{t_n+1}[j] = min(max(x_{a\to b}^{t_n-1}[j] + (x_{b\to a}^{t_n-2\delta}[j] - x_{b\to a}^{t_n-2\delta+2}[j]) + (-1)^S R(M), min_j^a), max_j^a) \tag{2.11}$$

Onde:

- $x_{a\to b}^t$: oferta do agente a para o agente b no tempo t;
- min_j^a : valor mínimo do critério j para o agente a;
- max_j^a : valor máximo do critério j para o agente a;

- δ : número de passos anteriores que serão analisados;
- R(M): gera valor randômico no intervalo [0,M].

AVERAGE TIT-FOR-TAT: a média do percentual dos valores das ofertas é calculada pelo agente. A condição para esta tática é: $4n > 2\gamma$.

$$x_{a\to b}^{t_n+1}[j] = min(max(\frac{x_{b\to a}^{t_n-2\gamma}[j]}{x_{b\to a}^{t_n}[j]} x_{a\to b}^{t_n-1}[j], min_j^a), max_j^a) \tag{2.12}$$

Onde:

- $x_{a\to b}^t$: oferta do agente a para o agente b no tempo t;
- min_j^a : valor mínimo do critério j para o agente a;
- max_j^a : valor máximo do critério j para o agente a;
- γ : número de ofertas para calcular a média.

Os agentes podem alterar a importância dos seus critérios com o passar do tempo para se tornarem mais flexíveis. Este processo de mudar os pesos das táticas é conhecido como estratégia e é baseado no histórico de negociações e no modelo de raciocínio de cada agente. Diferentes combinações de táticas podem ser utilizadas na geração de ofertas, uma estratégia determinará qual combinação de táticas deverá ser usado.

Muitos projetos foram desenvolvidos utilizando táticas de negociação. Entre eles, Faratin (1998) desenvolveu um modelo de negociação entre agentes autônomos, definindo táticas e estratégias que os agentes podem utilizar ao gerar uma oferta inicial, avaliar propostas ou gerar contrapropostas. Em seus estudos, ele se concentrou em uma negociação multilateral e multi-dimensão, um ambiente com recursos limitados e em uma classe de negociação chamada negociação orientada a serviço, onde um agente (cliente) requisita a execução de um serviço a outro agente (servidor). Neste modelo os agentes possuem papéis (comprador / vendedor) que estão em conflito e possuem interesses opostos em qualquer uma das dimensões da negociação. Primeiramente, é definido sobre qual grupo de variáveis a negociação irá ser executada e então o processo consiste na troca de ofertas entre os agentes que utilizam táticas e estratégias para relaxarem seus valores. As táticas propostas são as já apresentadas na subsessão (2.0.3 Táticas de Negociação). O agente pode rejeitar ou aceitar uma oferta de acordo com as relações apresentadas na Equação 2.2.

2.4 Considerações Finais

Um negociador conduz o processo com base nos interesses e estratégias definidas; entretanto, é certo que podem ocorrer mudanças significativas nos interesses, estratégias e objetivos do negociador mesmo quando o processo de negociação já está em andamento. A causa da mudança pode estar relacionada a diversos motivos, desde uma súbita alteração na preferência do negociador, até fatores externos que ocasionem essas mudanças de interesse; mudanças estas, que irão alterar o comportamento e a forma de agir do participante. Conhecer os interesses e o padrão de comportamento da outra parte representa a experiência do negociador, que é formada por meio da participação em várias negociações. Possuir esses conhecimentos, pelo menos em partes, pode alterar significativamente o valor de uma oferta e consequentemente os resultados da negociação. Portanto, além de um agente ser equipado com estratégias e táticas para um processo de negociação, ele também pode possuir a capacidade de aprender a partir de negociações passadas e de aplicar esse conhecimento em negociações futuras. O próximo capítulo apresenta a importância de processos de negociação que utilizam técnicas de aprendizagem e como elas podem ser utilizadas para melhorar a capacidade de negociação entre os agentes.

Capítulo 3

Negociação e Aprendizagem

É possível que ambas as partes alcancem satisfação em um processo de negociação, isto pode ocorrer porque cada uma das partes pode dar peso diferente a cada um dos itens sendo negociado. Na maioria das vezes, a dificuldade em encontrar soluções ganha-ganha é devida aos agentes não revelarem suas preferências, funções de utilidade ou preço de reserva. Buscando melhorar a capacidade de negociação, um bom negociador deve avaliar o perfil do seu oponente, descobrir os interesses por trás de suas ações, possuir habilidade para interpretar as mensagens e identificar o que é importante para o estabelecimento de um acordo; para isso, o agente deve aprender valores aproximados baseando-se na troca de ofertas e contraofertas durante a negociação e negociações anteriores. Muitos projetos tem sido desenvolvidos visando a aprendizagem das preferências do oponente e, além disso, a identificação das políticas de negociação do competidor, melhorando a capacidade de negociação entre os agentes.

Menke e Martinez (2009) trabalharam com o conceito de adaptação de problema. Eles propuseram duas variações do método *Learning with Confidence-based Target Relabeling*, adaptando o problema ao aprendiz e fazendo assim, que o algoritmo de aprendizagem aprenda mais facilmente. Wersing et al. (2007) apresentaram uma arquitetura capaz de realizar a aprendizagem *online* de diversos objetos utilizando processamento biológico visual, permitindo um processo de treinamento simples e intuitivo. Muitos outros estudos foram desenvolvidos visando especificamente a aprendizagem das preferências do oponente em um processo de negociação (Faratin et al., 1998), (Coehoorn, 2004) (Romanhuki, 2008); entretanto, identificar e tratar as alterações de conceito é uma atividade essencial em sistemas que envolvem modelos do comportamento humano, por isso, além da capacidade de aprender as preferências do oponente, o agente deve ser capaz de detectar alterações nesses valores dinamicamente.

Um dos eventos mais conhecidos que envolve a negociação em ambientes dinâmicos

é o TAC (*Trade Agent Competition*). Trata-se de uma competição anual entre agentes negociadores, onde múltiplos agentes competem entre si, negociando em leilões bens relativos a uma viagem. Os agentes podem usar estratégias de negociação complexas, táticas, modelação do oponente e capacidade de aprendizagem. As mais variadas metodologias da Inteligência Artificial são utilizadas para capacitar os agentes, entre elas: redes neurais, lógica difusa, algoritmo genético, processos de decisão de Markov, etc. TAC SCM (*Supply Chain Management*) fornece um ambiente de simulação de cadeia de suprimentos no segmento de computadores, permitindo a competição entre agentes, cada um representando uma organização. Neste ambiente, os agentes devem tomar decisões de forma autônoma e a disputa ocorre por meio de um protocolo de leilão fechado e de primeiro preço. Este evento tem contribuído para o desenvolvimento de muitos estudos importantes (Pereira et al., 2010, etc).

A Inteligência Artificial, área de pesquisa que busca métodos ou dispositivos que simulem a capacidade que o ser humano tem de raciocinar e resolver problemas, existe há décadas e é impulsionada pelo rápido avanço da informática e da computação. A sua progressão permite a integração da inteligência humana e da tecnologia em áreas de reconhecimento de padrões, reconhecimento de voz, automação e controle, técnicas de procura e de aprendizagem. As técnicas descritas nesta seção visam aumentar a eficiência de processos que envolvem a tomada de decisão, como os de negociação, por exemplo. Neste tipo de ambiente, a detecção das variações de conceitos é essêncial e será discutida pelos métodos SEA, DWM e AddExp. Os algoritmos de aprendizagem baseados em instância utilizam um grupo específico de instâncias para realizar classificações, enquanto a rede Bayesiana, um método probabilístico, realiza predições levando em consideração o grupo de instâncias como um todo.

3.1 Drift Detection

Drift Detection é uma atividade do aprendizado de máquina que consiste em identificar alterações que ocorrem na descrição dos conceitos com o decorrer do tempo. Um conceito pode ser definido como um conjunto de variáveis e seus respectives atributos que compõem um determinado produto, por exemplo:

Conceito A: AtrasoEntrega = normal e Preço = normal.

Conceito B: AtrasoEntrega = muitobaixo, Preço = baixo e Cor = branco.

Alterações em conceitos (*concept drift*) acontecem frequentemente no mundo real, um dos exemplos mais típicos são alteração nas preferências dos clientes. Suponha a seguinte situação:

Uma concessionária vende seus veículos a um preço normal e com um atraso de entrega também normal (Conceito A). Um certo dia, a empresa recebe uma grande quantidade de carros na cor branca e armazenar todos esses veículos por muito tempo geraria um alto custo. A empresa altera suas condições, vendendo os veículos da cor branca a um preço baixo e com atraso de entrega muito baixo (Conceito B).

Esta situação demonstra uma mudança do conceito de oferta interessante do vendedor. Se o comprador é capaz de detectar essas mudanças, o processo de negociação pode ser mais eficiente e alcançar um acordo mais rapidamente. Em muitas situações, a causa da alteração de um conceito não é conhecida, fazendo a tarefa de *drift detection* complicada, principalmente quando a causa está ligada a um fenômeno irregular. A tarefa de *concept drift*, como no caso exemplificado, pode acontecer das formas descritas a seguir e ilustradas pela Figura 3.1.

- Abrupto: neste tipo de $drift$, um conceito A é repentinamente substituído por um conceito B;
- Gradual: este tipo de $drift$ mistura os conceitos A e B, fazendo ambos ficarem ativos durante um determinado período. Com o passar do tempo, o número de ocorrência do conceito A diminui à medida que o conceito B aumenta;
- Moderado (incremental): este tipo de $drift$ trabalha com mais de dois conceitos, sendo que as diferenças entre eles são notáveis apenas em longos períodos de tempo.

A região de transição entre os conceitos é conhecida como zona de *drift* e está localizada entre i_x e $i_x + \triangle_x$, onde i_x é uma instância do espaço de todas as possíveis combinações dos valores dos atributos, $\triangle_x = 1$ para *drifts* abruptos e $\triangle_x > 1$ para *drifts* graduais (Figura 3.2).

De acordo com Stanley (2003), uma mudança de conceito gradual pode ser modelada utilizando a função α, que representa o domínio de um conceito sobre o outro. Nesta função, i_c é uma instância atual durante o período de $drift$. Antes de i_x, $\alpha = 1$, após $i_x + \triangle_x$, $\alpha = 0$. Quanto menor $\triangle_x$, mais rápida a alteração de conceito vai ser realizada.

$$\alpha = \frac{c - x}{\triangle_x} \tag{3.1}$$

Para que uma técnica de $drift$ *detection* seja eficiente, ela precisa ser capaz de:

- Detectar *drifts* rapidamente, porque detectar a alteração de um conceito rapidamente é essencial, ou seja, menos erros serão cometidos;

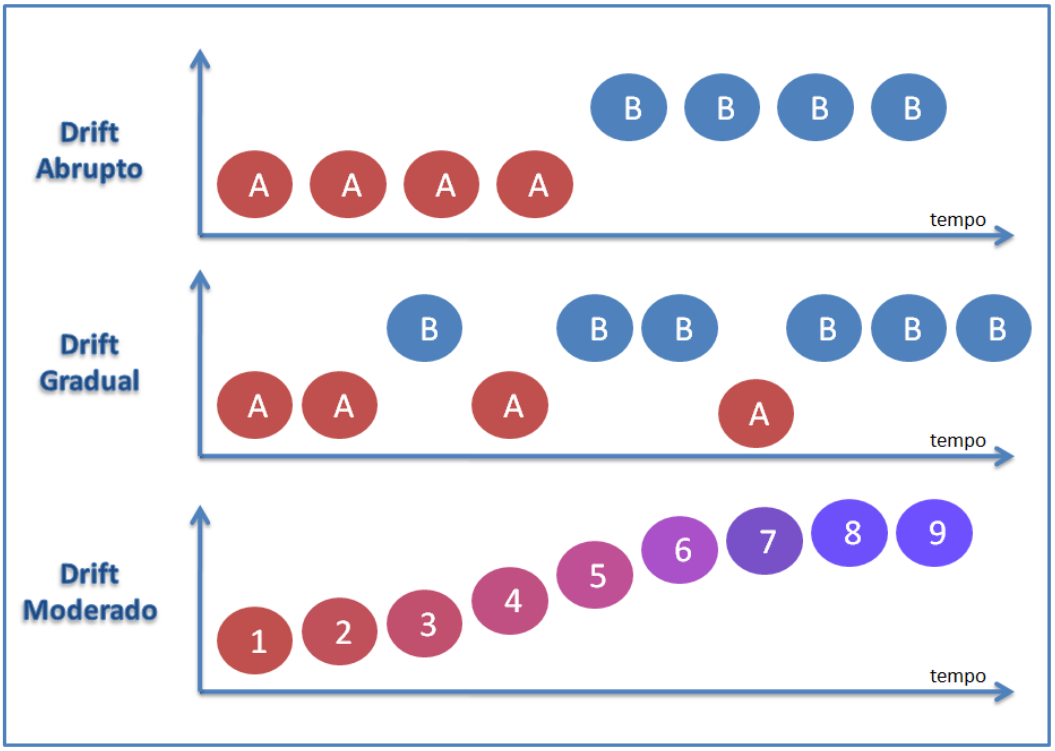

Figura 3.1: Ilustração de três tipos de *concept drift*.

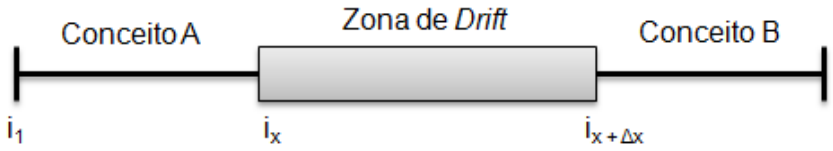

Figura 3.2: Ilustração de mudança de conceito (A - B) (Enembreck et al., 2007).

- Distinguir *drift* de ruído, à medida que alguns sistemas podem receber ruídos nos dados e essas anomalias não devem ser tratadas como alterações reais;
- Reconhecer e lidar com contextos recorrentes, ou seja, possuir a habilidade de identificar conceitos que alternam seus valores com algum grau de regularidade, como as estações do ano, por exemplo. Esses conceitos devem ser armazenados e reusados em situações futuras para que o sistema não repita o processo de aprendizagem.

Os diversos métodos existentes capazes de realizar a tarefa de *drift detection* podem ser classificados como métodos *batch* ou métodos *online*. Os métodos *batch*, ou *offline*, armazenam uma certa quantidade de instâncias para então realizar o aprendizado. Durante o intervalo entre os aprendizados, mudanças significativas podem acontecer e o modelo se tornar desatualizado, ocasionando classificações incorretas. Um exemplo deste tipo de método é o *windowing*. Nesta técnica, uma área de armazenamento chamada *window* é criada para funcionar como um container para as n instâncias mais recentes observadas;

essas instâncias são utilizadas para aprender o atual modelo de dados. A complexidade desta técnica é identificar um bom tamanho de janela, já que janelas pequenas garantem rápida adaptação, mas não podem identificar relações entre variáveis devido à visão particionada dos dados e janelas grandes possuem bons classificadores para zonas estáveis, mas perdem a capacidade de identificar alterações rapidamente. Os métodos *online* são capazes de atualizar o seu modelo toda vez que uma nova instância é observada. Nos últimos anos, diversas técnicas mais sofisticadas (*ensemble methods*) capazes de manter um grupo de classificadores tem surgido, é o caso do algoritmo *Weighted Majority* (WM) (Littlestone e Warmuth, 1994), *Streaming Ensemble Algorithm* (SEA) (Street e Kim, 2001), *Accuracy-weighted Ensemble* (AWE) (Wang et all., 2003), *Dynamic Weighted Majority* (Kolter e Maloof, 2003) e *Additive Expert* (AddExp) (Kolter e Maloof, 2005). Alguns desses métodos de conjunto são descritos a seguir com mais detalhes.

3.2 Métodos de conjunto

Um método de conjunto se caracteriza pela predição da classe de um registro elegendo a maioria dos votos feitas pelos classificadores base. Por meio da combinação de classificadores, as competências podem ser complementadas e por meio do treinamento do grupo, esses métodos são capazes de obterem desempenhos melhores que um classificador único.

3.2.1 Streaming Ensemble Algorithm

Street e Kim (2001) propuseram o *Streaming Ensemble Algorithm* (SEA). O funcionamento deste algoritmo ocorre como descrito abaixo e apresentado pela Figura 3.3.

- Os classificadores são construídos a partir de pequenos subgrupos de dados;
- Os classificadores construídos são combinados em um grupo de tamanho máximo fixo;
- As predições são realizadas utilizando o voto da maioria;
- Quando o grupo atinge seu tamanho máximo, um novo classificador será adicionado apenas se este melhorar o desempenho do grupo e um dos classificadores do grupo for removido.

Cada classificador corresponde a uma árvore de decisão construída a partir do subgrupo de dados por meio do algoritmo C4.5 (Quinlan, 1993). O desempenho é medido

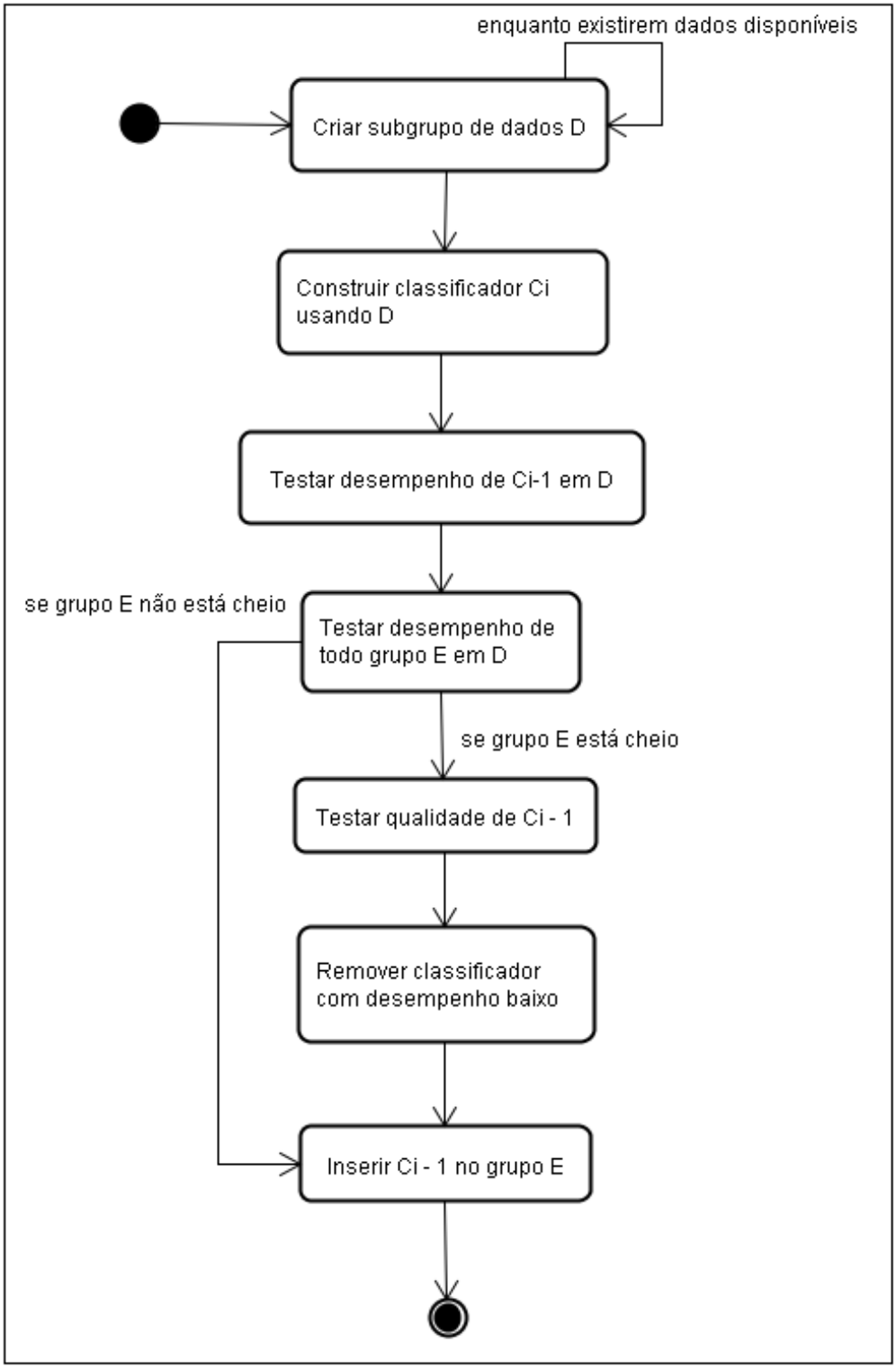

Figura 3.3: Fluxo de atividades do algoritmo SEA.

testando a nova árvore (C_i) e o grupo existente com o próximo subgrupo de dados (D).

Deve-se levar em consideração algumas limitações do algoritmo SEA:

- Nenhum peso é atribuído ao classificador para determinar o seu desempenho;

- O classificador é treinado em um grupo fixo de dados. Isto pode levar a um treinamento com dados inconsistentes.

3.2.2 Dynamic Weighted Majority

Dynamic Weighted Majority (DWM) é um método *ensemble* que contorna a limitação do *Weighted Majority* (Littlestone e Warmuth, 1994), permitindo que especialistas sejam adicionados ou removidos de acordo com as mudanças de desempenho. O DWM utiliza os quatro mecanismos descritos a seguir para lidar com *concept drift*:

- realiza o treinamento dos especialistas do grupo de forma *online*;
- atribui pesos aos especialistas de acordo com o desempenho deles;
- remove especialistas do grupo de acordo com o desempenho deles;
- cria especialistas de acordo com o desempenho do grupo.

