UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

APRENDIZADO SUPERVISIONADO INCREMENTAL DE REDES BAYESIANAS PARA MINERAÇÃO DE DADOS

Murilo Lacerda Yoshida

SÃO CARLOS
AGOSTO/2007

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

Y54as

Yoshida, Murilo Lacerda.
Aprendizado supervisionado incremental de Redes Bayesianas para mineração de dados / Murilo Lacerda Yoshida. -- São Carlos : UFSCar, 2007.
132 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Inteligência artificial. 2. Aprendizado incremental. 3. Data mining (Mineração de dados). 4. Representação do conhecimento. I. Título.

CDD: 006.3 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Aprendizado Supervisionado Incremental de Redes Bayesianas para Mineração de Dados"

MURILO LACERDA YOSHIDA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Membros da Banca:

Prof. Dr. Estevam Rafael Hruschka Junior
(Orientador – DC/UFSCar)

Profa. Dra. Maria do Carmo Nicoletti
(DC/UFSCar)

Prof. Dr. Júlio César Nievola
(PUC/PR)

Profa. Dra. Vilma Alves de Oliveira
(EESC/USP)

São Carlos
Agosto/2007

À minha família,
Sempre presentes,
Mesmo quando ausentes

Agradecimentos

Gostaria de agradecer:

À minha mãe e meu irmão, pelo apoio incondicional;

Ao meu falecido pai, pela motivação;

À minha noiva, por todo o bem que ela me faz;

À minha família,

À família da minha noiva;

Ao meu orientador, Estevam Rafael Hruschka Junior, por toda a ajuda e orientação ao longo desses anos;

Aos meus amigos, por suportarem minha ausência;

À professora Maria do Carmo Nicoletti;

Aos professores e funcionários do Departamento de Computação (DC/UFSCar);

À tia Beth, Dri, Vânia, Jana e João Caetano pela amizade e pelo apoio;

Ao Edimilson Batista dos Santos, por toda a ajuda durante os experimentos e ao longo dos estudos;

Às empresas em que trabalhei durante o período de mestrado:

AGX Tecnologia,

IWS do Brasil,

VoxAge TeleInformática,

Obrigado por todo o suporte e compreensão;

Aos meus colegas de trabalho;

Aos meus colegas de mestrado.

Resumo

Esse trabalho tem como objetivo propor dois algoritmos para aprendizado incremental supervisionado de redes Bayesianas, o AIP (Aprendizado Incremental ingênuo de Parâmetros) e o ABC (Aprendizado Bayesiano em Camadas). Para isso se pesquisou conceitos teóricos de redes Bayesianas, algoritmos de aprendizado de redes Bayesianas e métodos de aprendizado incremental de redes Bayesianas relevantes na literatura. Para melhorar o desempenho do aprendizado incremental se pesquisou uma estrutura de representação de conhecimento chamada AD-Tree. Para aferir a qualidade das redes Bayesianas produzidas se utilizou essas redes para classificar conjuntos de teste, obtendo assim o índice de classificação correta (ICC). Foi pesquisado o processo de classificação de conjuntos de teste e o processo de propagação de evidências. O resultado dessas pesquisas está descrito no trabalho, junto com os resultados e discussões sobre os experimentos feitos com os algoritmos propostos.

Palavras-chave: Inteligência Artificial, Redes Bayesianas, Aprendizado Incremental, Mineração de Dados, Representação do Conhecimento.

Abstract

The objective of this work is to introduce two algorithms for supervised Bayesian network incremental learning, AIP (Algorithm for simple Bayesian network numerical parameters supervised incremental learning) and ABC (Algorithm for Bayesian network supervised incremental learning in layers). In order to develop these algorithms we studied relevant works about the Bayesian networks concepts, the algorithms for supervised Bayesian network learning and the algorithms for incremental supervised Bayesian network learning. To improve the performance of the ABC algorithm, we studied the AD-Tree structure and implemented it on the algorithm. To measure the quality of the networks learned by the algorithms we used these networks learnt to classify a test set, resulting in the correct classification rate (ICC). To do that we studied the test set classification process and the propagation of evidences along the Bayesian network. The result of the studies is described on this work, along with the results and discussions about the experiments made with the introduced algorithms.