No grupo de agentes especialistas (classificadores base) mantido pelo DWM, cada um implementa um algoritmo de aprendizagem. Por ser um método de coordenação genérico, é permitido que qualquer algoritmo de aprendizagem seja utilizado. O funcionamento do DWM é detalhado e posteriormente apresentado pela Figura 3.4.

1. para toda nova instância (i) recebida, o DWM:
   (a) inicia o peso de predição de cada classe com 0 (zero);
   (b) interage com cada especialista do grupo, passando a instância de treinamento:
      i. cada especialista realiza a sua predição local;
      ii. se o tempo de aprendizado não excedeu ($i \ mod \ p = 0$), os especialistas que realizarem predições incorretas terão o seu peso diminuído. O novo peso é determinado pela função:

$$w(j) = w(j) * \beta$$

      Onde:
      - $w(j)$: é o peso do especialista j;
      - β: é uma constante com valor entre 0 e 1, usada para diminuir o peso dos especialistas.
      iii. o peso de predição da classe da instância recebida é atualizado com base no peso do especialista em uso. Por exemplo, se a classe da predição local é interessante, o peso da classe interessante será atualizado.

2. após todos os especialistas terem realizado suas classificações, a classe com maior peso será considerada a predição global;

3. se o tempo de aprendizado não excedeu ($i \ mod \ p = 0$):

   (a) o peso de cada especialista é normalizado, onde o maior peso deve ser 1;

   (b) os especialistas com um peso menor que um mínimo determinado ($w[j] < \Theta$) são suprimidos, onde Θ é um limiar para a remoção de especialistas;

   (c) se a predição global for incorreta, um novo especialista é criado com peso 1.

   Deve ser observado que o parâmetro p (período de tempo) controla a frequência com que o coordenador cria novos especialistas, remove especialistas velhos, normaliza ou reduz o peso dos especialistas. Este parâmetro é necessário para ambientes com uma grande quantidade de ruídos.

4. Por fim, a instância é enviada para o treinamento dos especialistas.

No inicio do processo, o DWM cria um grupo contendo apenas um especialista e a sua predição será considerada a predição global. Quando o grupo contém diversos especialistas, DWM obtém a classificação local e o peso de cada especialista para definir a predição global. Como já mencionado, este é um método genérico que permite o uso de qualquer algoritmo de apredizagem como especialista.

3.2.3 Additive Expert

Kolter e Maloof (2005) apresentaram o algoritmo de conjunto *Additive Expert* (AddExp). Este é um método genérico capaz de utilizar qualquer especialista de forma *online* para a detecção de alterações de conceito. AddExp possui muitas similaridades em relação ao DWM, pois também mantém um conjunto de especialistas com pesos associados, o especialista que realizar classificação incorreta tem seu peso diminuído, se a predição global for incorreta um novo especialista é criado e adicionado ao grupo; e por fim, cada instância também é enviada para o treinamento dos especialistas. É importante ressaltar que existem duas versões deste algoritmo, AddExp.D para classes discretas e o AddExp.C para classes contínuas. O funcionamento geral de ambos é o mesmo, exceto para a predição das classes; enquanto o AddExp.D trabalha com um grupo discreto de classes, o AddExp.C trabalha com [0,1]. Pode-se observar na Figura 3.5, que nenhuma atividade é executada para remover especialistas ruims. Objetivando limitar o número de especialistas do conjunto, foram propostas duas técnicas de remoção de especialistas, conforme descrito a seguir:

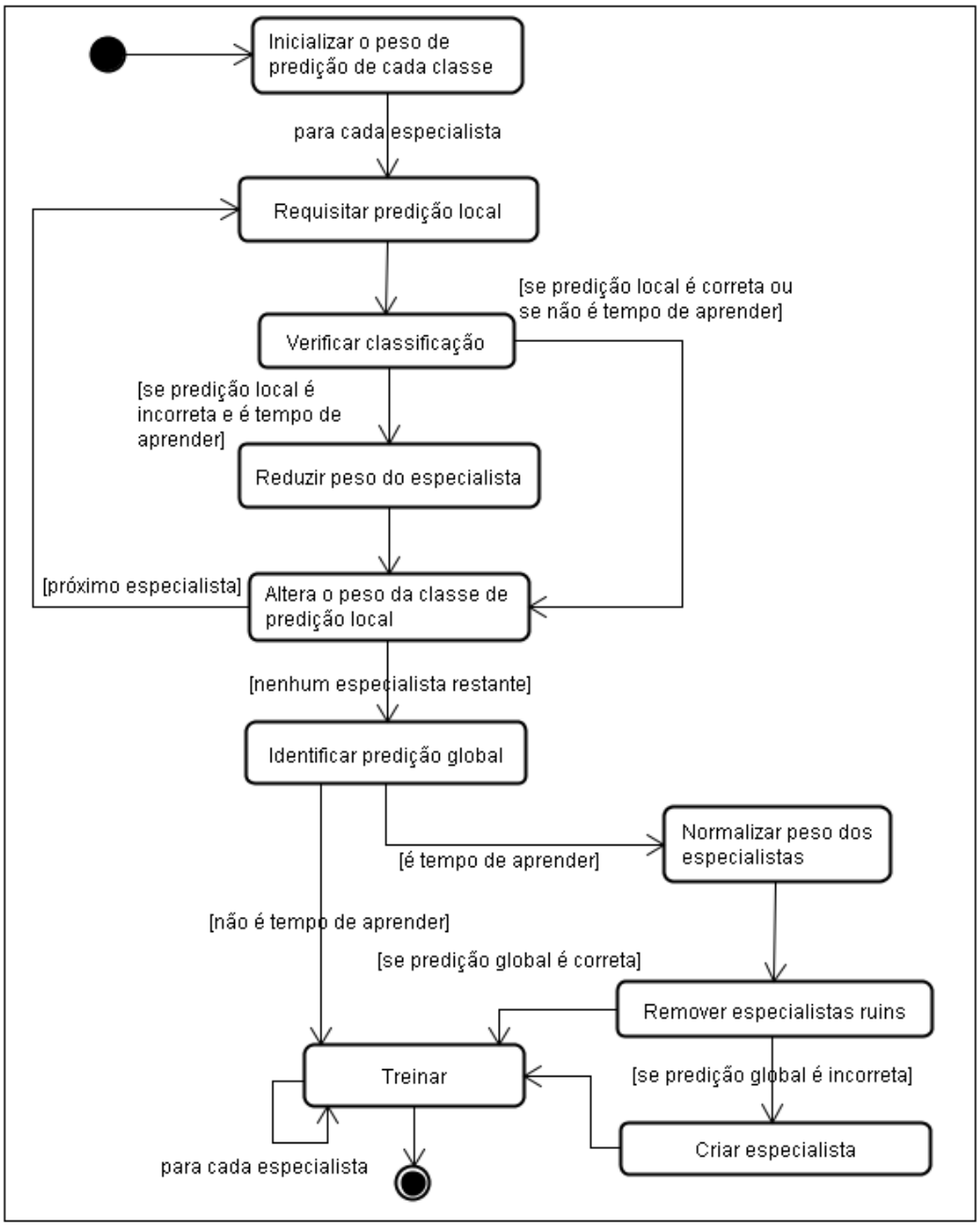

Figura 3.4: Fluxo de atividades do algoritmo DWM.

- Método 1 (*Oldest First*): Quando um novo especialista é adicionado ao conjunto, se o número de especialistas for maior que uma constante K definida, remove-se o especialista mais antigo antes de adicionar o novo membro ao grupo (Kolter e Maloof, 2005).

- Método 2 (*Weakest First*): Quando um novo especialista é adicionado ao conjunto, se o número de especialistas for maior que uma constante K definida, remove-se o especialista com o menor peso antes de adicionar o novo membro ao grupo (Kolter

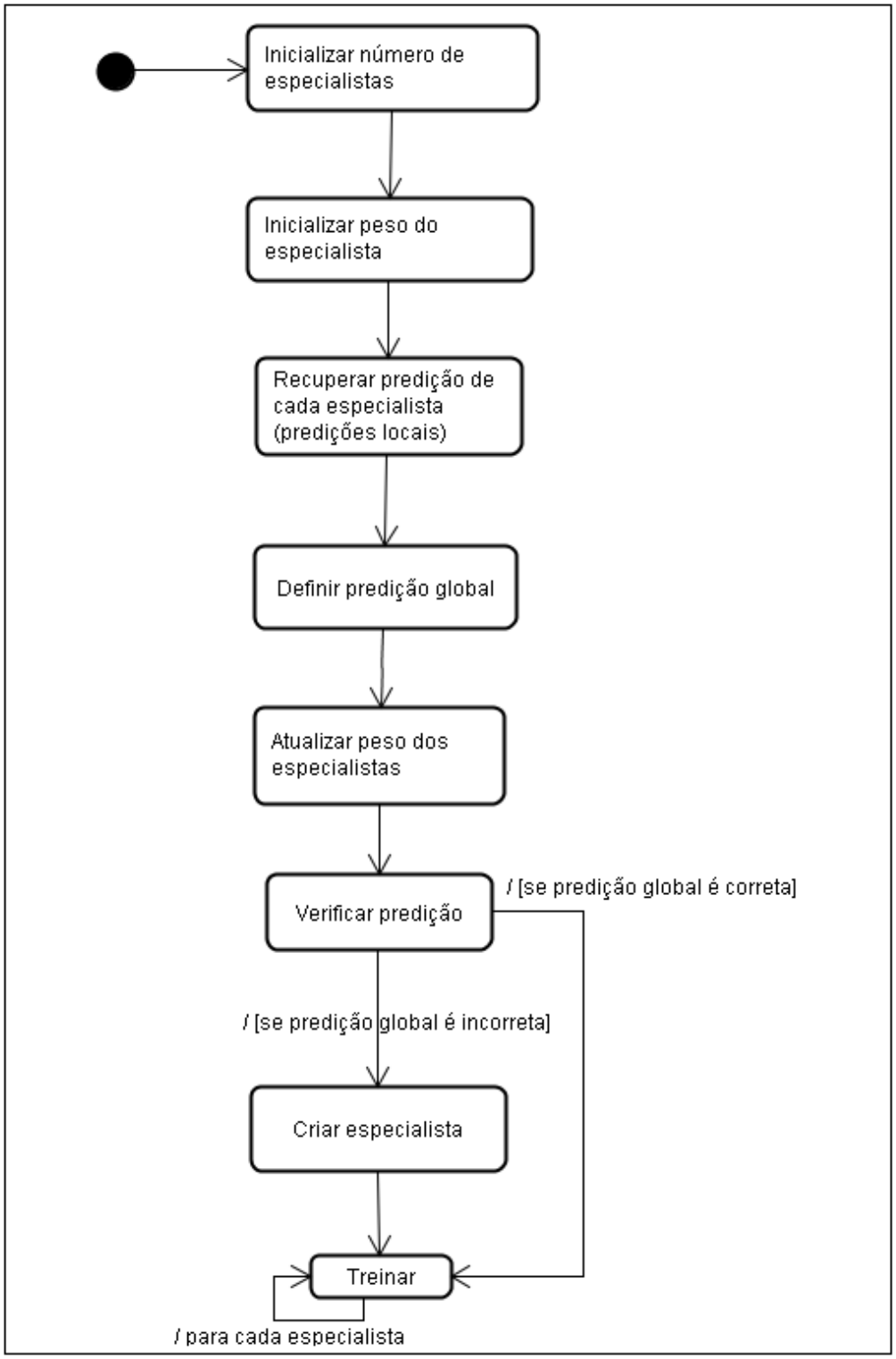

Figura 3.5: Fluxo geral de atividades do algoritmo AddExp.

e Maloof, 2005).

Para mais detalhes do funcionamento do algoritmo AddExp, consultar Kolter e Maloof (2005).

3.3 Métodos de Aprendizagem

Os métodos descritos a seguir utilizam técnicas de aprendizagem distintas. A rede Bayesiana trabalha com probabilidades e considera todas as instâncias do seu grupo (descrição do conceito) para realizar predições. Os algoritmos baseados em instância utilizam um grupo específico de instâncias para realizar classificações. Esses dois métodos são utilizados para o desenvimento dos experimentos e portanto, são descritos a seguir em detalhes.

3.3.1 Rede Bayesiana

Os processos de tomada de decisão e os problemas que a inteligência artificial se propõem a resolver, geralmente, são problemas de natureza não determinística que objetivam representar e tratar as incertezas. A incerteza pode ter origem em informações incompletas ou na imprecisão da definição de variáveis. O primeiro e mais abrangente formalismo matemático que surgiu para tratar as incertezas e auxiliar na tomada de decisões, foi a teoria da probabilidade. A teoria da probabilidade se mostra satisfatória para problemas com um pequeno número de variáveis, mas à medida que a quantidade de variáveis aumenta, problemas computacionais tendem a aparecer. Atualmente, vários métodos baseados na teoria da probabilidade são capazes de amenizar esses problemas, entre eles, as redes Bayesianas.

Uma rede Bayesiana é um grafo acíclico dirigido. Ele é utilizado para representar o conhecimento em um domínio com incertezas. Popular em meios como inteligência artificial, aprendizado de máquina e estatística, a rede Bayesiana combina princípios da teoria da probabilidade, estatística e ciência da computação. Sua estrutura é formada por nós e arestas; cada nó representa uma variável (discreta ou contínua) e cada aresta representa a conexão entre os nós, ou seja, a dependência probabilística entre eles. Portanto, uma aresta do nó A para o nó B representa a dependência estre as duas variáveis, indicando que o valor da variável B depende do valor da variável A ou que A influencia B. Nesse caso, o nó A é denominado pai ou predecessor do nó B e o nó B é nomeado filho ou descendente de A. Em uma rede Bayesiana, cada nó é condicionalmente independente de seus não descendentes, dados os seus predecessores imediatos, possibilitando assim reduzir o número de parâmetros de uma distribuição de probabilidade conjunta. A distribuição de probabilidade condicional de um grafo acíclico dirigido atende a propriedade de Markov (Spirtes, 1993), onde cada nó depende apenas de seus predecessores imediatos. Para os casos em que o nó não possui predecessor, diz-se que a sua distribuição de probabilidade

local é incondicional, para os outros casos, é condicional.

Probabilidade condicional é a probabilidade de um evento A ocorrer, dada a ocorrência de um evento B. Ela é denotada por $P(A|B)$. Uma das opções mais utilizadas para o cálculo da probabilidade é o teorema de *Bayes* (Equação 3.2), desenvolvido no século XVIII pelo inglês Thomas Bayes. Esta equação é utilizada quando não é possível determinar a probabilidade condicional diretamente e quando se tem $P(B|A)$, $P(A)$ e $P(B)$.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.2}$$

Onde:

- $P(A)$ e $P(B)$ são probabilidades *a priori* de A e B;
- $P(A|B)$ é a probabilidade *a posteriori* de A condicional a B;
- $P(B|A)$ é a probabilidade *a posteriori* de B condicional a A.

A probabilidade condicional é representada por uma tabela contendo a probabilidade local combinada com os valores dos predecessores, caracterizando uma tabela de probabilidade condicional.

A Figura 3.6 representa uma estrutura clássica de Pearl (1988) e ilustra algumas características de uma rede Bayesiana. Formada pelas variáveis Roubo, Terremoto, Alarme, JoãoLiga e MariaLiga, a topologia desta rede permite determinar que:

- um roubo pode fazer o alarme soar;
- um terremoto pode fazer o alarme soar;
- um alarme soando pode fazer João telefonar;
- o alarme soando pode fazer Maria telefonar.

Os nós Roubo e Terremoto são pais do nó Alarme, enquanto Alarme é pai de JoãoLiga e MariaLiga. Roubo e Terremoto não possuem pai e portanto não são influenciados por outro nó. Roubo e Terremoto afetam diretamente a probabilidade de o alarme soar, mas o fato de João e Maria ligarem só depende da variável alarme. Na figura, as distribuições condicionais são exibidas sob a forma de uma tabela de probabilidade condicional. Cada linha da tabela contém a probabilidade condicional de cada valor de nó para uma combinação possível de valores dos nós superiores. Os nós que não possuem pai, apresentam apenas uma linha, representando a probabilidade incondicional.

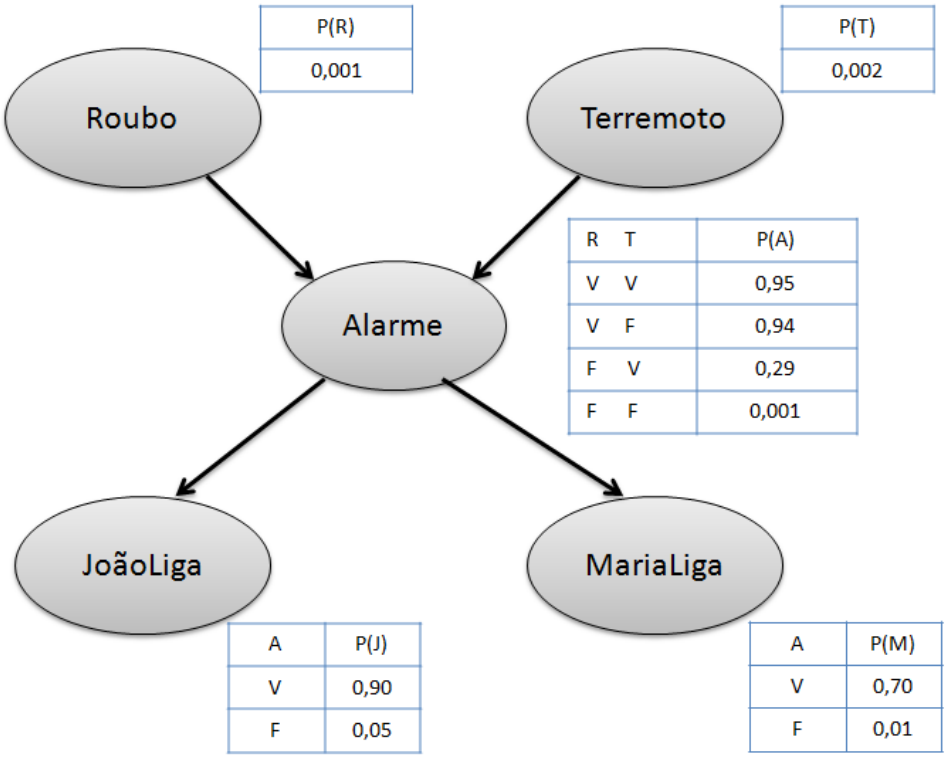

Figura 3.6: Exemplo de rede Bayesiana típica.

A construção de uma rede Bayesiana é considerada uma tarefa difícil, principalmente em situações de informação incompleta. Uma rede Bayesiana pode ser construída utilizando o conhecimento de um especialista, um processo de aprendizagem automática ou uma combinação dos dois. Uma construção automática pode ser realizada a partir de dados na forma de casos. Cada caso representa um exemplo, uma situação do mundo. Cada variável se torna um nó e os valores da variável se tornam os estados do nó. Neste caso, a construção é dividida em duas partes:

- aprendizagem da estrutura;
- aprendizagem dos parâmetros.

A aprendizagem da estrutura determina a disposição das arestas na rede, determinando a dependência e independência das variáveis; no exemplo da Figura 3.6 , determinaria as conexões entre os nós Roubo, Terremoto, Alarme, JoãoLiga e MariaLiga. A aprendizagem dos parâmetros determina a tabela de probabilidade condicional para cada nó e é simplificado se todos os casos fornecem valores para cada variável da rede. O aprendizado pode ser realizado automaticamente por meio de algoritmos, sendo que a escolha de um algoritmo ou outro deve considerar a existência de informação faltante

e o conhecimento da estrutura da rede, como pode ser observado na Tabela 3.1 [Murphy, 1998]. Por não ser objetivo deste estudo analisar os algoritmos de aprendizagem Bayesiana, Russell e Norving (2004) podem ser consultados para mais detalhes.

Tabela 3.1: Algoritmos de aprendizagem de rede Bayesiana

Estrutura	**Observabilidade**	**Algoritmo**
Conhecido	Total	*Maximum Likelihood Estimation*
Conhecido	Parcial	*EM* (ou *gradientascent*)
Desconhecido	Total	*Search through model space*
Desconhecido	Parcial	*EM* + *search through model space*

Após a construção da rede é possível realizar inferências probabilísticas. Inferência probabilística consiste em obter conclusões à medida que novas informações ou evidências são conhecidas. Para cada variável com valor conhecido, o valor é inserido no nó como uma evidência; a inferência probabilística é então executada para encontrar crenças para todas as outras variáveis. As crenças formadas após a inferência são chamadas probabilidades posteriores, sendo probabilidades *a priori* as probabilidades antes que qualquer evidência seja informada. As inferências probabilísticas geram crenças para cada nó da rede, mas não alteram a base de conhecimento. Se a evidência utilizada for verdadeira e possível de utilização no futuro, a base de dados pode ser alterada para refletir essa nova evidência por meio do processo de revisão de probabilidades. Existem vários métodos capazes de realizarem inferências probabilísticas, entre eles, $JunctionTree$, $NodeAbsorptions$ e $Sampling$. Para mais detalhes sobre esses métodos o leitor é convidado a consultar Russell e Norving (2004).

Uma rede Bayesiana também pode ser modelada como um classificador. Neste caso, o classificador pode ser representado por uma rede Bayesiana de estrutura simples, que tem o nó classe como nó pai de todos os outros nós atributos, como apresentado pela Figura 3.7. O classificador mais conhecido entre os classificadores Bayesianos é o *Naive Bayes*. O *Naive Bayes* é um modelo simples, que considera as variáveis do domínio condicionalmente independentes, mas que em muitos domínios, teve resultados de desempenho de predição tão bons quanto aos de algoritmos de aprendizagem mais complexos (Friedman et al, 1997).

O funcionamento de uma rede Bayesiana para classificar uma determinada instância x é ilustrado pela Figura 3.8. A atividade 1 representa uma instância x sendo recebida como uma evidência. No passo 2, os valores dos atributos de x são utilizados para realizar uma inferência probabilística na rede (nós: A_1, $A_2...A_n$) e obter a classe com maior pro-

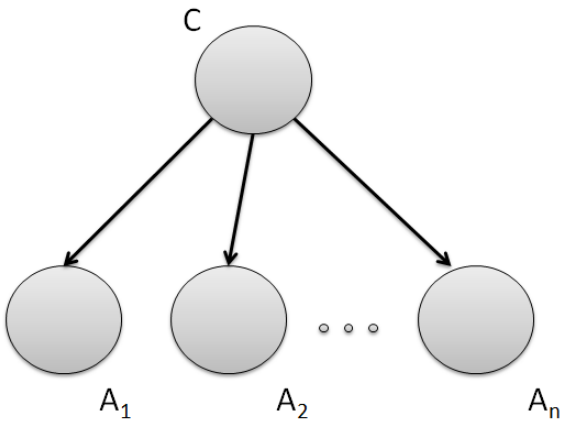

Figura 3.7: Estrutura de uma rede Bayesiana *Naive Bayes*. (Friedman et al., 1997)

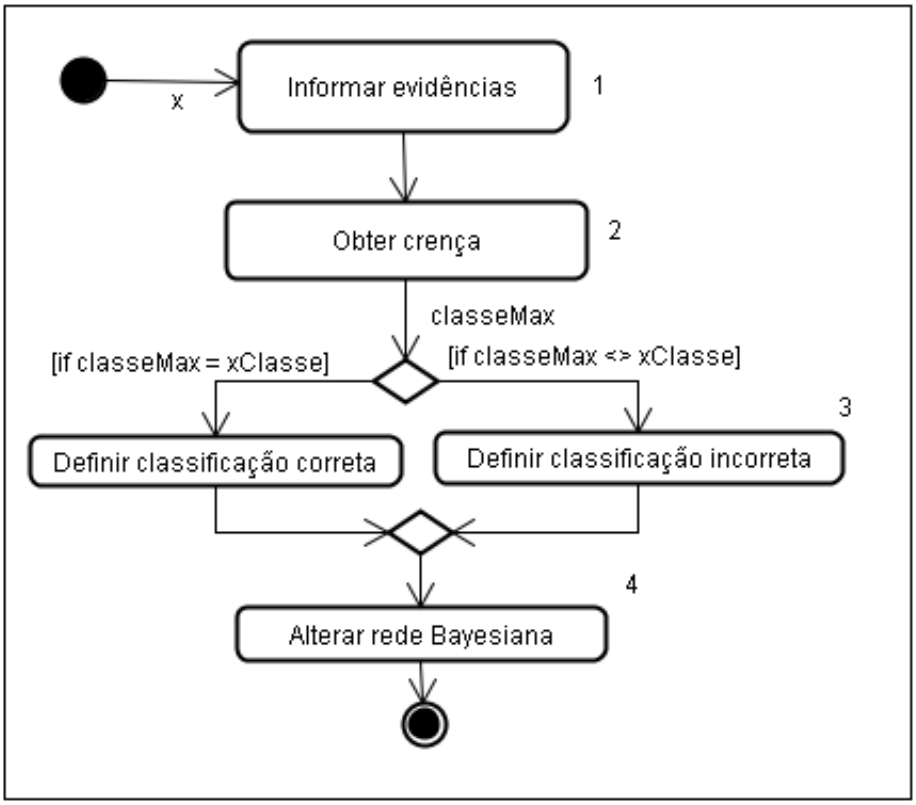

Figura 3.8: Fluxo de atividades de um classificador Bayesiano.

Tabela 3.2: Descrição das variáveis do fluxo de atividades de uma classificador Bayesiano

x	Instância de treinamento
classeMax	Classe com maior probabilidade
xClasse	Classe da instância de treinamento x

babilidade ($classeMax$) (nó: C). Se a classe com maior probabilidade é igual a classe da

instância x ($xClasse$), a classificação é considerada correta, caso contrário a classificação estará incorreta. Para que a rede seja capaz de refletir as informações atuais, em ambos os casos a evidência x é utilizada para o aprendizado dos parâmetros (atualização das tabelas de probabilidades condicionais).

Diversos softwares estão disponíveis no mercado para a contrução e manipulação de redes Bayesianas. Entre as opções encontradas, algumas estão listadas na Tabela 3.3. O Netica, apesar de sua versão livre limitar a quantidade de nós da rede, é uma ferramenta robusta que permite tanto o aprendizado de parâmetros como a realização de inferências, além de possuir uma biblioteca completa em Java com funções de fácil utilização. Alguns exemplos, utilizando a API do Netica em Java, são apresentados no Apêndice A. Eles monstram como as redes Bayesianas podem ser construídas, como a aprendizagem das probabilidades condicionais a partir de um conjunto dados pode ser realizada e como as inferências probabilísticas são executadas.

Tabela 3.3: Ferramentas para construção e manipulação de redes Bayesianas.

Nome	**Linguagem**	**Aprendizagem de Parâmetros**	**Livre**	**Inferência**
Bayda	Java	Sim	Sim	Não
Banko	Java	Não	Sim	Não
BNJ	Java	Não	Sim	Sim
CIspace	Java	Não	Sim	Sim
Hydra	Java	Sim	Sim	Sim
Java Bayes	Java	Não	Sim	Sim
RISO	Java	Não	Sim	Sim
UnBBayes	Java	Não	Sim	Sim
Vibes	Java	Sim	Sim	Sim
Web Weaver	Java	Não	Sim	Não
Netica	Java/C++	Sim	Limitado	Sim

3.3.2 Aprendizagem Baseada em Instâncias

IBL (*Instance-Based Learning*) é uma metodologia onde as predições são realizadas utilizando instâncias específicas. Os algoritmos que fazem parte dessa família são: IB1, IB2, IB3, IB4 e IB5. Eles utilizam a técnica do *"Nearest Neighbour"* (NN) para classificar cada nova instância. Apesar da simplicidade e da capacidade em realizar classificações após um processo de aprendizagem trivial, o algoritmo NN possui diversas limitações, como a não tolerância a ruídos, não tolerância a atributos irrelevantes e o fato de ser um classificador computacionalmente caro. Cada uma das variações dos algoritmos da

família IBL surgiu para contornar alguma das limitações do NN ou de uma versão anterior a sua. IB1 é o algoritmo mais simples da família, IB2 foca na redução da necessidade de armazenamento e o IB3 é tolerante a instâncias ruidosas. O algoritmo IB4 tem como objetivo contornar a sensibilidade a atributos irrelevantes e o IB5 busca tratar atributos ausentes; entretanto, os últimos dois não são detalhados neste trabalho.

O conjunto de exemplos de treinamento de um algoritmo IBL expressa a descrição do conceito. Tal descrição é determinada por meio de funções de similaridade e de classificação, definidas por Aha (1991), que são:

- sF: função responsável por calcular a similaridade entre a instância de treinamento x e as instâncias da descrição do conceito;
- cF: fornece a classificação da instância x por meio dos resultados da função de similaridade e dos registros de desempenho em classificações anteriores das instâncias pertencentes ao conceito;
- aF: atualizador da descrição do conceito. Ele é responsável pelos registros de desempenho de classificação e pela modificação da descrição do conceito. Para decidir se uma instância será incluída na descrição do conceito utiliza-se a instância x, os resultados da função de classificação e a descrição atual do conceito.

Algoritmo IB1

O IB1 é o algoritmo mais simples da família IBL e muito similar ao NN, à medida que ele armazena todas as instâncias de treinamento. A diferença entre o NN e o IB1 está no fato de o IB1 apresentar tolerância a valores ausentes de atributos, avaliar as instâncias incrementalmente e normalizar o escopo dos atributos.

A Figura 3.9 ilustra o funcionamento do IB1, cujos passos são:

1. para cada instância de treinamento x recebida, o IB1 calcula a similaridade entre x e cada instância y armazenada na descrição do conceito;
2. a instância mais similar ($yMax$) é selecionada;
3. se a classe de x for igual a classe de $yMax$, a classificação é considerada correta, caso contrário a classificação é considerada incorreta;
4. a instância de treinamento x é armazenada pelo classificador na sua descrição do conceito.

Na primeira atividade, a função de similaridade utilizada é responsável por calcular a similaridade entre a instância de treinamento x e cada uma das instâncias y

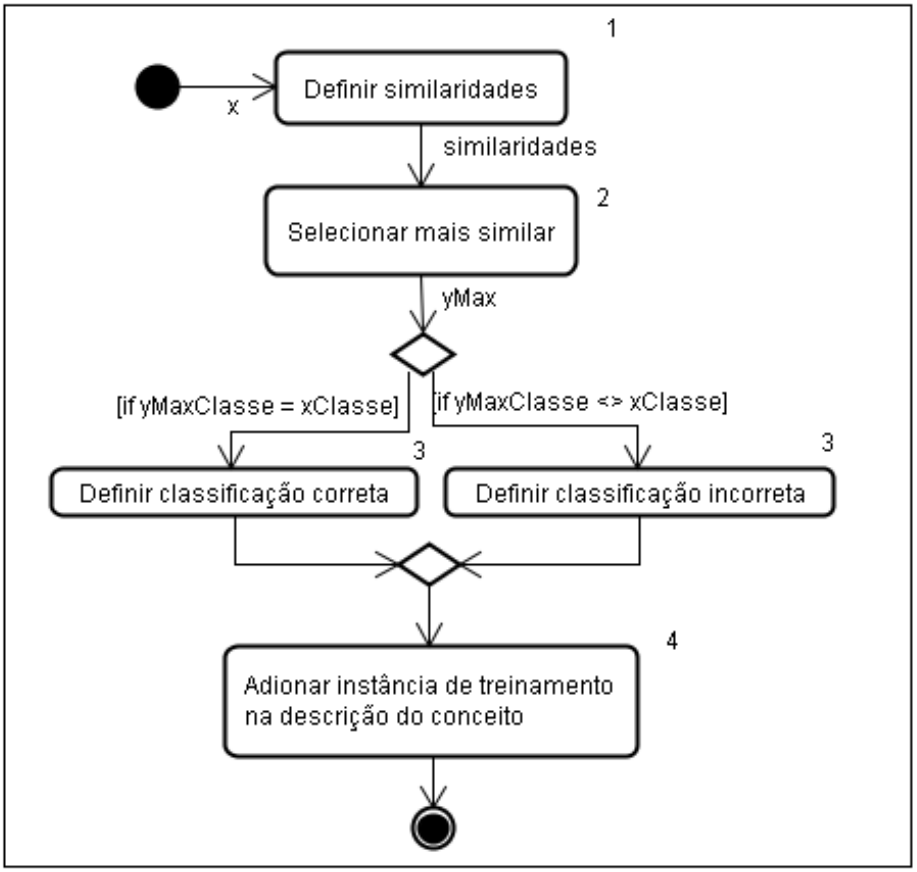

Figura 3.9: Fluxo de atividades do algoritmo IB1.

Tabela 3.4: Descrição das variáveis do fluxo de atividades do algoritmo IB1

x	Instância de treinamento
similaridades	Similaridade entre cada instância y armazenada e a instância de treinamento x
yMax	Instância mais similar armazenada
yMaxClasse	Classe da instância ($yMax$) mais similar armazenada
xClasse	Classe da instância de treinamento x

armazenadas. Essa função é baseada na distância Euclidiana e também é utilizada pelos algoritmos IB2 e IB3. Tem-se como objetivo obter a instância y mais similar a instância x, onde a menor distância representa a maior similaridade. Uma possível função de similaridade (Aha, 1991) é apresentada na Equação 3.3:

$$Similaridade(x, y) = -\sqrt{\sum_{i=1}^{n} f(x_i, y_i)} \tag{3.3}$$

Onde:

- n: é o número de atributos que descrevem a instância;

- $f(x_i, y_i) = (x_i - y_i)^2$ para atributos numéricos;
- $f(x_i, y_i) = \begin{cases} 1\ if\ (x_i = x_y) \\ 0\ if\ (x_i \neq x_y) \end{cases}$ para atributos qualitativos.

O exemplo abaixo calcula a similaridade entre *Oferta A* - *Oferta B* e *Oferta A* - *Oferta C*.

Tabela 3.5: Objetos de exemplo para cálculo de similaridade.

	Preço	**Quantidade**
Oferta A	15	8
Oferta B	10	5
Oferta C	12	6

$$Similaridade(A, B) = -\sqrt{(15 - 10)^2 + (8 - 5)^2} \tag{3.4}$$

$$Similaridade(A, B) = -\sqrt{25 + 9} = -5.83 \tag{3.5}$$

$$Similaridade(A, C) = -\sqrt{(15 - 12)^2 + (8 - 6)^2} \tag{3.6}$$

$$Similaridade(A, C) = -\sqrt{9 + 4} = -3.60 \tag{3.7}$$

A similaridade entre *Oferta A* e *Oferta B* é -5.83, já a similaridade entre a *Oferta A* e *Oferta C* é -3.60. Portanto, a *Oferta A* é mais similar à *Oferta C*, sendo $Similaridade(A, C) > Similaridade(A, B)$.

Na quarta atividade (Figura 3.9), observa-se que todas as instâncias de treinamento são armazenadas na descrição do conceito, o que causa um alto custo de armazenamento e de processamento para a classificação. Para que uma instância possa ser classificada, ela precisa ser comparada a todas as instâncias persistidas. Essa limitação do IB1 passou a ser contornada pelo seu sucessor, o IB2.

Algoritmo IB2

O algoritmo IB2 diferencia-se do IB1 apenas na forma como as instâncias são armazenas na descrição do conceito. Visando a redução de dados armazenados, o IB2 armazena apenas instâncias que não foram classificadas corretamente, à medida que as instâncias classificadas incorretamente são as instâncias de fronteira e, portanto, delimitam o conceito.

A Figura 3.10 ilustra o funcionando do IB2, cujos passos são:

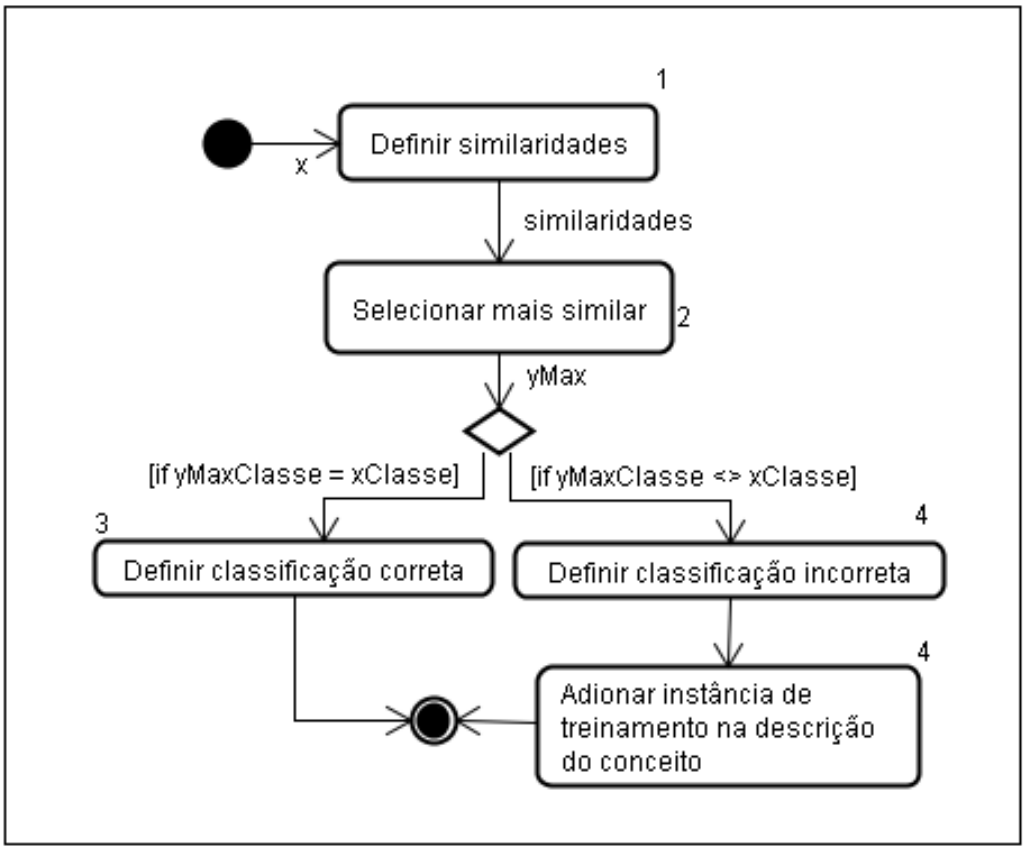

Figura 3.10: Fluxo de atividades do algoritmo IB2.

Tabela 3.6: Descrição das variáveis do fluxo de atividades do algoritmo IB2

x	Instância de treinamento
similaridades	Similaridade entre cada instância y armazenada e a instância de treinamento x
$yMax$	Instância mais similar armazenada
yMaxClasse	Classe da instância ($yMax$) mais similar armazenada
xClasse	Classe da instância de treinamento x

1. para cada instância x de treinamento, o IB2 calcula a similaridade entre x e cada exemplo y armazenado na descrição do conceito;

2. a instância mais similar ($yMax$) é a selecionada;

3. se a classe de x for igual a classe de $yMax$, a classificação é considerada correta;

4. se a classe de x for diferente da classe de $yMax$, a classificação é considerada incorreta e a instância de treinamento é inserida na descrição do conceito do classificador.

Como já dito, o IB2 possui a mesma função de similaridade e de classificação do IB1. A diferença está no quarto passo, onde apenas as instâncias classificadas incorre-

tamente são armazenadas. Foi mostrado em Aha (1991), que o IB2 diminuiu consideravelmente a necessidade de armazenamento, mas tornou-se mais sensível à presença de ruídos nos dados de treinamento que o IB1. Essa sensibilidade aos ruídos se deve ao armazenamento das instâncias classificadas incorretamente, uma vez que instâncias com ruídos geralmente não são classificadas corretamente.

Para contornar a sensibilidade aos ruídos, foi proposta uma extensão do IB2 com a capacidade de determinar quais instâncias armazenadas devem ser utilizadas no momento da classificação. Esta extensão materializou-se no algoritmo IB3.

Algoritmo IB3

O algoritmo IB3 visa contornar a intolerância a ruídos por meio de um teste que determina se uma dada instância será relevante para as classificações futuras ou se tal instância representará apenas ruído. Para tal, mantém-se um registro de classificação com cada instância armazenada. Este registro identifica o desempenho de classificação da instância.

A Figura 3.11 ilustra o funcionamento do IB3, cujos passos 1, 3 e 4 são os mesmos do IB2. Entretanto, no passo 2, a instância aceitável com maior similaridade ($yMax$) é selecionada. Para uma instância ser considerada aceitável, o seu limite inferior do intervalo de precisão de classificação da instância deve ser maior que o limite superior do intervalo de frequência da classe observada.