Sumário

Lista de figuras

Lista de tabelas

Lista de Algoritmos

Lista de Equações

Capítulo 1. Introdução

A mineração de dados é uma área de pesquisa destinada ao estudo, desenvolvimento e implementação de técnicas que permitam a extração de conhecimento de grande quantidade de dados armazenados em meio eletrônico [16]. O conhecimento extraído destas grandes bases de dados é então armazenado em uma estrutura bem definida que facilita a sua manipulação e interpretação chamada de estrutura de representação do conhecimento.

O Aprendizado de Máquina Indutivo é uma forma de Aprendizado de Máquina (AM) efetuado a partir do raciocínio sobre exemplos (registros de um banco de dados, por exemplo) fornecidos por um processo externo ao aprendiz [46]. Métodos tradicionais de aprendizado de máquina indutivo [33] são muito utilizados para se realizar a extração do conhecimento de bases de dados numa tarefa de mineração, porém muitos destes métodos possuem uma deficiência vinculada à taxa de atualização das bases de dados.

Considere um grande supermercado (com venda inclusive através da Internet) como exemplo. Suponha então, que se deseja aprender o comportamento dos clientes através da análise das transações efetuadas. O volume de transações diárias é bastante elevado em um ambiente como este. Assim, o conhecimento a ser aprendido está se alterando a todo o momento e, ao se utilizar métodos de aprendizado de máquina tradicionais para se obter o conhecimento a partir das bases de dados, tal conhecimento se torna desatualizado em um curto período de tempo.

Neste contexto, nota-se a necessidade de que o processo de aprendizado seja repetido sempre que novas informações são armazenadas nas bases de dados (o que ocorre com uma freqüência muito elevada). O maior problema da repetição do processo de aprendizado é que o seu custo computacional é elevado e sendo assim, se torna inviável a reconstrução da estrutura de representação de conhecimento através de métodos não incrementais[1] de AM sempre que novos dados precisam ser incorporados à estrutura.

[1] Neste trabalho, os algoritmos não incrementais de AM serão também chamados de algoritmos de batelada.

O aprendizado de máquina incremental [39] surge como uma busca de solução para o problema definido acima. Ou seja, a idéia central deste tipo de aprendizado é que uma base de conhecimento (estrutura de representação do conhecimento) possa ser atualizada sem que o processo de aprendizado seja refeito por completo. É uma forma de incorporar novos conhecimentos em uma base de conhecimento já definida.

Uma rede Bayesiana tem duas representações: a gráfica, que é altamente intuitiva, e a lógica, na qual podem ser feitas operações como classificação e inferência, por exemplo. Por causa de sua praticidade, ela é muito utilizada. Seu estudo tem se intensificado nos últimos anos [22], mas na área de aprendizado incremental ainda não existem muitos trabalhos.

Este trabalho tem por objetivo apresentar uma proposta para um método de aprendizado incremental de redes Bayesianas, levando em conta alguns trabalhos já feitos na área e, também, alguns dos algoritmos clássicos de aprendizado de redes Bayesianas. O método proposto é o ABC (Aprendizado Bayesiano em Camadas) e seus resultados serão analisados no contexto da mineração de dados, mais especificamente em tarefas de classificação. Durante o período de estudo do aprendizado incremental, descrito neste documento, foi proposto outro método, o AIP (Aprendizado Incremental de Parâmetros), que é um método embrionário de aprendizado incremental de redes Bayesianas. Ele foi importante no processo de desenvolvimento do método ABC.

O próximo capítulo trará uma descrição de uma rede Bayesiana, já que ela é efetivamente o objeto de estudo do trabalho. Este capítulo também traz a descrição de três algoritmos clássicos, K2, IC e PC, que são algoritmos que utilizam métodos heurísticos ou de independência condicional para fazer o aprendizado da rede.