Para calcular o limite superior de frequência da classe observada os seguintes passos são necessários:

- recuperar o número de instâncias de treinamento processadas que são elementos dessa classe (f). Por exemplo, se a classe observada é interessante, deve-se então recuperar a quantidade de instâncias armazenadas cuja classe seja interessante;
- recuperar o número total de instâncias de treinamento processadas (n);
- calcular a frequência da classe observada (p), fazendo f/n; e
- aplicar a frequência da classe observada (p) e o número de instâncias armazenadas (n) na Equação 3.8 abaixo para encontrar o limite superior da frequência da classe observada:

$$\frac{p + \frac{z^2}{2n} + z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \tag{3.8}$$

Onde:

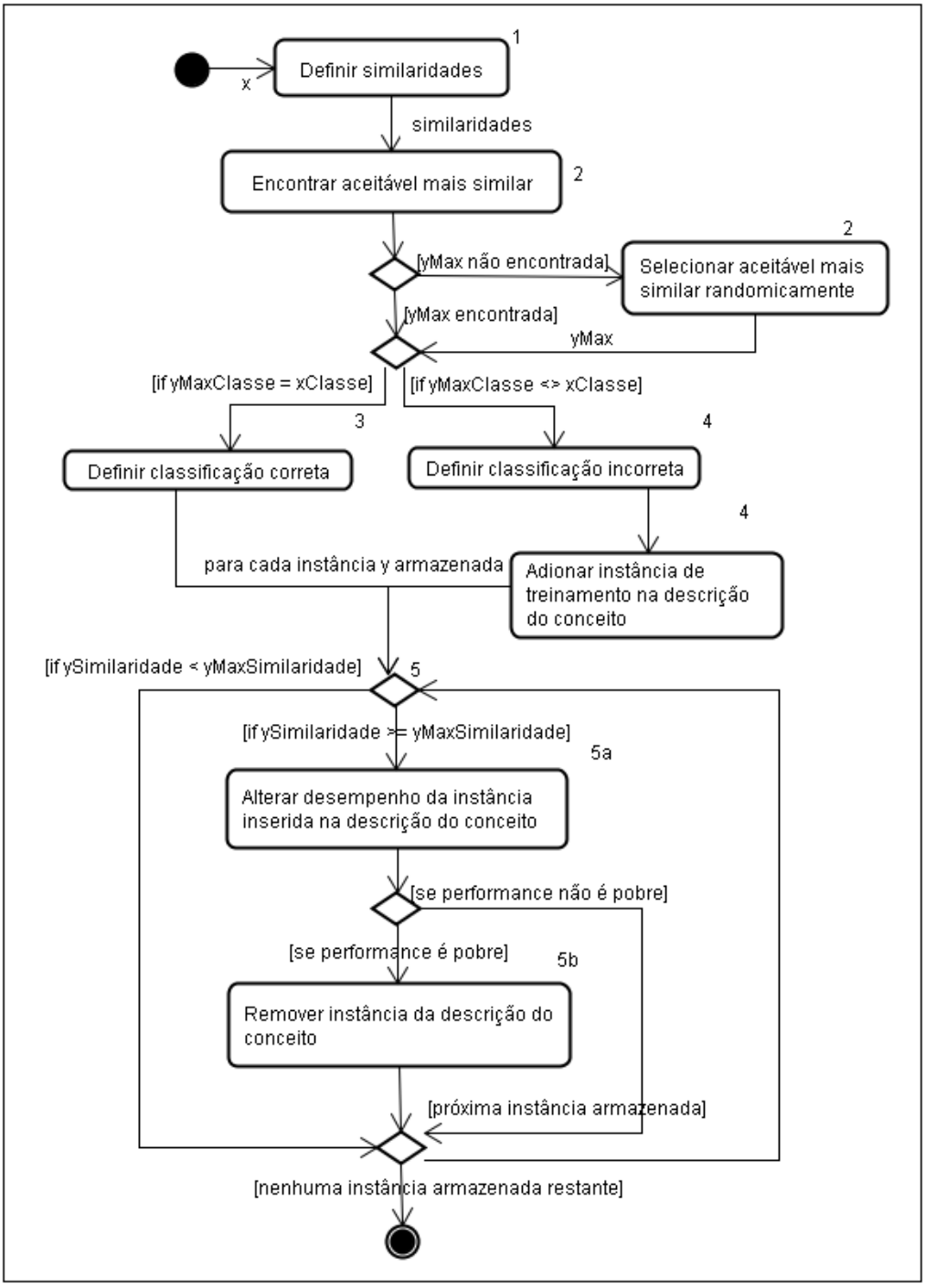

Figura 3.11: Fluxo de atividades do algoritmo IB3.

- p: é a frequência da classe observada;
- n: é número de instâncias de treinamento previamente processadas;
- z: é o nível de confidência.

Para calcular o limite inferior da precisão de classificação da instância os seguintes passos são necessários:

Tabela 3.7: Descrição das variáveis do fluxo de atividades do algoritmo IB3

x	Instância de treinamento
similaridades	Similaridade entre cada instância y armazenada e a instância de treinamento x
yMax	Instância mais similar aceitável armazenada
yMaxClasse	Classe da instância mais similar aceitável ($yMax$) armazenada
xClasse	Classe da instância de treinamento x
ySimilaridade	Similaridade entre a instância armazenada y e a instância de treinamento x
yMaxSimilaridade	Similaridade entre a instância mais similar armazenada $yMax$ e a instância de treinamento x

- recuperar o número de predições corretas da instância (r);
- recuperar o número total de tentativas de classificação, ou seja, o número de predições corretas mais o número de predições incorretas (n);
- calcular a precisão da classificação da instância (p) , fazendo r/n; e
- aplicar a precisão de classificação da instância (p) e o número de tentativas de classificação (n) na Equação 3.9 para encontrar o limite inferior da precisão de classificação da instância:

$$\frac{p + \frac{z^2}{2n} - z\sqrt{\frac{p(1-p)}{n} + \frac{z^2}{4n^2}}}{1 + \frac{z^2}{n}} \tag{3.9}$$

Onde:

- p: é a precisão da instância observada;
- n: é número de tentativas de classificação da instância;
- z: é o nível de confidência.

Os únicos parâmetros requisitados pelo IB3 são os níveis de confidência para a aceitação ou remoção de uma instância. Os níveis apresentados aqui se referem aos propostos por Aha (1991), onde um nível de confidência alto ($z = 0.9$) é usado para que a aceitação de uma instância se torne difícil, enquanto que um nível de confidência baixo ($z = 0.75$) é usado para a remoção de uma instância, fazendo que até instâncias com desempenho de classificação moderado sejam removidas.

Uma instância aceitável será selecionada randomicamente dentro do grupo de instâncias armazenadas caso nenhuma instância aceitável seja encontrada.

No quinto passo, a similaridade da instância aceitável mais similar é comparada com a similaridade de cada exemplo y armazenado na descrição do conceito. Apenas os exemplos com uma similaridade maior ou igual a similaridade de $yMax$ terão o seu desempenho alterado, estando sujeitos a remoção da descrição do conceito. Para decidir quais instâncias possuem um desempenho não satisfatório e que poderão ser removidas, o mesmo teste de aceitação de instância é realizado, porém com um nível de confidência menor, $z = 0.75$. Este procedimento visa descartar instâncias com um desempenho moderado. Se o limite superior do intervalo de precisão de classificação da instância for menor que o limite inferior do intervalo de frequência da classe observada, a instância será removida.

Utilizando mecanismos diferenciados para atualização dos registros e filtragem de instâncias com ruídos, o IB3 é capaz de reduzir a demanda por armazenamento presente no IB1 e contornar a sensibilidade aos ruídos presente no IB2. O IB3 mostra-se um método com grande potencial para desempenhar a atividade de detecção de *drift* e pode obter resultados ainda melhores se o trabalho for realizado por um grupo de especialistas IB3 ao invés de um especialista IB3 isolado; por isso, as próximas seções apresentam métodos capazes de coordenar um conjunto de especialistas. Os métodos de conjunto, por meio do estabelecimento de um comitê de especialistas, trabalham com complementação de competências e voto do melhor resultado. Esses métodos devem evitar subconjuntos idênticos ou disjuntos, para evitar que os erros sejam os mesmos ou não correlacionados. A seguir, são descritos alguns trabalhos que utilizaram os métodos de conjunto e de aprendizagem descritos anteriormente.

3.4 Trabalhos Relacionados

STAGGER (Littlestone e Warmuth, 1986) foi o primeiro sistema desenvolvido capaz de tolerar ruídos e *drifts*. O método utiliza estatística Bayesiana para tolerar ruídos regulares e fazer a distinção entre ruído e *drift*. Este sistema representa os conceitos como um grupo de caracterizações simbólicas duplamente ponderadas, conforme as três definições para o mesmo conceito utilizadas na avaliação do desempenho do sistema: (1) *color red or shape squarish*, (2) *size small or shape circular*, (3) *color (blue or green)*. STAGGER adapta a descrição dos seus conceitos por meio de modificações incrementais aos pesos associados com as caracterizações. Apesar de não ser uma solução completa, limitando-se ao aprendizado de combinações booleanas de atributos e necessitando receber *feedbacks*, os resultados dos experimentos realizados mostraram que o STAGGER é capaz de identificar as mudanças de conceitos e de diferenciar *drifts* de ruídos.

O algoritmo IB3 foi utilizado por Enembreck et al. (2007) em um processo de

negociação bilateral multicritério entre um agente comprador e um agente vendedor. O objetivo do trabalho foi testar o IB3 como técnica de detecção de *drift* no aprendizado de políticas de oferta. No sistema desenvolvido, o agente vendedor inicia o processo de negociação com um determinado conceito (definição de oferta interessante) e o agente comprador tenta descobrir esta definição. Com o passar do tempo, o agente vendedor altera a sua definição de oferta interessante e o agente comprador deve detectar e se adaptar a essa mudança. As instâncias geradas são similares as instâncias do sistema STAGGER e as sequências de conceitos definidos permitiram testar o IB3 para mudanças de conceito de forma abrupta, moderada e gradual. Os resultados dos experimentos mostraram que o algoritmo IB3 é capaz de detectar alterações rapidamente e de adaptar a sua descrição do conceito visando a melhoria do desempenho.

Baseados no algoritmo *Weighted Majority*, Kolter e Maloof (2003) propuseram o uso do método genérico *Dynamic Weighted Majority*. Dois sistemas experimentais foram avaliados, utilizando o *Naive Bayes* (NB) e o *Incremental Tree Inducer* (ITI) como algoritmos base. Ambos os sistemas, DWM-NB e DWM-ITI, tiveram seus desempenhos avaliados usando os conceitos STAGGER. Os resultados obtidos, quando comparados ao *Weighted Majority* ou ao uso de um especialista exclusivo, mostraram que o DWM é capaz de aprender *drifts* de conceito quase tão bem quanto os algoritmos de aprendizagem base aprendem cada conceito individualmente, realizando predições mais eficazes e recuperando o desempenho mais rapidamente após uma situação de *drift*. Também comprovou-se em (Kolter e Maloof, 2007), que o DWM conseguiu resultados melhores que o SEA (Street e Kim, 2001) para todos os casos. Enquanto o SEA atingiu uma acurácia entre 90 e 94%, o DWM atingiu 96% - 98%.

Visando melhorar os resultados obtidos por Enembreck et al. (2007), Enembreck et al. (2008) utilizou uma técnica de detecção de *drift* baseada em conjunto. O processo de negociação bilateral multicritério em que o agente comprador tenta descobrir a descrição de oferta interessante do agente vendedor foi mantido, porém, na nova arquitetura, o agente comprador possui um grupo de agentes especialistas (cada um correspondendo a um método de aprendizagem) controlados pelo algoritmo *Dynamic Weighted Majority*. Os resultados obtidos para mudanças abruptas, graduais e moderadas comprovaram que o algoritmo DWM elevou o desempenho do IB3 para todos os casos. Isso foi possível porque técnicas baseadas em conjunto fazem o uso de mais informações para tomar decisões e são menos propícias a tendências que um especialista isolado.

Pereira et al. (2010) propuseram a adaptação da abordagem proposta por Enembreck et al. (2008), adequando a mesma por meio de estratégias baseadas em buscas sequências e binárias para determinar preços de vendas de computadores. Esta adaptação

permitiu tratar situações em que não se tem todos os atributos para formar a instância de classificação.

Kolter e Maloof (2005) testaram os algoritmos AddExp.C e AddExp.D com os conceitos STAGGER e também com um conjunto de dados envolvendo ruídos. Quando comparado ao DWM, o AddExp teve um desempenho ligeiramente inferior apenas nos momentos de utilização dos conceitos STAGGER, conseguindo se sobressair em todas as outras situações. Os métodos de remoção permitiram que um número significativamente menor de especialistas fosse gerado.

Diversos estudos foram desenvolvidos utilizando o aprendizado Bayesiano no processo de aprendizagem. Zeng e Sycara (1998) utilizaram a regra de Bayes para aprender o preço de reserva do oponente em um processo de negociação com um único critério. Em Lin et al. (2006), o aprendizado Bayesiano é utilizado para determinar a probabilidade do oponente possuir um dos perfis de um grupo fixo de possíveis perfis do oponente. Hindriks e Tykhonov (2008) apresentaram um *framework* baseado no aprendizado Bayesiano capaz de aprender as preferências e as prioridades do oponente em um processo de negociação bilateral automatizada multicritério. Este método trabalha com hipóteses sobre a estrutura das preferências do oponente e a racionalidade do processo de negociação. O objetivo é calcular $P(h_t|b_t)$, onde b_t é uma oferta proposta no tempo t. Para obter alguma informação sobre a utilidade do oponente associado a b_t, assume-se que os agentes utilizam uma estratégia baseada na concessão. O aprendizado acontece pela movimentação durante o processo; toda vez que uma oferta é recebida do oponente, ela funciona como uma evidência e é utilizada para alterar as probabilidades associadas com cada hipótese por meio da regra de Bayes. O método foi testado em diversos domínios e comparado a estratégias como ABMP (Jonker et al., 2006), que não utiliza as informações de utilidade do oponente para a negociação. Os resultados mostraram a eficiência em criar um modelo do oponente visando a melhoria do processo de negociação.

Zheng (1998) propôs um método chamado *Naive Bayesian Classifier Committe* (NBC) com o objetivo de melhorar o desempenho de aprendizado dos classificadores Bayesianos. A idéia deste método é gerar um grupo de classificadores Bayesianos em experimentos sequenciais para a formação de um comitê. O NBC gera n classificadores Bayesianos utilizando diferentes subgrupos de atributos, onde n é o numero de atributos. No processo de classificação, cada classificador do comitê calcula a probabilidade de uma determinada instância pertencer a cada classe possível. As probabilidades são somadas e a classe com o maior valor é escolhida como predição final. Os resultados dos experimentos mostraram que em média, um comite de classificadores Bayesianos aumenta significativamente a precisão das predições dos classificadores Bayesianos. Este método não utiliza

pesos para o processo de votação.

3.5 Considerações Finais

Os métodos de conjunto, por meio do estabelecimento de um grupo de especialistas, são capazes de trabalhar com a complementação de competências e com um sistema de votação. O algoritmo SEA gera árvores de decisão a partir de blocos de dados lidos sequencialmente. Este método trabalha com um grupo de classificadores onde o número máximo é fixo, portanto, toda vez que um classificador é adicionado, outro é removido. A classificação de uma instância é realizada por meio de um processo simples de votação da maioria. Os experimentos apresentaram o SEA capaz de identificar situações de *drift* corretamente, entretanto, atualmente são preferíveis métodos que consideram o peso de cada classificador no processo de votação, que é o caso do DWM e AddExp.

Os métodos DWM e AddExp inovam ao permitir que classificadores sejam adicionados ou removidos do grupo dinamicamente, respondendo muito bem a ambientes dinâmicos, apresentando desempenho superior a outros métodos que mantém um grupo fixo de especialistas ou que armazenam e aprendem a partir de exemplos encontrados anteriormente, além de possuirem uns dos melhores resultados na detecção de *drift* entre os o metódos *ensemble* desenvolvidos até o momento.

Em Enembreck et al. (2008), o DWM foi utilizado para coordenar um grupo de especialistas IB3 (algoritmo de aprendizagem baseado em instâncias) e apresentou resultados satisfatórios para a detecção de *drift*. O algoritmo IB3, por sua vez, possui características valiosas para um processo de negociação, como robustez contra ruídos e atributos irrelevantes. Essas características permitem que os agentes distinguam as reais alterações de ruídos e detectem alterações na política de oferta a um baixo custo. Uma técnica que trata ruídos e dados ausentes de forma automática, é a rede Bayesiana, que possibilita a modelagem de relacionamentos complexos por meio de uma estrutura de visualização clara e de fácil entendimento, permitindo a fácil integração de conhecimento *a priori*; mas que, ao contrário do IB3, considera todas as instâncias do grupo para realizar predições.

Este trabalho tem o interesse de analisar a capacidade do DWM em coordenar diferentes métodos de aprendizagem e de verificar a eficiência dos algoritmos IB3 e rede Bayesiana em um ambiente com a existência de variações de conceitos. Optou-se pela utilização do DWM como método de conjunto por ele possuir características valiosas como: atribuir peso aos especialistas e realizar predições por meio do voto da maioria, assim como por motivos de comparação com o estudo já desenvolvido por Enembreck

et al. (2008). Para isso, foram desenvolvidos dois ambientes que simulam processos de negociação, em um deles o DWM coordena especialistas IB3 e no outro, o DWM coordena especialistas rede Bayesiana. O desenvolvimento deste ambiente e os resultados dos experimentos são apresentados no próximo capítulo.

Capítulo 4

Ambiente de Teste e Experimentos

O ambiente desenvolvido visa testar a eficiência da rede Bayesiana e do algoritmo IB3 coordenados pelo método DWM na detecção de *drift* abrupto, moderado e gradual. Por motivos de comparação, optou-se por reproduzir a configuração DWM-IB3 de Enembreck et al. (2008) e utilizar um ambiente com as mesmas características para o desenvolvimento e testes da configuração DWM-RB (Rede Bayesiana). As características do ambiente e os resultados dos experimentos são apresentados neste capítulo.

4.1 Ambiente

Para avaliar a proposta de estudo foram definidos alguns módulos de software que simulam um processo de negociação bilateral entre um agente comprador e um agente vendedor. O agente vendedor inicia o processo de negociação com a publicação de uma definição de oferta interessante e o agente comprador tenta gerar ofertas que sejam interessantes para o seu oponente. Com o passar do tempo, o agente vendedor muda o seu conceito de oferta interessante e o agente comprador deve automaticamente detectar e se adaptar a essa mudança.

Duas configurações foram desenvolvidas. Na primeira, a classificação das ofertas e a detecção de alteração no conceito do vendedor são realizadas por um conjunto de especialistas IB3 coordenados pelo algoritmo DWM e na segunda, o DWM coordena um grupo de redes Bayesianas. Cada especialista mantém uma base atualizada de instâncias, que representa a descrição do conceito (DC). A Figura 4.1 ilustra a arquitetura do ambiente de teste.

A geração das ofertas se baseia no sistema STAGGER (Littlestone e Warmuth, 1986). A Tabela 4.1 apresenta o domínio dos atributos para a definição das ofertas. As Tabelas 4.2, 4.3 e 4.4 apresentam a descrição conceitual de ofertas interessantes. Estes

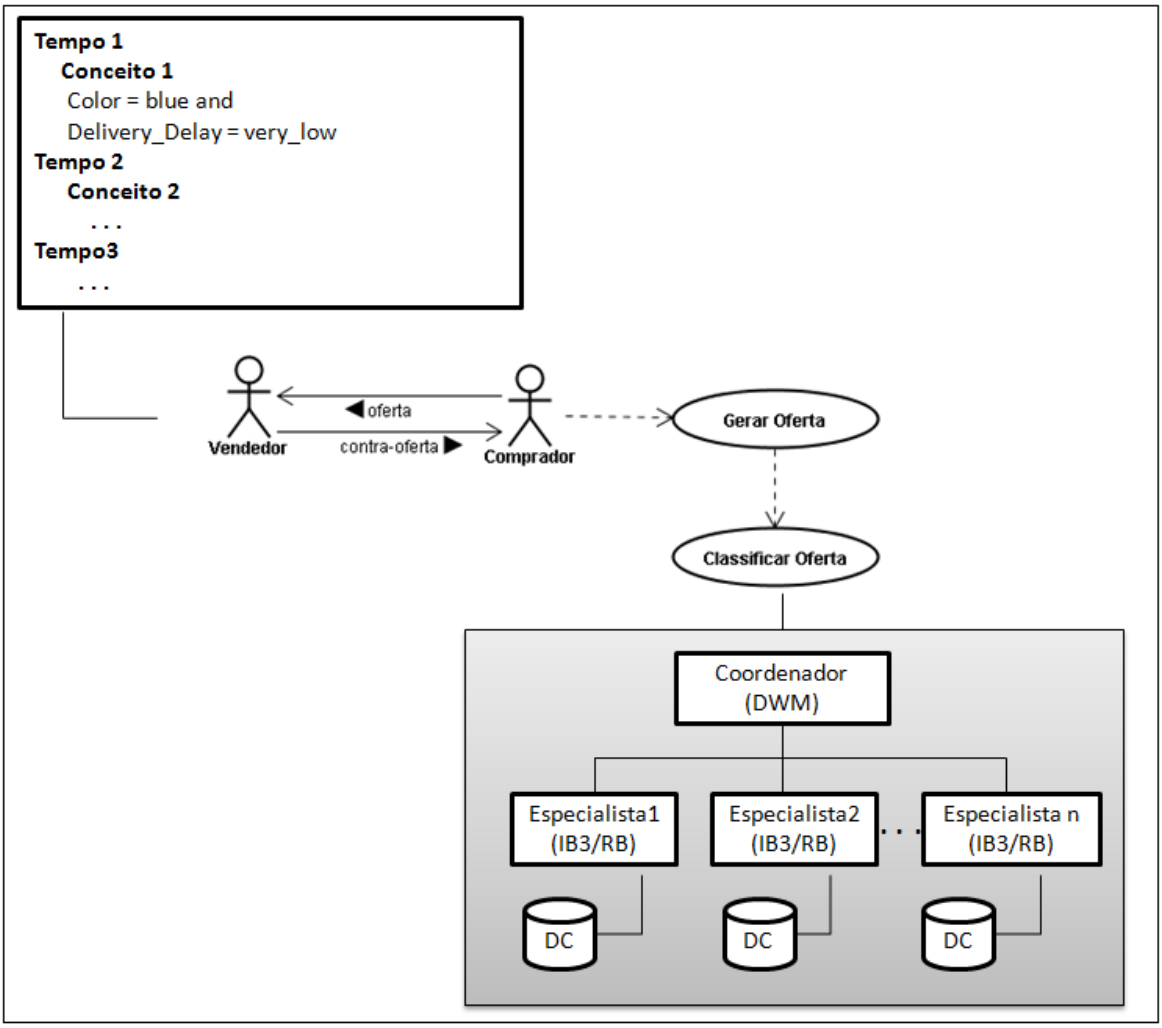

Figura 4.1: Arquitetura do ambiente de negociação.

conceitos e os atributos usados no ambiente de teste foram baseados em Enembreck et al. (2008).

Tabela 4.1: Descrição dos atributos do ambiente de teste.

Atributo	**Domínio**
Color	[black, blue, cyan, brown, red, green, yellow, magenta]
Price	[very low, low, normal, high, very high, quite high, enormous, non salable]
Payment	[0, 30, 60, 90, 120, 150, 180, 210, 240]
Amount	[very low, low, normal, high, very high, quite high, enormous, non ensured]
Delivery Delay	[very low, low, normal, high, very high]

O processo inicia pela geração de um conjunto T de 50 instâncias de teste para cada conceito (c). Estas instâncias são utilizadas para avaliar a acurácia do sistema. O DWM, no papel de coordenador do processo, repassa individualmente cada instância

Tabela 4.2: Descrição conceitual para *drift* abrupto.

ID	Description
1	(Price = normal and Amount = high) or (Color = brown and Price = verylow and DeliveryDelay = high)
2	(Price = high and Amount = veryhigh) and (DeliveryDelay = verylow)
3	(Price = verylow and Payment = 0 and Amount = high) or (Color = red and Price = low and Payment = 30)
4	(Color = black and Payment = 90 and DeliveryDelay = verylow) or (Color = magenta and Price = high and DeliveryDelay = verylow)
5	(Color = blue and Payment = 60 and Amount = low and DeliveryDelay = normal) or (Color = cyan and Amount = low and DeliveryDelay = normal)

Tabela 4.3: Descrição conceitual para *drift* moderado.

ID	Description
1	(DeliveryDelay = verylow and Amount = verylow)
2	(DeliveryDelay = verylow and Amount = low)
3	(DeliveryDelay = verylow and Amount = normal)
4	(DeliveryDelay = verylow and Amount = high)
5	(DeliveryDelay = verylow and Amount = quitehigh)
6	(DeliveryDelay = verylow and Amount = veryhigh)
7	(DeliveryDelay = verylow and Amount = enormous)
8	(DeliveryDelay = verylow and Amount = nonensured)

Tabela 4.4: Descrição conceitual para *drift* gradual.

ID	Description
A	(Color = blue and Delivery Delay = verylow)
B	(Color = black and Price = high) or (Color = magenta and Payment = 0)

de teste para cada especialista efetuar a sua predição local. Após todos os especialistas terem efetuado a sua predição para uma mesma instância de T, o DWM escolhe a classe com maior peso como sendo a predição global. Este processo se repete até que todas as instâncias de teste tenham sido classificadas. A acurácia do sistema é então calculada por meio da Equação 4.1. As atividades deste processo, de avaliação do desempenho do

sistema, são apresentadas pela Figura 4.2.

$$Acuracia(c, i) = \frac{TP + FN}{|T|} \times 100 \tag{4.1}$$

Onde:

- TP: (*True Positives*) é o número de instâncias de teste interessantes classificadas corretamente;
- FN: (*False Negatives*) é o número de instâncias de teste não interessantes classificadas corretamente;
- $|T|$: é o tamanho do grupo de instâncias de teste.

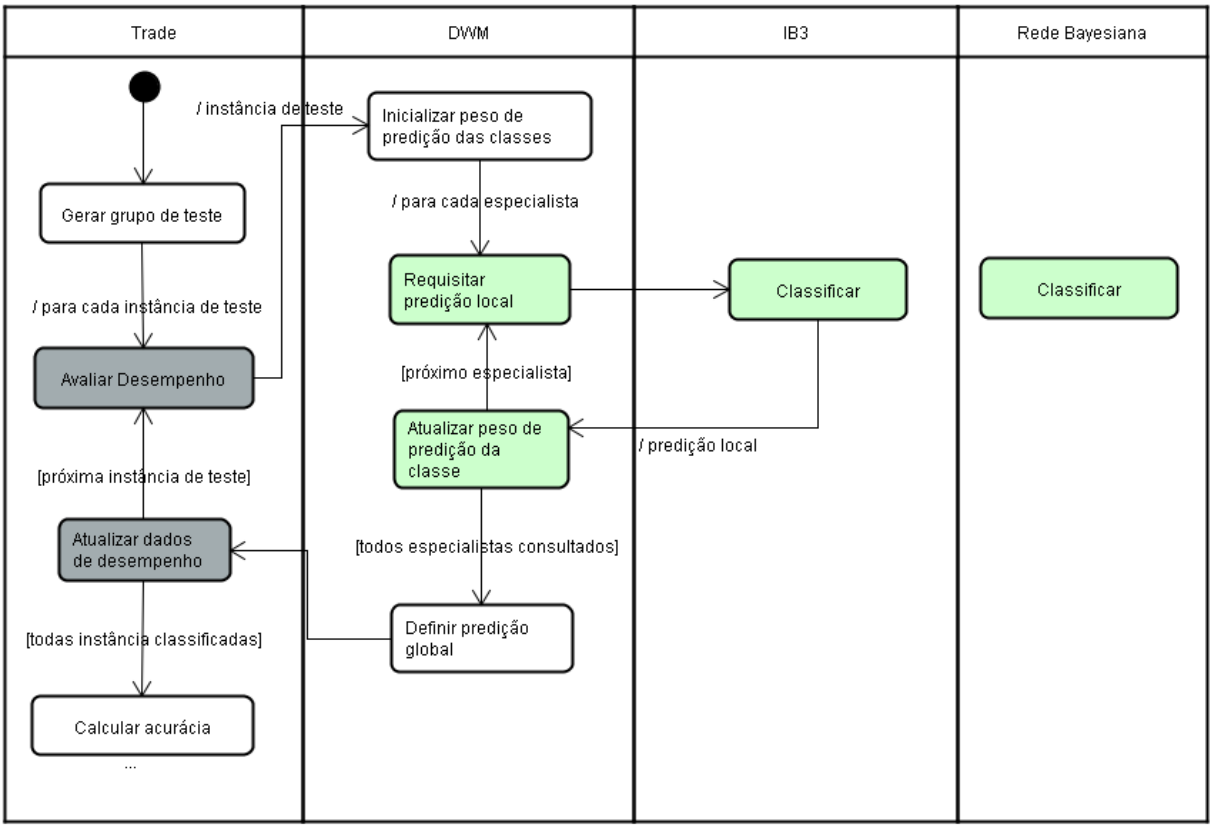

Figura 4.2: Fluxo de atividades para a avaliação do desempenho do sistema.

Para cada conceito também são geradas 50 instâncias de treinamento. Os especialistas IB3 realizam a classificação utilizando a função de similaridade descrita anteriormente, enquanto que as redes Bayesianas realizam inferências probabilísticas para obter crenças a partir das evidências informadas. De acordo com o resultado da classificação da instância de treinamento, o peso de cada classificador é definido, especialistas com um desempenho baixo são removidos, novos especialistas são adicionados ao grupo

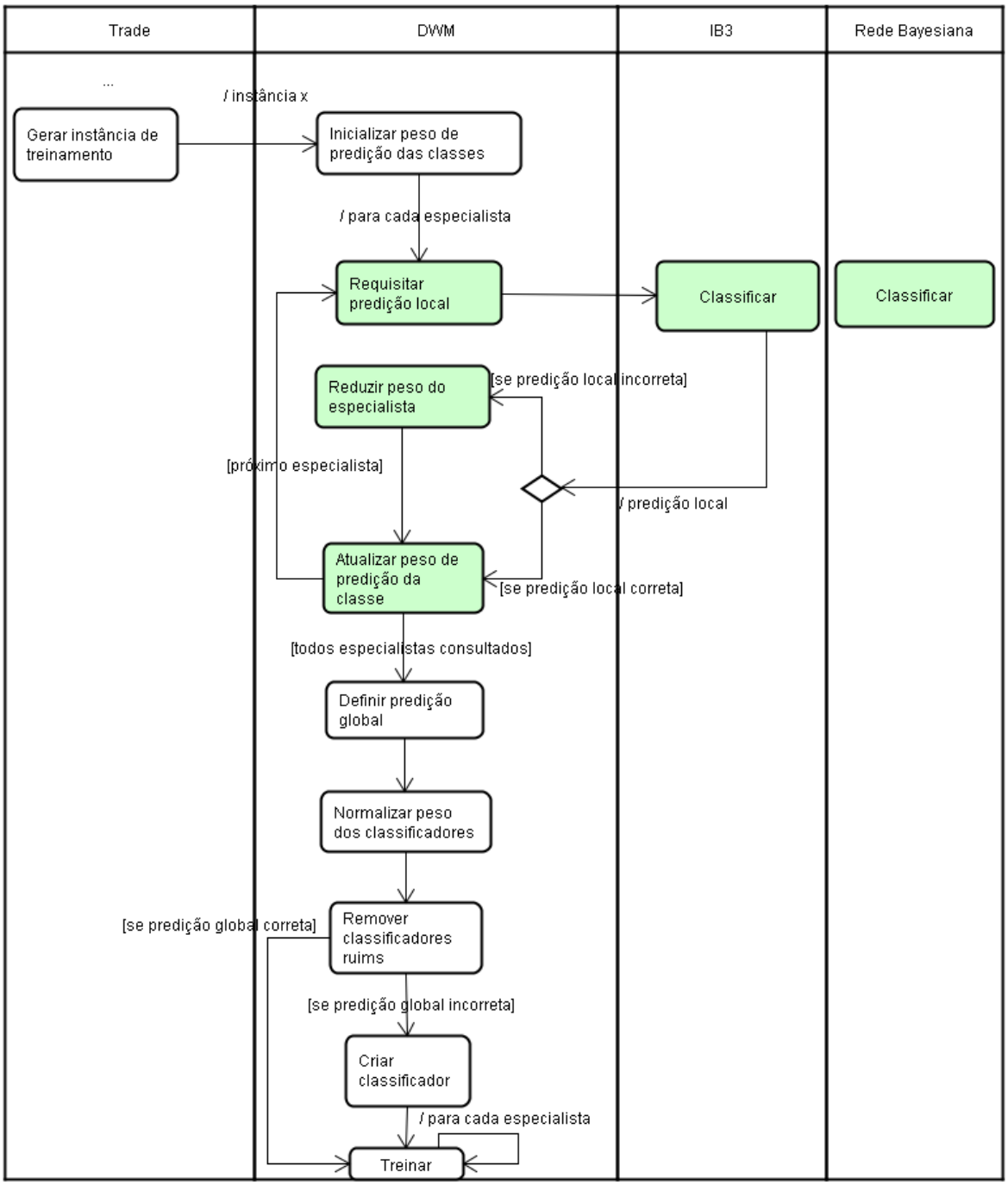

Figura 4.3: Fluxo de atividades para a classificação de uma instância.

quando necessário e por fim, cada instância de treinamento é enviada para o treinamento dos especialistas. Este procedimento pode ser visualizado na Figura 4.3. A atividade de atualização da descrição do conceito fica restrita ao momento do treinamento, como apresentado pela Figura 4.4.

As redes Bayesianas foram construídas de forma automática utilizando a API do Netica (Norsys Software Corp, 2009) e no momento da criação, receberam uma distribuição de probabilidades condicionais uniforme. Cada nó da rede corresponde a uma

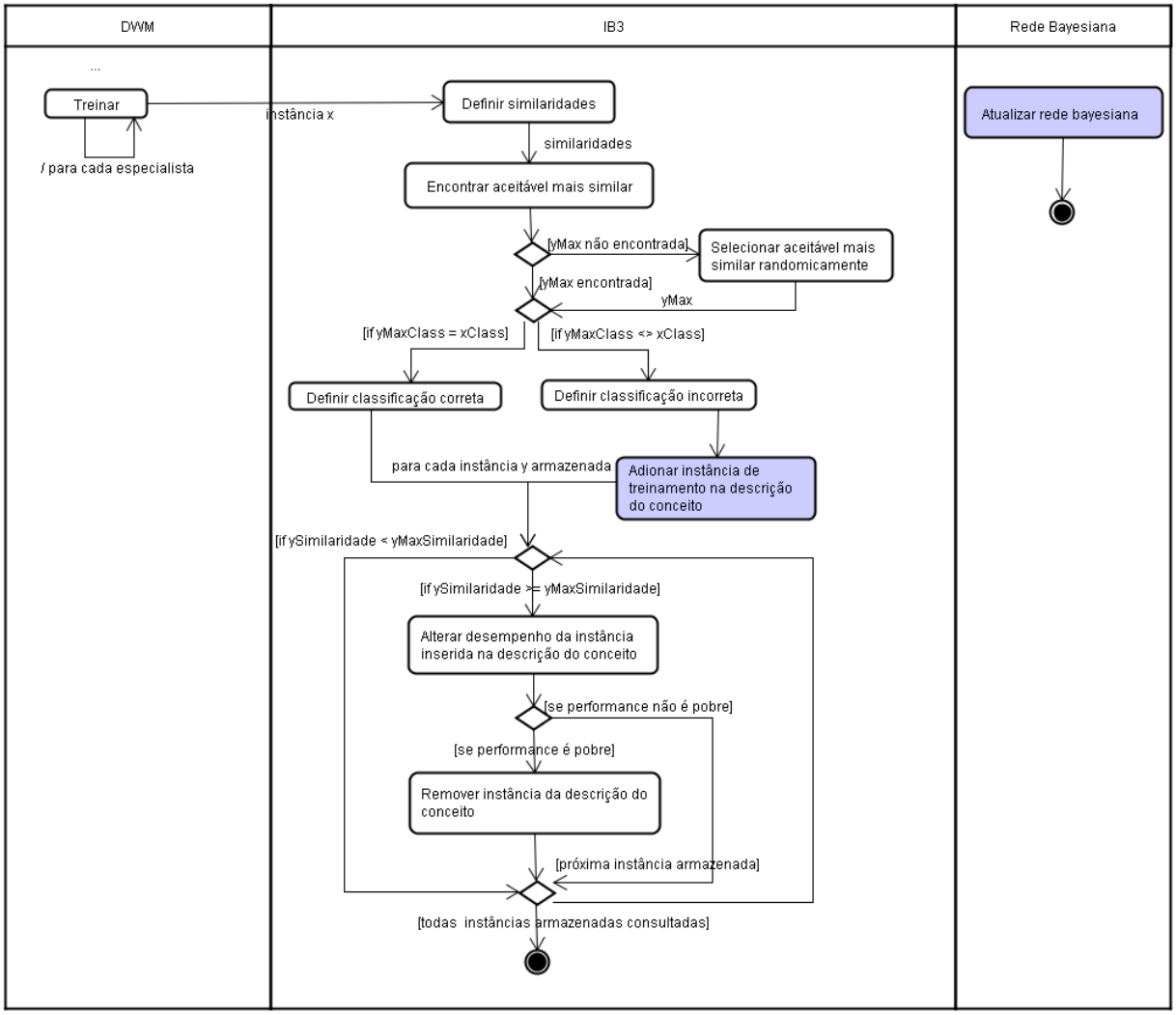

Figura 4.4: Fluxo de atividades para o treinamento de especialistas.

variável escalar a saber: *Color*, *Price*, *Payment*, *Amount*, *DeliveryDelay* e *ClassLabel*. O nó *ClassLabel* é o nó pai de todos os outros nós atributos e representa a classe de uma determinada oferta, que pode ser classificada como interessante ou não interessante. Cada nova instância de treinamento corresponde a um novo caso e os seus valores são utilizados para realizar uma inferência probabilística na rede e obter a classe com maior probabilidade. A inferência probabilística não altera as probabilidades condicionais da rede, ela é executada para calcular a probabilidade de uma oferta ser interessante dados os valores observados dos atributos *Color*, *Price*, *Payment*, *Amount* e *DeliveryDelay*. Para as inferências probabilísticas, o framework utilizou o algoritmo JunctionTree. Na aprendizagem dos parâmetros, o framework utilizou o algoritmo Counting, por ser simples, rápido e indicado para situações em que não há muita falta de informação. A arquitetura da rede Bayesiana é ilustrada pela Figura 4.5. A seguir são apresentados alguns exemplos de código para a criação de rede Bayesiana, realização de inferências probabilísticas e

atualização das tabelas de probabilidades condicionais.

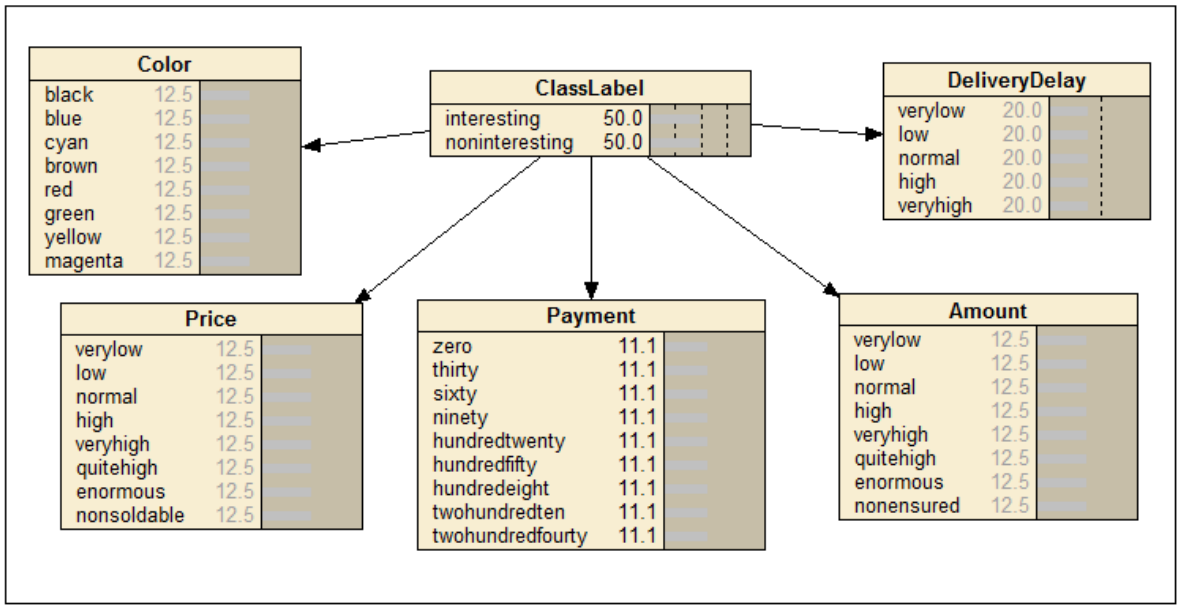

Figura 4.5: Arquitetura da rede Bayesiana proposta.

4.1.1 Implementação

Os ambientes de testes foram implementados na linguagem de programação Java. O Netica-Java API, uma biblioteca de classes em Java, foi utilizado para a construção, manipulação e aprendizagem de redes Bayesianas. Esta subseção do trabalho objetiva apresentar como as redes Bayesianas foram construídas, como o aprendizado foi realizado e como as inferências probabilísticas foram executadas.

A construção das redes Bayesianas foi realizada por meio de chamadas a procedimentos da API Netica. Uma vez que a rede foi construída, ela foi armazenada em um arquivo para uso posterior. A Figura 4.6 apresenta o código para a construção de uma rede Bayesiana com a estrutura da rede definida na Figura 4.5. O programa apresentado constrói uma rede vazia com *new Net()* na linha 5, nomeia a rede na linha 6 e adiciona os nós da rede com *new Node()* entre as linhas 9 e 15. Neste caso, a estrutura da rede é conhecida e por isso a identificação e as ligações entre os nós *Color*, *Price*, *Payment*, *Amount*, *DeliveryDelay* e *ClassLabel* foram realizadas manualmente. As conexões (dependências) entre os nós são definidas entre as linhas 24 e 28, onde uma chamada *color.addLink*{*classLabel*} faz o nó *ClassLabel* pai do nó *Color* e subsequentemente pai de todos os outros nós. Ao fim, toda a estrutura da rede é salva no arquivo RB1.dne, cuja estrutura é apresentada pela Figura 4.7. Este arquivo é utilizado para todas as atividades relacionadas com a manipulação da rede, como inferência probabilística

e atualização das probabilidades condicionais. No momento da criação não foi definida nenhuma probabilidade, por esse motivo uma distribuição uniforme foi realizada automaticamente.

```
public void criarRB() throws NeticaException{
   Node.setConstructorClass ("norsys.neticaEx.aliases.Node");
   Environ env = new Environ (null);
   Net net = new Net();
   net.setName("RB1");

   //Cria os nós da rede bayesiana
   Node color    = new Node ("Color", "black, blue, cyan, brown, red, green, yellow, magenta",  net);
   Node price    = new Node ("Price", "verylow, low, normal, high, veryhigh, quitehigh, enormous, nonsoldable",
                             net);
   Node payment    = new Node ("Payment", "zero, thirty, sixty, ninety, hundredtwenty, " +
                               "hundredfifty, hundredeight, twohundredten, twohundredfourty",  net);
   Node amount    = new Node ("Amount", "verylow, low, normal, high, veryhigh, quitehigh, enormous, nonensured
                              net);
   Node deliveryDelay    = new Node ("DeliveryDelay", "verylow, low, normal, high, veryhigh",  net);
   Node classLabel       = new Node ("ClassLabel", "interesting, noninteressante",  net);

   //Atribui título para cada nó da rede
   color.setTitle ("Color");
   price.setTitle("Price");
   payment.setTitle("Payment");
   amount.setTitle("Amount");
   deliveryDelay.setTitle("DeliveryDelay");
   classLabel.setTitle("ClassLabel");

   //Cria os links entre os nós
   color.addLink (classLabel);
   price.addLink (classLabel);
   payment.addLink (classLabel);
   amount.addLink (classLabel);
   deliveryDelay.addLink (classLabel);

   //Gera arquivo com a rede Bayesiana
   Streamer stream = new Streamer ("RB1.dne");
   net.write (stream);
}
```

Figura 4.6: Código para construção de uma rede Bayesiana.

```
node ClassLabel {
        kind = NATURE;
        discrete = TRUE;
        chance = CHANCE;
        states = (interesting, noninteresting);
        parents = ();
        probs =
                // interesting  noninteresting
                  (0.5,         0.5);
        numcases = 228;
        title = "ClassLabel";
        whenchanged = 1279337754;
        visual V1 {
                center = (468, 132);
                height = 1;
                };
        };

node Color {
        kind = NATURE;
        discrete = TRUE;
        chance = CHANCE;
        states = (black, blue, cyan, brown, red, green, yellow, magenta);
        parents = (ClassLabel);
        probs =
                // black        blue         cyan         brown        red          green        yellow       magenta
                 ((0.140625,    0.046875,    0.2265625,   0.1484375,   0.1328125,   0.046875,    0.0625,      0.1953125),
                  (0.2017544,   0.0877193,   0.0877193,   0.0877193,   0.122807,    0.1842105,   0.1403509,   0.0877193))
        numcases =
                                 // ClassLabel
                   (128,         // interesting
                    114);        // noninteresting ;
        title = "Color";
        whenchanged = 1274324530;
        visual V1 {
                center = (198, 162);
                height = 2;
                };
        };
```

Figura 4.7: Estrutura de um arquivo de criação de rede Bayesiana.