O Capítulo 3 traz uma descrição do aprendizado incremental e alguns métodos de aprendizado incremental existentes para redes Bayesianas, enquanto o Capítulo 4 traz uma descrição da AD-Tree, que é uma estrutura de armazenamento de dados, mais especificamente é uma estrutura que armazena contagens de freqüência para uma base de dados, sendo possível fazer pesquisas nessa estrutura e assim extrair dela os dados necessários de forma rápida e simples. Essa estrutura possibilita uma melhora significativa no tempo de processamento do algoritmo de aprendizado de redes Bayesianas.

O Capítulo 5 traz um método inicial proposto aqui nesse trabalho, o AIP (Aprendizado Incremental de Parâmetros) que serviu de base para o desenvolvimento do método ABC, que é principal método proposto neste documento.

O Capítulo 6 descreve o método ABC (Aprendizado Bayesiano em Camadas), que é um método mais completo para aprendizado incremental de redes Bayesianas que o AIP, descrito no Capítulo 5. O Capítulo 7 traz então a conclusão deste trabalho.

Capítulo 2. Redes Bayesianas

Uma rede Bayesiana [37] é um grafo acíclico orientado, com os nós do grafo representando variáveis da base de dados, e os arcos do grafo representando uma relação entre as duas variáveis que definem o arco. Um arco de uma variável X para uma variável Y indica uma relação entre as duas variáveis (nesse caso, X é chamada de pai, e Y de filho), que pode ser vista (em alguns contextos) como uma relação causal, ou seja, X causa Y.

Cada variável X possui um conjunto de pais (π_X), sendo que esse conjunto pode ser vazio. A Figura 1 traz uma representação visual de uma rede Bayesiana, no caso a rede Bayesiana Credit [14].

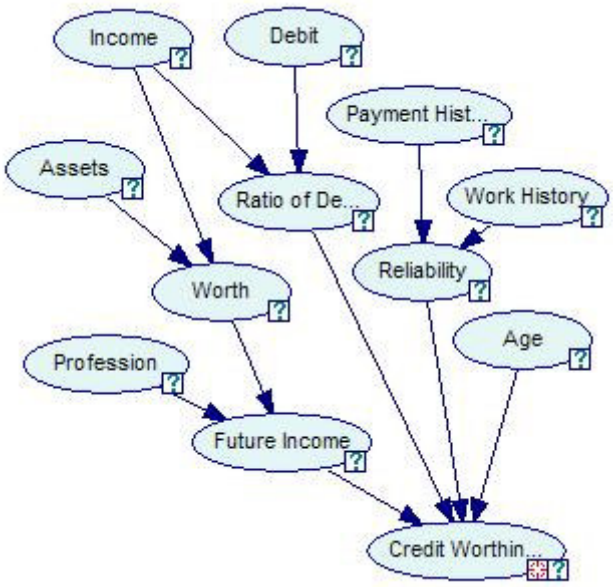

Figura 1: Rede Bayesiana Credit (fonte: GeNie[2] 2.0 [14]).

Cada nó do grafo, representando uma variável[3] (ou atributo) da base de dados, tem um parâmetro numérico associado a ele, que é a probabilidade condicional, da variável em questão, em relação ao estado de suas variáveis pais $P(X \mid \pi_X)$. Se uma variável não tem variáveis pais, a probabilidade associada a ela é a sua probabilidade marginal P(X).

[2] Software desenvolvido pelo laboratório Decision Systems Laboratory, da University of Pittsburgh.
[3] Como fazem referência ao mesmo elemento que descreve um problema, os termos nó (da rede Bayesiana), variável (do problema) e atributo (da base de dados) serão utilizados neste trabalho como sinônimos.