Após uma rede Bayesiana ser construída, ela pode ser utilizada para a execução

de inferências probabilísticas. Cada valor conhecido de variável é informado como uma evidência e a rede se encarrega de encontrar as crenças para todas as outras variáveis. A Figura 4.8 mostra como as inferências probabilísticas foram implementadas para que fosse possível classificar uma oferta como interessante ou não interessante. Primeiramente, a rede RB1 criada anteriormente é recuperada e criada na memória sem que nenhuma informação seja descartada (linhas 4 - 10). Após a sua compilação, as evidências provenientes da instância de treinamento foram informadas para as variáveis *Color*, *Price*, *Payment*, *Amount* e *DeliveryDelay* (linhas 13 - 17), com o objetivo de obter a probabilidade da instância representar uma oferta interessante (linha 19) ou não interessante (linha 20). A opção que possuir a maior probabilidade é retornada como uma predição local.

```
public String classificar(Instance trainingInstance) throws NeticaException{

    float probInteresting = 0;
    float probNonInteresting = 0;
    Net net = new Net (new Streamer ("RB1.dne"));
    Node color = net.getNode("Color");
    Node price = net.getNode("Price");
    Node payment = net.getNode("Payment");
    Node amount = net.getNode("Amount");
    Node deliveryDelay = net.getNode("DeliveryDelay");
    Node classLabel = net.getNode("ClassLabel");
    net.compile();

    //Entra com os findings
    color.finding().enterState (trainingInstance.getColor());
    price.finding().enterState (trainingInstance.getPrice());
    payment.finding().enterState (trainingInstance.getPayment());
    amount.finding().enterState (trainingInstance.getAmount());
    deliveryDelay.finding().enterState (trainingInstance.getDeliveryDelay());

    //Obtém crença
    probInteresting = classLabel.getBelief(Constant.INTERESSANTE);
    probNonInteresting = classLabel.getBelief(Constant.NAO_INTERESSANTE);

    //Se as probabilidades são iguais é sinal de que a rede não aprendeu suficientemente e a instancia deve ser
    //incorporada a rede.
    if(probInteresting == probNonInteresting){
         if(trainingInstance.getClassLabel().equals(Constant.INTERESSANTE)){
                 return Constant.NAO_INTERESSANTE;
         }else{
                 return Constant.INTERESSANTE;
         }
    }
    if(probInteresting > probNonInteresting){
         return Constant.INTERESSANTE;
    }else{
```

Figura 4.8: Código para realizar inferências probabilísticas.

A aprendizagem das redes Bayesianas foi realizada toda vez que uma nova evidência (instância de treinamento) foi observada. Toda nova evidência recebida precisou ser armazenada em um arquivo (base de dados), para que pudesse ser recuperada e utilizada posteriormente. Estes dados poderiam ser criados manualmente ou provenientes de uma base de dados; para os experimentos realizados, eles foram gerados aleatoriamente a partir da estrutura de rede definida e do conceito em questão. A Figura 4.9 apresenta um conjunto de dados de exemplo que representa a descrição do conceito da rede RB1. O cabeçalho do arquivo representa as variáveis e as linhas subsequentes representam os va-

lores dessas variáveis para cada caso. Este arquivo foi nomeado RB1.cas e foi gerado com base na descrição conceitual para *drift* abrupto - conceito 1 (Tabela 3.2).

Color	Price	Payment	Amount	DeliveryDelay	ClassLabel
cyan	verylow	thirty	veryhigh	verylow	noninteresting
magenta	normal	thirty	high	normal	interesting
green	normal	hundredfifty	quitehigh	low	noninteresting
brown	verylow	hundredeighty	low	high	interesting
blue	veryhigh	sixty	enormous	high	noninteresting
cyan	normal	hundredfifty	high	veryhigh	interesting
red	quitehigh	hundredeighty	quitehigh	normal	noninteresting
black	enormous	twohundredten	enormous	veryhigh	noninteresting
magenta	veryhigh	twohundredfourty	verylow	high	noninteresting
brown	high	thirty	nonensured	low	noninteresting
cyan	normal	hundredtwenty	low	high	noninteresting
blue	normal	twohundredfourty	high	verylow	noninteresting
brown	verylow	ninety	verylow	high	interesting
brown	low	hundredeighty	low	normal	noninteresting
cyan	normal	thirty	high	verylow	interesting

Figura 4.9: Dados gerados aleatoriamente para o aprendizado da rede Bayesiana.

A Figura 4.10 mostra como foi realizada a aprendizagem das tabelas de probabilidades condicionais a partir de um grupo de dados, como o RB1.cas. Este programa realiza a leitura da rede criada anteriormente, remove as probabilidades que possam existir em qualquer nó (linhas 11-14) e com base no grupo de casos, realiza a atualização das tabelas de probabilidades condicionais (linha 19). Por fim, a rede atualizada é armazenada em um arquivo para que possa ser recuperada posteriormente e utilizada para outras inferências e aprendizagens. A Figura 4.11 apresenta a visualização da distribuição de probabilidades da rede RB1 após a aprendizagem com os casos da Figura 4.9.

```
public void atualizarCPTs(Instance trainingInstance) throws NeticaException{
   //Atualiza a descrição do conceito com a nova instância de treinamento
   updateCD(trainingInstance);

  //Recupera a rede criada - RB1
   Net net = new Net(new Streamer ("RB1.dne"));
   NodeList nodes = net.getNodes();
   int numNodes = nodes.size();

  //Remove as tabelas de probabilidades condicionais existentes
   for (int n = 0; n < numNodes; n++){
      Node node = (Node) nodes.get(n);
      node.deleteTables();
   }


   //Le o arquivo de casos gerado aleatoriamente
   Streamer caseFile = new Streamer ("RB1.cas");
   //Revisa as tabelas de probabilidades condicionais
   net.reviseCPTsByCaseFile (caseFile, nodes, 1.0);
   net.write (new Streamer ("RB1.dne"));
}
```

Figura 4.10: Código para atualização das probabilidades condicionais.

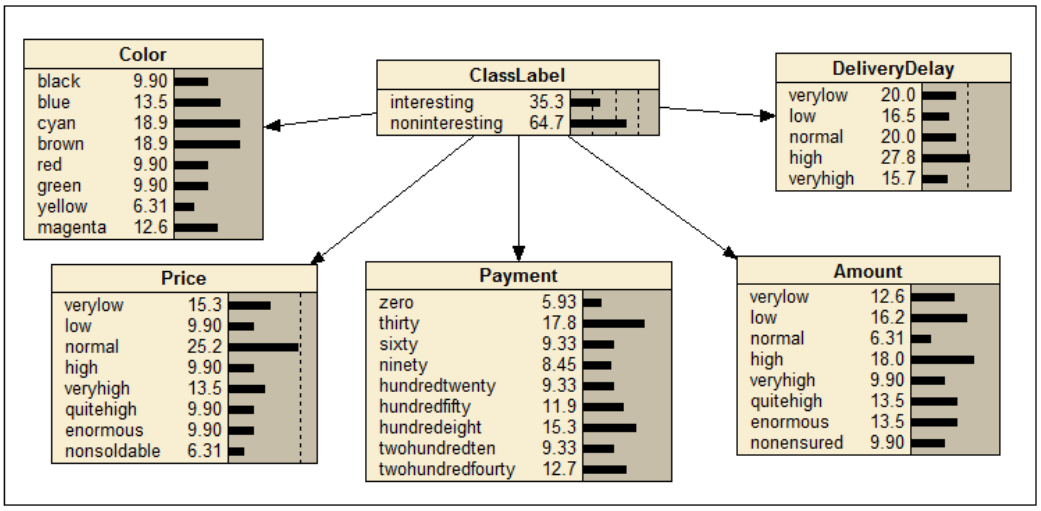

Figura 4.11: Visualização da rede Bayesiana após aprendizagem.

As tabelas de probabilidades condicionais dos nós *Color*, *Price*, *Payment*, *Amount*, *DeliveryDelay* e *ClassLabel* são apresentadas a seguir. Os seus valores são atualizados em cada processo de aprendizagem, buscando refletir a descrição do conceito em questão.

ClassLabel	black	blue	cyan	brown	red	green	yellow	magenta
interesting	7.692	7.692	23.077	23.077	7.692	7.692	7.692	15.385
noninteresting	11.111	16.667	16.667	16.667	11.111	11.111	5.556	11.111

Figura 4.12: Tabela de probabilidades condicionais do nó *Color*

ClassLabel	verylow	low	normal	high	veryhigh	quitehigh	enormous	nonsoldable
interesting	23.077	7.692	30.769	7.692	7.692	7.692	7.692	7.692
noninteresting	11.111	11.111	22.222	11.111	16.667	11.111	11.111	5.556

Figura 4.13: Tabela de probabilidades condicionais do nó *Price*

ClassLabel	zero	thirty	sixty	ninety	hundredtwenty	hundredfifty...	twohundredfourty
interesting	7.143	21.429	7.143	14.286	7.143	14.286	7.143
noninteresting	5.263	15.789	10.526	5.263	10.526	10.526	15.789

Figura 4.14: Tabela de probabilidades condicionais do nó *Payment*

ClassLabel	verylow	low	normal	high	veryhigh	quitehigh	enormous	nonensur...
interesting	15.385	15.385	7.692	30.769	7.692	7.692	7.692	7.692
noninteresting	11.111	16.667	5.556	11.111	11.111	16.667	16.667	11.111

Figura 4.15: Tabela de probabilidades condicionais do nó *Amount*

ClassLabel	verylow	low	normal	high	veryhigh
interesting	20.000	10.000	20.000	30.000	20.000
noninteresting	20.000	20.000	20.000	26.667	13.333

Figura 4.16: Tabela de probabilidades condicionais do nó *DeliveryDelay*

interesting	noninteresting
35.294	64.706

Figura 4.17: Tabela de probabilidades condicionais do nó *ClassLabel*

4.2 Experimentos

Os experimentos foram realizados por meio da implementação das configurações DWM-IB3 e DWM-RB descritas anteriormente. Cada configuração foi testada para as mudanças de conceito moderada, gradual e abrupta. Os resultados apresentados nesta seção foram gerados a partir de uma média de 10 execuções de cada configuração.

Uma das características mais notáveis é a diferença na quantidade de especialistas gerados e mantidos por cada configuração. A Figura 4.18 apresenta a quantidade de especialistas gerados quando a mudança de conceito aconteceu de forma abrupta. O algoritmo DWM alcançou estabilidade em seu número de especialistas IB3 após o conceito 3, mantendo uma média de 28 especialistas. Na configuração DWM-RB, o número continuou crescente até o inicio do conceito 5, alcançando a marca de 46 especialistas na instância 207 e então iniciando a redução desse número, finalizando com 38 especialistas.

Para o *concept drift* moderado, ambas as configurações mantiveram uma média similar de classificadores até a instância 151, alcançando a quantidade de 17 classificadores nesse momento. Deste ponto em diante, o DWM-RB apresentou uma quantidade maior de classificadores, enquanto o IB3 tendeu para a estabilização. A configuração DWM-RB finalizou com 39 classificadores e a configuração DWM-IB3 concluiu com 18 classificadores. (Figura 4.19)

Quando a mudança de conceito aconteceu gradualmente, o número de classifi-

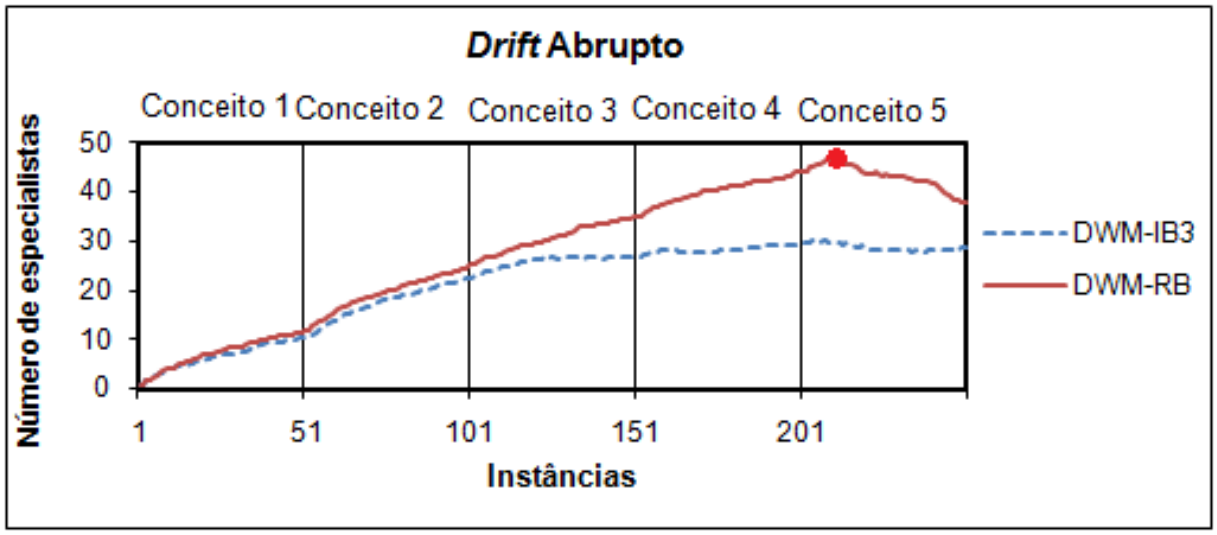

Figura 4.18: Número de especialistas com *concept drift* abrupto.

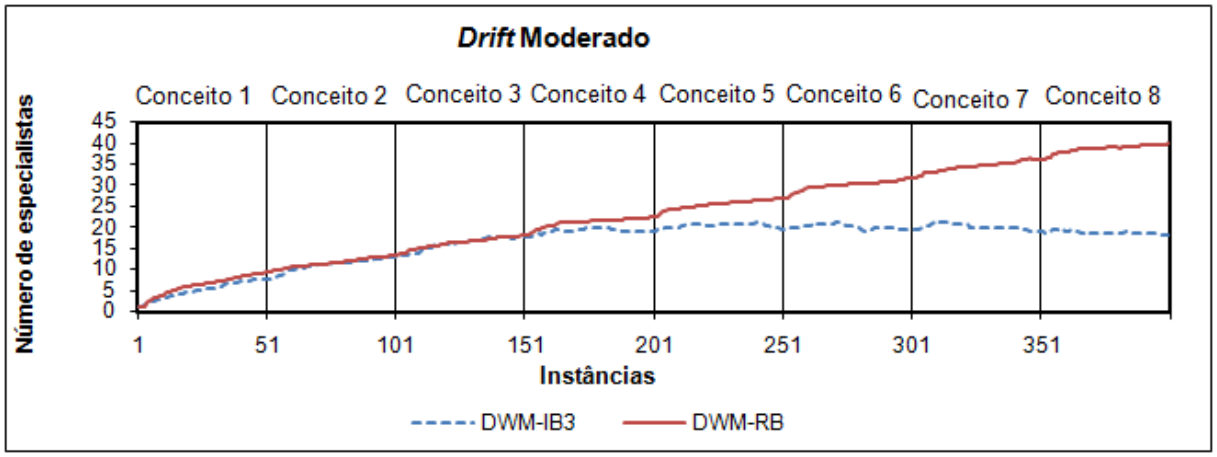

Figura 4.19: Número de especialistas com *concept drift* moderado.

cadores apresentou crescimento contínuo para a quantidade de dados gerados nos dois sistemas, tendendo para o início de uma estabilização no conceito B. DWM-RB concluiu com 27 classificadores, enquanto DWM-IB3 finalizou com 22. (Figura 4.20)

O número de especialistas gerados está relacionado ao tamanho e distribuição do grupo de dados. Em geral, espera-se que o número de especialistas tenda para a estabilização / redução após um valor máximo; isto deve-se a capacidade que o algoritmo DWM tem de utilizar a diversidade dos especialistas para cobrir diferentes regiões do espaço. Quando um especialista representa uma região já coberta por outro, torna-se sobressalente; permitindo que especialistas sejam removidos do grupo. A estabilização no número de especialistas é mais visível na configuração DWM-IB3. De um modo geral, um grupo de especialistas IB3 é capaz de cobrir uma região com menos esforço que um grupo de especialistas RB. Os fatores que podem contribuir para isso são o fato de o IB3 trabalhar com a técnica do vizinho mais próximo e de utilizar um grupo selecionado de

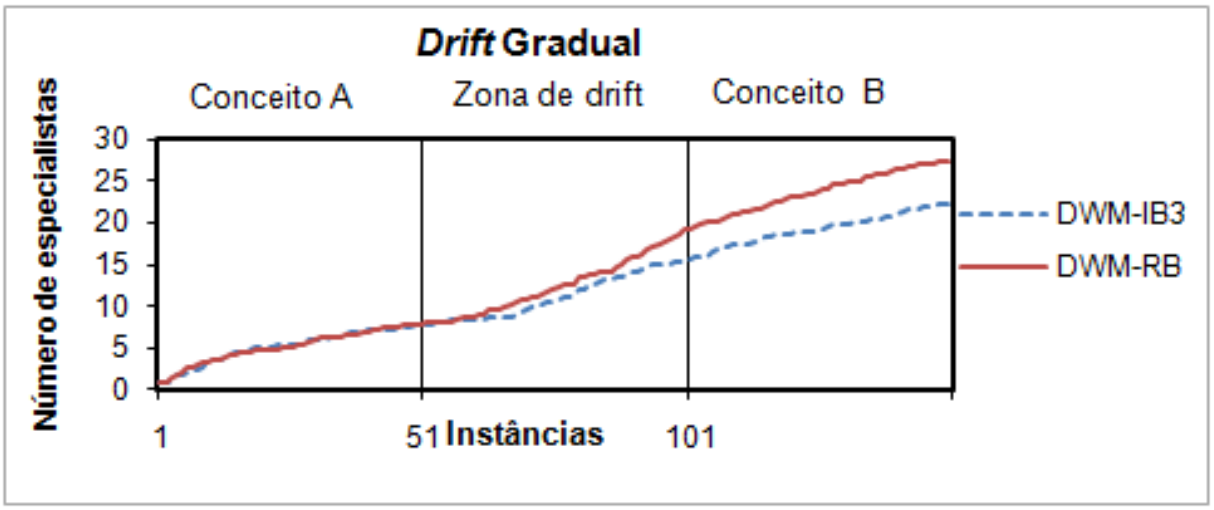

Figura 4.20: Número de especialistas com *concept drift* gradual.

instâncias para realizar as classificações, ou seja, o IB3 trabalha com grupos seletos de instâncias que conseguem rapidamente se adequar e representar uma região enquanto a rede Bayesiana trabalha com todas as instâncias do seu grupo e precisa de mais tempo para conseguir representar uma região específica do espaço de dados.

Em relação a acurácia quando a mudança de conceito aconteceu de forma abrupta, os experimentos geraram resultados bastante similares para os dois sistemas. A Figura 4.21 mostra que os sistemas mantiveram as linhas que descrevem os seus desempenhos muito similares, 80% na média. As quedas de desempenho nos momentos de *drift* foram significativas para os dois sistemas, conforme apresentado pela Figura 4.22. Esta figura apresenta a variação de desempenho que ocorre nos momentos de *drift*, identificando a acurácia no inicio e fim de cada conceito. As configurações DWM-RB e DWM-IB3 alcançaram a sua melhor acurácia no Conceito 5, mas também foi nesse momento que ambas apresentaram a maior queda de desempenho. Neste caso, o DWM-RB possuia uma acurácia de 90,4% ao fim do Conceito 4 e logo após o *concept drift* reduziu para uma acurácia de 39,8%. O DWM-IB3 apresentou a mesma acurácia de 90,4% quando o Conceito 4 finalizou, caindo para o valor de 42% no inicio do Conceito 5. Na Figura 4.23 é possível identificar a velocidade (número de interações) de recuperação de desempenho dos sistemas. Na mudança do Conceito 1 para o Conceito 2, o DWM-RB recuperou o seu desempenho em 38 interações, enquanto que o DWM-IB3 não foi capaz de atingir no segundo conceito, o mesmo desempenho que possuia no Conceito 1. Em geral, o DWM-IB3 foi capaz de se recuperar das quedas de desempenho mais rapidamente que o DWM-RB, levando apenas 10 interações na maioria das vezes. Isto deve-se ao fato de a rede Bayesiana trabalhar com probabilidades, com um número maior de instâncias para realizar as classificações, precisando de mais tempo para atualizar o seu modelo.

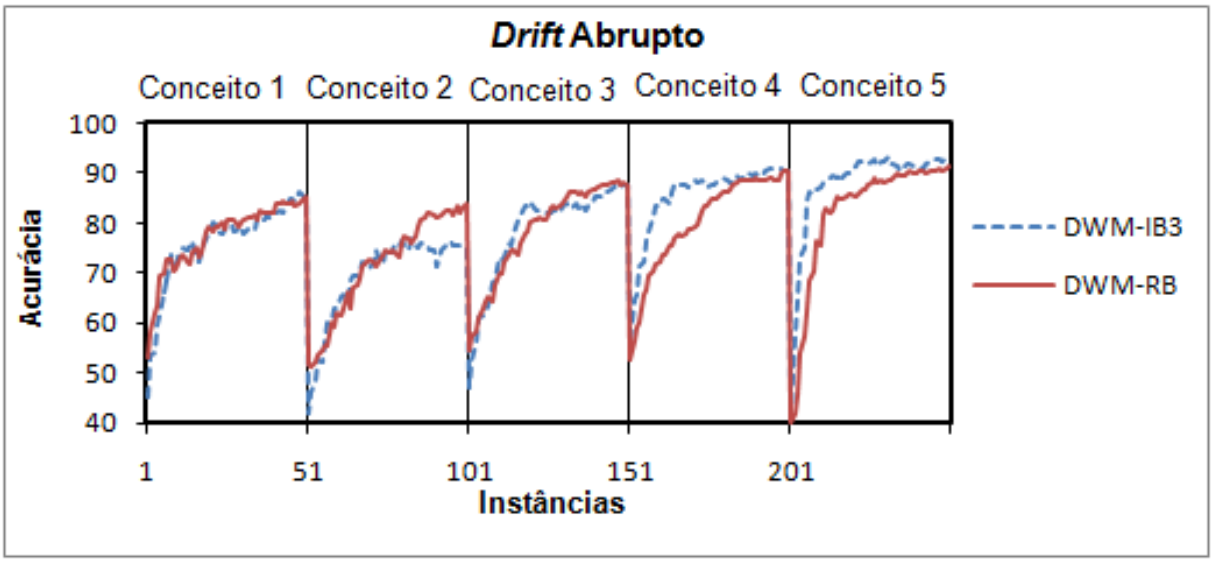

Figura 4.21: Desempenho com *concept drift* abrupto.

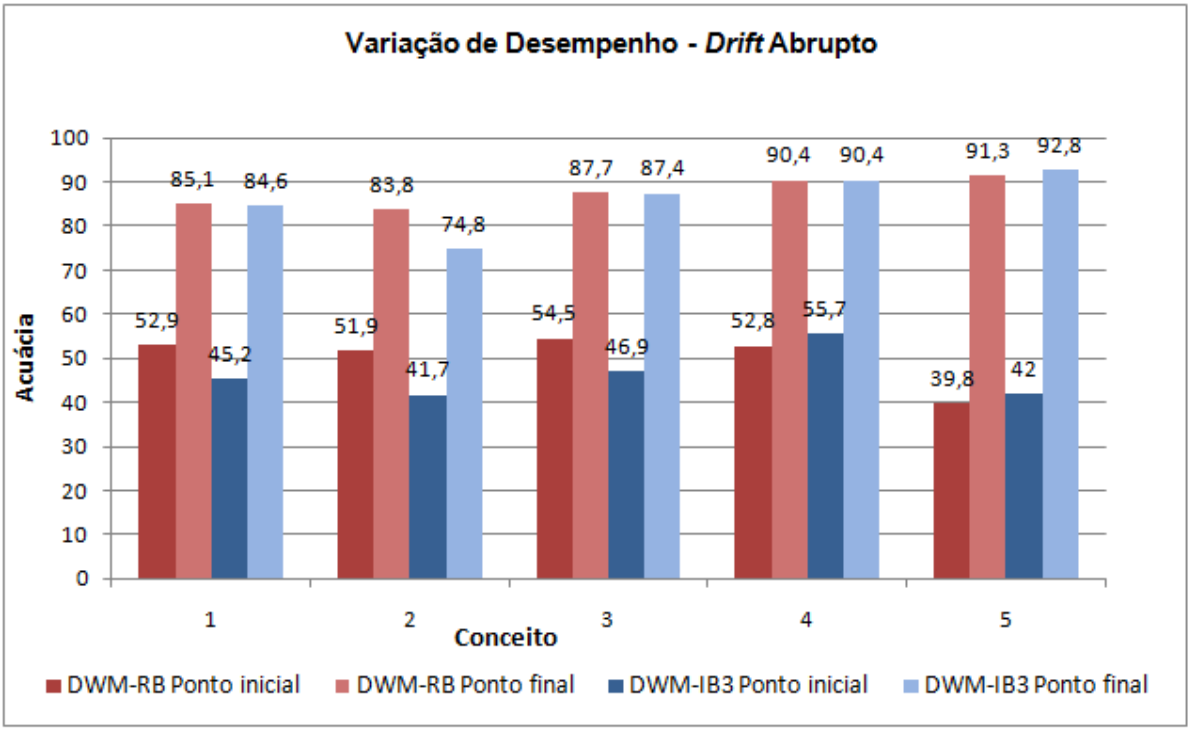

Figura 4.22: Variação de desempenho com *concept drift* abrupto.

Os resultados obtidos a partir dos dados com *drift* moderado apresentaram o desempenho do DWM-RB sobressaindo-se ligeiramente em relação ao desempenho do DWM-IB3. Este comportamento pode ser explicado pelo fato das atualizações ocorrerem, por natureza, mais lentamente na rede Bayesiana combinada com as alterações mais suaves de *drifts* moderados. O primeiro obteve uma acurácia média de 89% e o segundo de 84%. Apesar de em alguns momentos apresentar uma maior queda de desempenho, enfatizado

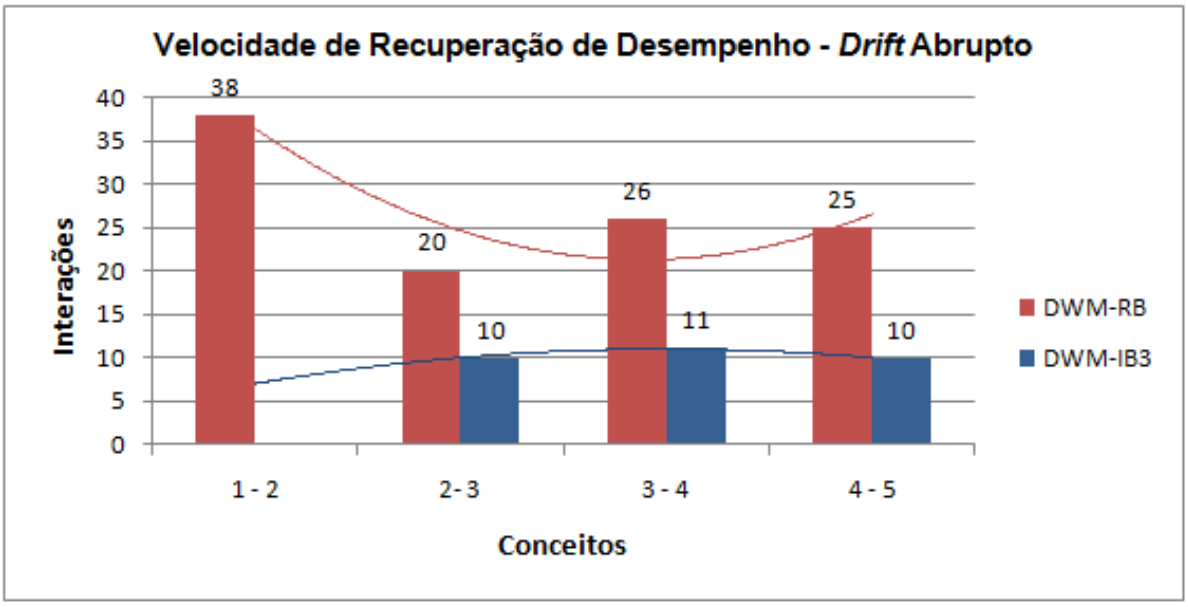

Figura 4.23: Tendência de recuperação de desempenho em *concept drift* abrupto.

pela Figura 4.25, o DWM-RB conseguiu recuperar o seu desempenho mais rapidamente que o DWM-IB3, conforme mostra a Figura 4.26. Neste caso, o IB3 reagiu de forma mais suave quando comparado as alterações abruptas, uma vez que um desempenho razoável foi alcançado, as instâncias armazenadas apresentaram uma boa acurácia na classificação e não foram alteradas com tanta frequência.

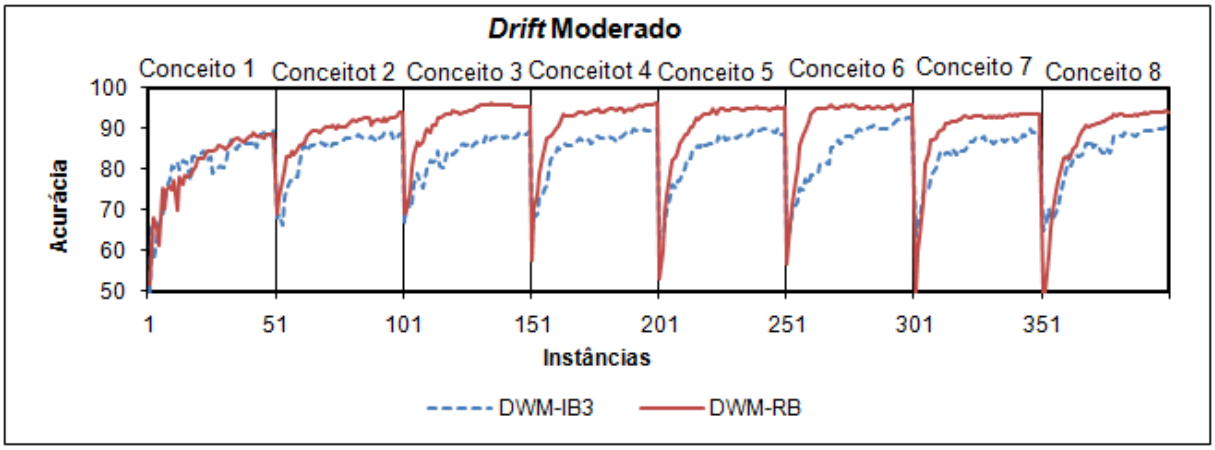

Figura 4.24: Desempenho com *concept drift* moderado.

A Figura 4.27 mostra que na mudança de conceito de forma gradual, o DWM-IB3 e o DWM-RB mantiveram um desempenho médio de 80%. A queda de desempenho de ambas as configurações foi muito similar e bem menor que as identificadas pelas Figuras 4.21 e 4.24, devido a similaridade entre os conceitos produzidos.

Os ambientes menos afetados pelas mudanças e que conseguiram manter um bom

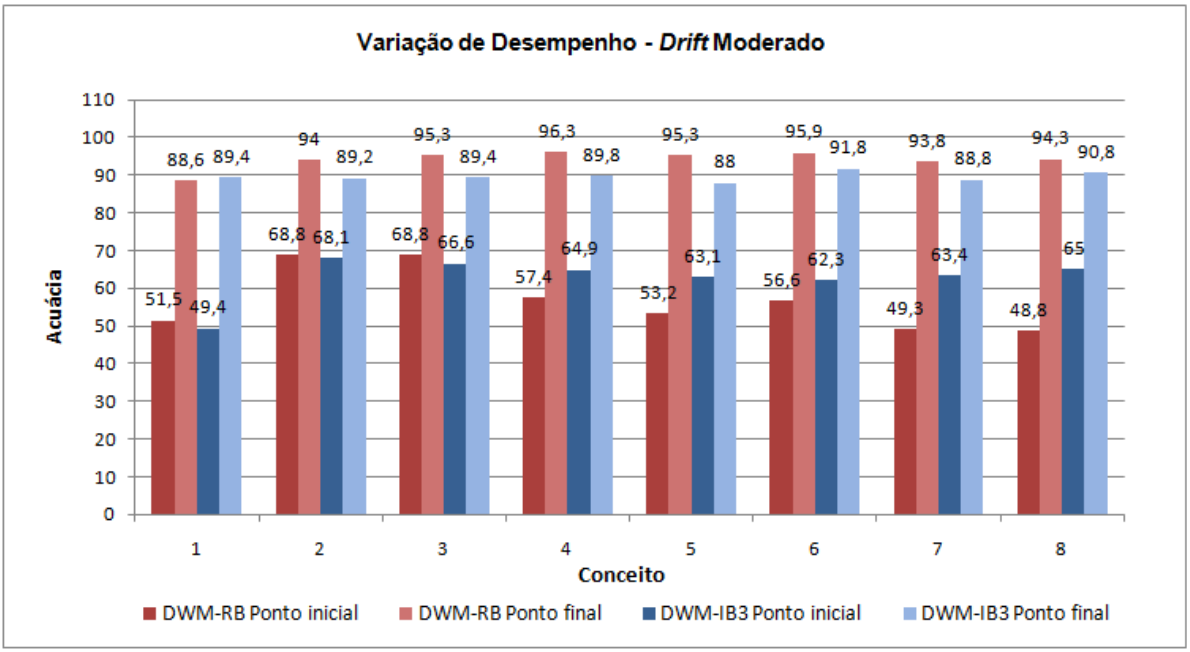

Figura 4.25: Variação de desempenho com *concept drift* moderado.

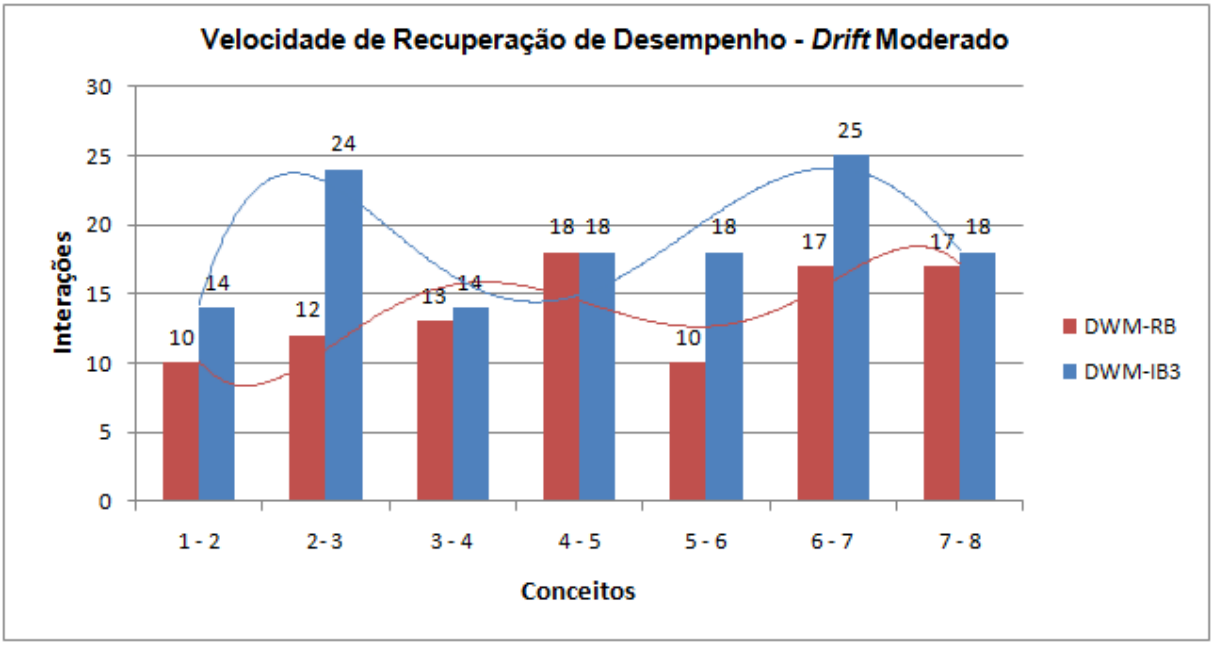

Figura 4.26: Tendência de recuperação de desempenho em *concept drift* moderado.

desempenho com quedas menos bruscas foram os que alteraram o conceito gradualmente e moderadamente, devido a similaridade entre os conceitos. Os *drifts* moderados e graduais também geraram o menor número de especialistas, uma vez que eles são, teoricamente, o tipo de *drift* mais fácil de aprender. Também são nesses cenários que o DWM detecta *drifts* mais lentamente, uma vez que o grupo de especialistas é alterado suavemente, os modelos levam um período maior para refletirem o novo conceito. Pode-se observar nos

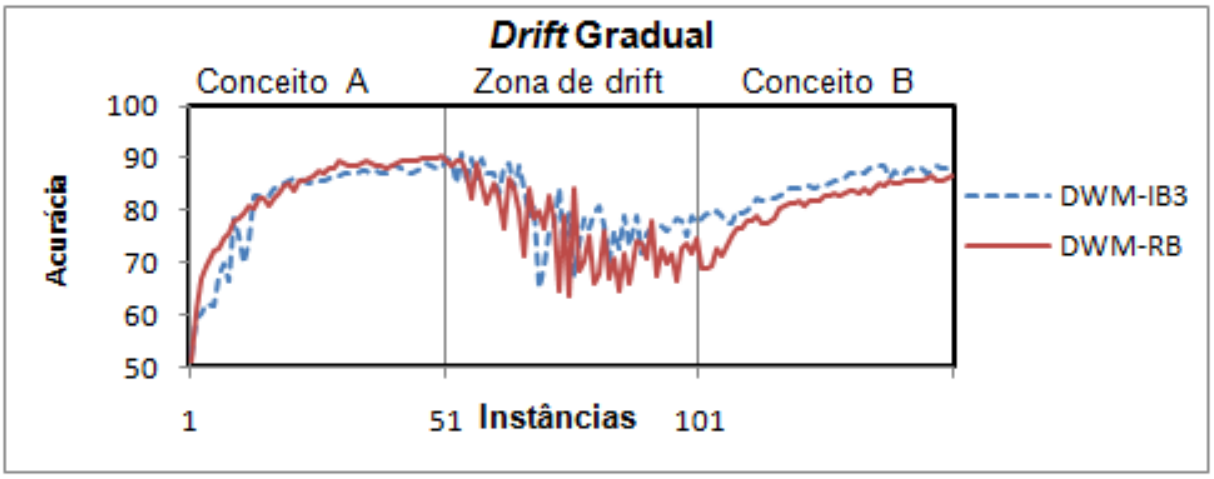

Figura 4.27: Desempenho com *concept drift* gradual.

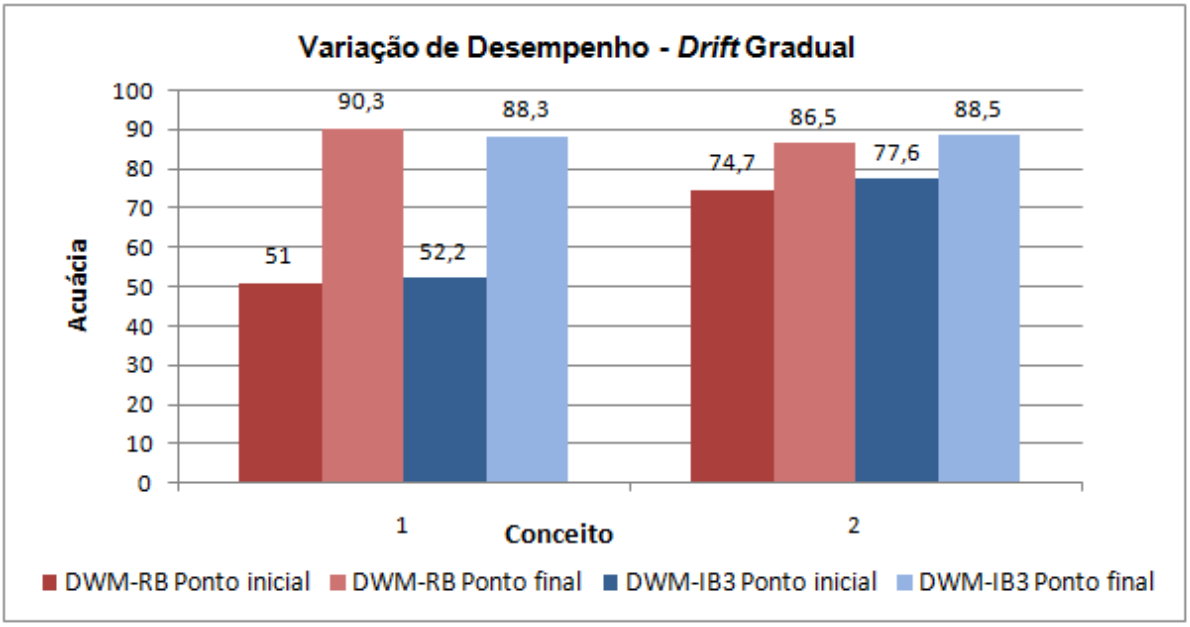

Figura 4.28: Variação de desempenho com *concept drift* gradual.

gráficos, que as linhas que apresentam a acurácia dos sistemas são mais suaves nos cenários em que os $drifts$ são moderados ou graduais. Uma vez que esses conceitos são mais fáceis de serem aprendidos e as quedas de desempenho não são muito bruscas, as instâncias armazenadas possuem um bom desempenho e não são removidas com tanta frequência, tornando o algoritmo IB3 mais conservador. O mesmo aplica-se para a rede Bayesiana, que devido a maior similaridade entre os conceitos, executa alterações no seu modelo que interferem mais lentamente na mudança das probabilidades.