No grafo, para cada nó n_i que tenha filhos ou pais existe um conjunto chamado Markov Blanket [24] que é formado pelos nós pais de n_i, pelos nós filhos de n_i e pelos nós pais dos nós filhos de n_i que não sejam o próprio nó n_i. Os nós desse conjunto são os únicos nós com influência sobre o cálculo da distribuição de probabilidade condicional do nó n_i.

A Figura 2 traz uma matriz de probabilidade de uma variável, no caso a variável Worth da rede Bayesiana Credit (Figura 1). Na matriz da Figura 2 pode-se ver que a primeira linha tem o valor da variável Income (s0_30000, s30001_70000, s70001_more). Na segunda tem-se o valor da variável Assets (wealthy, average, poor), para cada valor de Income. Income e Assets são as variáveis pais da variável Worth. As combinações dos valores das variáveis pais de Worth feitas na primeira e segunda linhas são os valores possíveis para π_{Worth}. A matriz é feita correlacionando os valores de π_{Worth} com os valores da variável Worth (High, Medium, Low). O valor definido pelo valor k (High, por exemplo) da variável Worth e o estado j (s30001_70000 e average, por exemplo) de suas variáveis pais define a probabilidade do valor de Worth ser k sendo que seus pais estão no estado j (no exemplo, 0,699).

Income	s0_30000			s30001_70000			s70001_more		
Assets	wealthy	average	poor	wealthy	average	poor	wealthy	average	poor
High	0.899	0.001	0.001	0.989	0.699	0.1	0.989	0.90734	0.69
Medium	0.1	0.3	0.1	0.01	0.3	0.8	0.01	0.091743	0.3
Low	0.001	0.699	0.899	0.001	0.001	0.1	0.001	0.000917	0.01

Figura 2: Matriz de probabilidades para a variável Worth da rede Bayesiana Credit (fonte: GeNie 2.0 [14]).

Devido a essas características, uma rede Bayesiana é uma estrutura recomendada para se modelar o conhecimento intrínseco a uma base de dados. Ela fornece uma representação gráfica bastante intuitiva, já que todas as variáveis da base de dados podem ser representadas, com suas relações causais explicitadas, bem como fornece uma representação formal da base de dados, na qual se pode efetuar cálculos e extrair informações. Mais detalhes sobre propagação de evidências em redes Bayesianas podem ser vistos em [36].

Devido também a essas características, uma rede Bayesiana é adequada para tratar incerteza no domínio do problema em questão. Ao instanciar o valor de uma variável X_2, por exemplo, essa evidência é propagada pela rede Bayesiana. Os parâmetros numéricos das outras variáveis da rede Bayesiana (X_1, X_3,..., X_n)

são atualizados de acordo com essa evidência e, ao fazer isso, se adequa as variáveis da rede Bayesiana ao contexto atual. Desse modo têm-se, a cada evidência propagada, os valores mais prováveis para cada variável.

O aprendizado em redes Bayesianas (a partir de dados) pode ser visto como um processo que gera uma representação interna (na máquina) das restrições que definem um dado problema de modo a facilitar a recuperação dos dados, gerando assim um menor esforço computacional para essa representação. Este processo deve: aprender a estrutura da rede, ou seja, identificar as relações de interdependência dadas pelos arcos, e aprender as distribuições de probabilidades (parâmetros numéricos) com base na estrutura já definida. Por isso é, normalmente, dividido em dois subprocessos: aprendizado da estrutura e aprendizado dos parâmetros numéricos [9]. O aprendizado da estrutura é um problema NP-completo [24], assim, muito mais atenção tem sido dada à definição e otimização de algoritmos de aprendizado da estrutura do que ao aprendizado de parâmetros numéricos.

De uma maneira geral, pode-se dizer que os métodos Bayesianos de aprendizado de estrutura de redes Bayesianas dividem-se em duas classes principais. A primeira é a classe dos algoritmos que geram a rede através de uma busca heurística em uma base de dados, sendo o algoritmo K2 [11] um exemplo dessa classe.