Outro fator que deve ser considerado é a diferença no tempo de execução entre as configurações. O maior tempo consumido pela configuração DWM-RB (em torno de 60 vezes maior que DWM-IB3) ocorreu devido a dois fatores: necessidade de alterar as

probabilidades condicionais de cada rede Bayesiana toda vez que uma nova instância é recebida e a maior quantidade de instâncias e de especialistas mantidos por essa configuração durante todo o processo.

O modelo do algoritmo de aprendizagem IB3 tende a se adaptar mais rapidamente ao novo conceito devido ao fato de realizar classificações utilizando um grupo específico de instâncias e de adequar o seu grupo por meio da remoção de instâncias com baixo desempenho. As redes Bayesianas, por trabalharem com um contexto probabilístico e não removerem instâncias com desempenho pobre, levam mais tempo para adequar o seu modelo ao conceito em questão, justificando uma estabilização mais tardia do número de especialistas. Apesar dessas pequenas divergências, o algoritmo DWM foi capaz de manter um desempenho satisfatório e similar para ambas as configurações.

4.3 Considerações Finais

O desenvolvimento dos ambientes de negociação descritos neste capítulo permitiram uma análise eficiente do algoritmo de aprendizagem IB3 e da rede Bayesiana, ambos coordenados pelo método DWM. De um modo geral, o desempenho dos sistemas foi muito similar, detectando *concept drifts* rapidamente e recuperando o desempenho em poucas interações. Apesar de as redes Bayesianas levarem um tempo maior para aprenderem os conceitos, é possível concluir que ambas as configurações, DWM-IB3 e DWM-RB, são similarmente eficientes na detecção de *concept drifts*, já que o DWM foi capaz de coordenar com a mesma eficiência técnicas de aprendizagem distintas; entretanto, a configuração DWM-RB requer grandes recursos de memória e um alto tempo de precessamento devido a necessidade de alterar cada rede com toda instância classificada, enquanto que para o DWM-IB3, a atualização é uma atividade de custo baixo. Essas considerações juntamente com os resultados obtidos, encorajam fortemente o uso da configuração DWM-IB3 para a detecção de diferentes tipos de *drifts* em processos de negociação bilateral multicritério.

Capítulo 5

Conclusão

A negociação é um processo não trivial que se caracteriza pela troca de ofertas e contraofertas, podendo utilizar múltiplos critérios. Entretanto, uma das maiores complexidades de um processo de negociação é fornecer aos agentes a habilidade de aprender as preferências do seu oponente e além disso, dotá-los com a capacidade de detectar e se adaptar automaticamente as mudanças de preferências que costumam acontecer com o decorrer do tempo. Os avanços tecnológicos permitem a construção de ambientes de negociação automatizados que refletem essas características.

O objetivo deste trabalho foi apresentar um sistema de aprendizagem baseado em agentes para a detecção de *drift* em um processo de negociação bilateral, simulando alterações nas preferências de uma das partes com o passar do tempo para que o oponente detecte e se adapte a essas mudanças automaticamente. Os experimentos realizados com as duas configurações desenvolvidas visaram apresentar resultados que comprovassem a eficiência de ambas na técnica de *drift detection* e permitiram também obter as seguintes conclusões:

- O algoritmo IB3, baseado em instâncias, é eficiente na redução da demanda de armazenamento e realiza a atualização de seu modelo a um custo baixo;
- A rede Bayesiana é uma estrutura de visualização e entendimento fácil, que permite a fácil integração de conhecimento *a priori*; no entanto, realiza a atualização de seu modelo a um custo alto;
- O uso do método de conjunto DWM foi capaz de manter um bom desempenho para ambas as configurações, pois ele faz uso de uma grande quantidade de informação (grupo de especialistas) para auxiliar nas tomadas de decisão, sendo menos suscetível a erros que podem ser cometidos por um especialista isolado;

- O método de conjunto DWM pode ser utilizado, com êxito, em processos de negociação bilateral para a detecção de $drifts$.
- O DWM foi capaz de coordenar, com a mesma eficiência, técnicas de aprendizagem distintas;
- Ambas as configurações desenvolvidas neste trabalho (DWM-IB3 e DWM-RB) apresentaram desempenhos satisfatórios na detecção de $drift$ em um processo de negociação bilateral.

Os resultados alcançados mostraram que os objetivos deste trabalho foram atingidos: detectar *drifts* em um processo de negociação bilateral utilizando rede Bayesiana e IB3 coordenados pelo mesmo método de grupo (DWM), e contribuir com o estudo de duas técnicas, cada uma com suas particularidades, mas ambas eficientes no processo de aprendizagem.

5.1 Trabalhos Futuros

Trabalhos serão desenvolvidos visando estudar com mais detalhes uma das configurações aqui apresentadas; trata-se do *Dynamic Weighted Majority* em conjunto com a rede Bayesiana. O estímulo para este estudo surgiu do interesse em testar diferentes parâmetos e estruturas para esta configuração, visando contornar a sua limitação de altos custos de processamento e armazenamento apresentados pelos experimentos atuais. Testes serão realizados por meio de alterações aos parâmetros do algoritmo DWM e pela escolha dos melhores momentos em que as redes passarão pelo processo de aprendizagem, visando primeiramente diminuir a quantidade de especialistas do grupo por meio de um número maior de remoções e então reaprender somente quando necessário. As redes Bayesianas também serão testadas com o método de conjunto AddExp, visando avaliar se os métodos de remoção propostos são capazes de reduzir a quantidade de especialistas do grupo. Espera-se encontrar soluções viáveis, que diminuam os custos e que mantenham ou melhorem o desempenho de detecção de *drift* aqui apresentado.

Referências Bibliográficas

Aha, D. W.; Kibler, D.; Albert, M. K.: Instance-based Learning Algorithms. Machine Learning, n. 6, v. 1, pp. 37-66, 1991.

Botelho, V.; Enembreck, F.; Avila, B.; Azevedo, H.; Scalabrin, E.: Using asymmetric keys in a certified trust model for multiagent systems, Expert Systems with Applications. Volume 38. Issue 2. Intelligent Collaboration and Design. p. 1233-1240, 2011.

Coehoorn, R. M.; Jennings, N. R.: Learning an Opponent's Preferences to Make Effective Multi-Issue Negotiation Trade-Offs. Proc. of the Sixth International Conference on Electronic Commerce, pp. 59-68, 2004.

Enembreck, F.; Avila, B. C.; Scalabrin, E. E.; Barthès, J-P.: Drifting Negotiations. Applied Artificial Intelligence, Taylor & Francis, pp. 861-881, v. 21, n. 9, 2007.

Enembreck, F.; Tacla, C. A.; Barthès, J. P.: Learning Negotiation Policies Using Ensemble-Based Drift Detection Techniques. In: International Journal of Artificial Intelligence Tools, pp. 173-196, v. 18, n. 2, 2008.

Faratin, P.; Sierra, C.; Jennings R. N.: Negotiation decision functions for autonomous agents. International Journal of Robotics and Autonomous Systems, 24: p. 3-4, 1998.

Friedman, N.; Geiger, D.; Goldszmidt, M.: Bayesian network classifiers. Machine Learning, v. 29, n. 2-3, p. 131-163, 1997.

Hart P. E.: The condensed nearest neighbor rule, IEEE Transactions of Information Theory (corresp.) 515-516, 1968.

Hindriks, K.; Tykhonov, D.: Opponent modelling in automated multi-issue negotiation using bayesian learning. In: AAMAS'08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems, pp. 331-338, 2008.

Jonker, C.; Robu, V.; Treur, J.: An Agent Architecture for Multi-Attribute Negotiation Using Incomplete Preference Information. In: Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS), Springer-Verlag, 2006.

Kersten, G. E.; Michalowski, W.; Szpakowicz, S.; Koperczak, Z.: Restructurable representations of negotiation. Manage. Sc., pp. 1269-1290, v. 37, n. 10, 1991.

Kersten, G. E.; Noronha, S. J.: WWW-based negotiation support: design, implementation and use. Decision Support Systems, v. 25, n. 2, pp. 135-154, 1999.

Kolter, J. Z.; Maloof, M. A.: Dynamic Weighted Majority: A New Ensemble Method for Tracking Concept Drift. IEEE Computer Society, pp. 123, 2003.

Kolter, J. Z.; Maloof, M. A.: Dynamic Weighted Majority: An Ensemble Method for Drifting Concepts. The Journal of Machine Learning Research , pp. 2755 - 2790, 2007.

Kolter, J. Z.; Maloof, M. A.: Using Additive Expert Ensembles to Cope with Concept Drift. In Proceedings of the 22nd International Conference on Machine Learning , pp. 449 - 456, 2005.

Lin, R.; Kraus, S.; Wilkenfeld, J.; Barry, J.: An Automated Agent for Bilateral Negotiation with Bounded Rational Agents with Incomplete Information. ECAI, pp. 270-274, 2006.

Littlestone, N.; Warmuth, M.: The Weighted Majority algorithm. Information and Computation, 108, pp. 212-261, 1994.

Menke, J. e Martinez, T.: Improving Supervised Learning by Adapting the Problem to the Learner. International Journal of Neural Systems, 19:1, pp. 1-9, 2009.

Murphy, K.: A brief introduction to graphical models and Bayesian networks. http://www.cs.ubc.ca/m̃urphyk/Bayes/bnintro.html, 1998.

Norsys Software Corp, Netica-j manual. Web Site, http://www.norsys.com/neticaj/docs/netica#jman.pdf, 2009.

Opitz, D.; Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11, pp. 169-198, 1999.

Pearl, J.: Probabilistic reasoning in intelligent systems: Networks of plausible inference. Morgan Kaufmann, San Mateo, CA, 1988.

Pereira, R. F.; Banaszewski F, R; Simão M. J.; Tacla A. C.: Método Baseado em Detecção de Mudanças para Determinar Preço de Oferta de Pedidos de Clientes no Ambiente TAC-SCM. II Mostra de Pesquisa e Pós-Graduação da UTFPR, 2010.

Pruitt, D. G.: Negotiation behavior. New York: Academic Press, 1981.

Quinlan, J. R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Mateo, CA, 1993.

Raiffa, H.: The Art & Science of Negotiation. Harvard University Press, Cambridge, Massachusetts, 1982.

Romanhuki, E.: Aprendizagem de Políticas de Oferta de Negociacão entre Agentes Cognitivos. Dissertação de Mestrado, Pontifícia Universidade Católica do Paraná, 2008.

Ruggeri, F.; Faltin, F.; Kenett, R.: Bayesian Networks. Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons, 2007.

Russell, S.; Norving, P.: Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs, NJ, 2nd edition, 2004.

Spirtes, P.; Glymour, C.; Scheines, R.: Causation, Prediction, and Search. Springer-Verlag, New York, 1993.

Stanley, K. O.: Learning Concept Drift with a Committee of Decision Trees, Department of Computer Sciences, University of Texas at Austin, Technical Report AI-03-302, September, 2003.

Street, W.; Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In Proceedings of the 7th ACMInternational Conference on Knowledge Discovery and Data Mining, p. 377-382, ACM Press, New York, NY, 2001.

Wang, H.; Fan, W.; Yu, S. P.; Han, J.: Mining concept-drifting data streams using ensemble classifiers. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, p. 226-235. ACM Press, New York, NY, 2003.

Wersing, H.; Kirstein, S.; Götting, M.; Brandl, H.; Dunn, M.; Mikhailova, I.; Goerick, C.; Steil, J.; Ritter, H.; Körner, E.: Online Learning of Objects in a Biologically Motivated Visual Architecture. International Journal of Neural Systems, 17(4), pp. 219-230, 2007.

Zeng, D.; Sycara, K.: Bayesian learning in negotiation. International Journal of Human-Computer Studies, n. 48, v. 1, pp. 125-141, 1998.

Zheng, Z.: Naive Bayesian Classifier Committees. In: Proceedings of the 10th European Conference on Machine Learning, Springet-Verlag, p. 196-207, 1998.

Apêndice A

Construção e Manipulação de rede Bayesiana utilizando a API-Netica

A.1 Caso Alarme

A Figura A.1 apresenta um exemplo de código para construção de uma rede Bayesiana com a mesma estrutura da rede do caso alarme (Figura 3.6). A rede *CasoAlarme* é criada nas linhas 4 e 5, os seus nós são definidos entre as linhas 8 e 12, e as conexões entre os nós são definidas entre as linhas 22 e 25. Ao fim, a rede é salva no arquivo CasoAlarme.dne para que possa ser recuperada quando necessário.

A Figura A.2 apresenta um conjunto de dados gerado aleatoriamente a partir da estrutura da rede criada. Esses dados visam simular um conjunto de informações históricas que podem ser utilizados para o aprendizado automático da rede Bayesiana. Neste caso, esses dados são utilizados pelo código da Figura A.3 para a aprendizagem das tabelas de probabilidades condicionais. O código apresentado realiza a leitura da rede criada anteriormente, remove as probabilidades que possam existir em qualquer nó (linhas 7-10) e com base nos casos gerados (linha 13), realiza a atualização das tabelas de probabilidades condicionais (linha 15). Por fim, salva a rede atualizada no arquivo CasoAlarme.dne.

A Figura A.4 mostra como é possível realizar inferências probabilísticas. Após realizar a leitura da rede e de seus nós (linha 1 - 9), obtém-se as probabilidades *a priori*, ou seja, as probabilidades antes que a rede seja alimentada com qualquer evidência. Dado o aprendizado que foi realizado com os dados da Figura A.2, os valores *a priori* obtidos aqui são:

- $P(J) = 0.44$: 44% de probabilidade de João ligar;
- $P(M) = 0.49$: 49% de probabilidade de Maria ligar;

```
    private void construirRedeBayesiana() throws NeticaException {
1           Node.setConstructorClass ("norsys.neticaEx.aliases.Node");
2           Environ env = new Environ (null);
3
4           Net net = new Net();
5           net.setName("CasoAlarme");
6
7           //Cria nós da rede Bayesiana
8           Node roubo     = new Node ("Roubo",   "verdadeiro, falso",  net);
9           Node terremoto = new Node ("Terremoto", "verdadeiro, falso",  net);
10          Node alarme = new Node ("Alarme","verdadeiro, falso", net);
11          Node joaoLiga       = new Node ("JoaoLiga", "verdadeiro, falso",  net);
12          Node mariaLiga       = new Node ("MariaLiga", "verdadeiro, falso",  net);
13
14          //Atribui título para cada nó da rede
15          roubo.setTitle ("Roubo");
16          terremoto.setTitle ("Terremoto");
17          alarme.setTitle ("Alarme");
18          joaoLiga.setTitle("JoaoLiga");
19          mariaLiga.setTitle("MariaLiga");
20
21          //Cria os links entre os nós
22          alarme.addLink (roubo);
23          alarme.addLink (terremoto);
24          joaoLiga.addLink(alarme);
25          mariaLiga.addLink(alarme);
26
27          //Gera arquivo com a rede Bayesiana
28          Streamer stream = new Streamer ("CasoAlarme.dne");
29          net.write (stream);
30
31          net.finalize();
    }
```

Figura A.1: Código para construção de uma rede Bayesiana.

IDnum	Roubo	Terremoto	Alarme	JoaoLiga	mariaLiga
0	verdadeiro	verdadeiro	verdadeiro	falso	falso
1	falso	verdadeiro	falso	verdadeiro	falso
2	verdadeiro	falso	falso	falso	verdadeiro
3	verdadeiro	verdadeiro	falso	verdadeiro	falso
4	falso	verdadeiro	verdadeiro	falso	falso
5	verdadeiro	verdadeiro	verdadeiro	falso	verdadeiro
6	falso	falso	verdadeiro	verdadeiro	verdadeiro
7	verdadeiro	verdadeiro	verdadeiro	falso	verdadeiro
8	verdadeiro	verdadeiro	verdadeiro	falso	verdadeiro
9	falso	falso	falso	verdadeiro	falso

Figura A.2: Dados gerados aleatoriamente para o aprendizado da rede Bayesiana.

- $P(A|J) = 0.32$: 32% de probabilidade de o alarme ter disparado, dado que João ligou;

```
    private void aprenderCPTs() throws NeticaException {
1       // Recupera a rede Bayesiana CasoAlarme
2       Net net = new Net (new Streamer ("CasoAlarme.dne"));
3       NodeList nodes    = net.getNodes();
4
5       int  numNodes = nodes.size();
6       // Remove as tabelas de probabilidades condicionais existentes.
7       for (int n = 0;  n < numNodes;  n++) {
8           Node node = (Node) nodes.get (n);
9           node.deleteTables();
10      }
11
12      // Le o arquivo de casos gerado aleatoriamente
13      Streamer caseFile = new Streamer ("CasoAlarme.cas");
14      //Revisa as tabelas de probabilidades condicionais
15      net.reviseCPTsByCaseFile (caseFile, nodes, 1.0);
16      net.write (new Streamer ("CasoAlarme.dne"));
17      net.finalize();
    }
```

Figura A.3: Código para atualização das probabilidades condicionais.

- $P(A|M) = 0.70$: 70% de probabilidade de o alarme ter disparado quando Maria ligou;
- $P(A) = 0.55$: 55% de probabilidade de o alarme disparar.

Após alimentar a rede com a evidência de que o alarme disparou, as linhas 19 e 20 retornam:

- $P(J|A) = 0.25$: 25% de chance do João ligar quando o alarme dispara;
- $P(M|A) = 0.62$: 62% de Maria telefonar quando o alarme dispara.

Uma situação como esta, permitiria identificar quem é mais eficaz em telefonar para Pedro quando o alarme dispara, João ou Maria. A estrutura da rede gerada pelos códigos descritos aqui é apresentada pela Figura A.5.

As mesmas probabilidades podem ser encontradas utilizando o teorema de Bayes (Equação 3.2). Os dois exemplos de cálculo são demonstrados a seguir, a Equação A.1 calcula a probabilidade de João ligar dado que o alarme disparou e a Equação A.2 calcula a probabilidade de Maria ligar dado que o alarme disparou.

```
    private void executarInferencia() throws NeticaException{
1       Net net = new Net (new Streamer ("CasoAlarme.dne"));
2       //Recupera os nós da rede
3       Node roubo = net.getNode("Roubo");
4       Node terremoto = net.getNode("Terremoto");
5       Node alarme = net.getNode("Alarme");
6       Node joaoLiga = net.getNode("JoaoLiga");
7       Node mariaLiga = net.getNode("mariaLiga");
8
9       net.compile();
10
11      //Probabilidades a priori
12      float probPrioriJoaoLigar = joaoLiga.getBelief("verdadeiro");
13      float probPrioriMariaLigar = mariaLiga.getBelief("verdadeiro");
14
15      //Evidências
16      alarme.finding().enterState ("verdadeiro");
17
18      //Probabilidades a posteriori
19      float probPosterioriJoaoLigar = joaoLiga.getBelief("verdadeiro");
20      float probPosterioriMariaLigar = mariaLiga.getBelief("verdadeiro");
21
    }
```

Figura A.4: Código para realizar inferências probabilísticas.

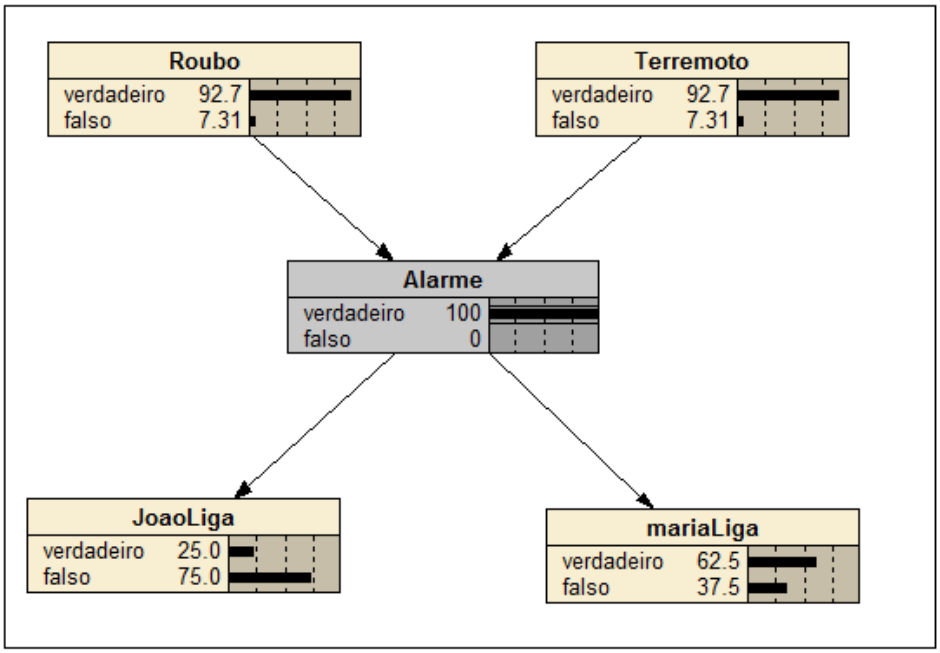

Figura A.5: Rede Bayesiana - Caso Alarme.

$$P(J|A) = \frac{P(A|J)P(J)}{P(A)} = \frac{0.32 \times 0.44}{0.55} = 0.25 = 25\% \tag{A.1}$$

$$P(M|A) = \frac{P(A|M)P(M)}{P(A)} = \frac{0.70 \times 0.49}{0.55} = 0.62 = 62\% \tag{A.2}$$

Neste caso, Maria poderia ser considerada a pessoa mais confiável a ligar para Pedro quando o alarme dispara.