O processo de busca heurística baseia-se na escolha de uma função de pontuação, também chamada de função heurística, e, com base na pontuação dada por essa função, na escolha da melhor dentre as opções possíveis.

Já na segunda classe estão os algoritmos que utilizam o conceito de independência condicional [37] para a construção da rede, como os algoritmos IC [38] e PC [45], por exemplo. Em algoritmos desta classe, o teste de independência condicional é utilizado para se fazer o aprendizado da estrutura de redes Bayesianas, pois este teste permite identificar se duas variáveis são condicionalmente independentes, dado um conjunto de variáveis. E, se duas variáveis são condicionalmente independentes, elas não podem estar conectadas no grafo pois, como visto, um arco entre duas variáveis denota uma relação de dependência.

As próximas três seções abordam de maneira mais detalhada algoritmos de aprendizado de redes Bayesianas representativas para as duas classes. A Seção 2.4 trata da classificação de dados usando redes Bayesianas e finaliza o capítulo.

2.1. Algoritmo K2

O algoritmo K2 foi proposto por Cooper e Herskovits [11] em 1992. É um algoritmo muito usado para aprender redes Bayesianas. Seu uso é freqüentemente motivado pela eficiência do K2 em encontrar uma estrutura de rede Bayesiana adequada, dada uma ordenação adequada das variáveis [23]. Neste trabalho o K2 é abordado como algoritmo referência para a classe de algoritmos de aprendizado de redes Bayesianas a partir de busca heurística

O K2 faz o aprendizado tanto da estrutura da rede Bayesiana quanto de seus parâmetros numéricos. Para o aprendizado de estrutura, utiliza uma função heurística para, com base em uma variável, identificar dentre os nós quais formam o conjunto de nós pais daquela variável. Realizando esse cálculo para todas as variáveis da base de dados o algoritmo consegue a estrutura da rede Bayesiana.

As funções de cálculo do algoritmo K2 só são válidas se aplicadas dentro do contexto definido pelo conjunto de hipóteses ξ e por um conjunto de notações e convenções, descritos na seqüência. Considere então o conjunto de hipóteses ξ:

- As variáveis da base de dados são discretas. Se forem contínuas, deverão ser discretizadas (ver [48]).
- Os dados que são usados para aprender a rede Bayesiana formam uma amostra aleatória, condicionalmente independente. Se cada instância (ou caso) da base não for condicionalmente independente uma das demais, o aprendizado da rede Bayesiana estará comprometido, pois a base de dados será tendenciosa, não retratando a realidade do problema em questão.
- Não há dados ausentes na amostra da base de dados utilizada para aprender a rede Bayesiana.
- Não se tem um conhecimento a priori dos parâmetros numéricos da rede Bayesiana.

O conjunto de notações e convenções é descrito abaixo:

- Z é um conjunto de variáveis aleatórias discretas, e x_i é uma variável em Z;
- D é uma amostra aleatória e condicionalmente independente;
- B_S é uma estrutura de rede Bayesiana;
- r_i é o tamanho do domínio de x_i, ou seja, o número de valores que x_i pode assumir;
- v_{ik} é o k-ésimo possível valor para x_i, com k variando de 1 a r_i;
- π_i é a lista (não ordenada) de variáveis pais de x_i;
- w_{ij} é a j-ésima instanciação possível de π_i, sendo que o número de estados possíveis de π_i é representado por q_i, que é obtido pela multiplicação dos r_i (tamanho do domínio) das variáveis de π_i;
- N_{ijk} é o número de vezes, em D, que π_i assume o estado w_{ij} e a variável x_i assume o estado v_{ik};
- N_{ij} é a soma dos N_{ijk}, ou seja, o número de vezes em que π_i assume o estado w_{ij}, em D.

O K2 supõe que o conjunto Z de variáveis aleatórias discretas é também ordenado. Essa ordenação é feita de modo que, para uma variável x_i, somente uma variável que venha antes dela na ordenação (x_{i-1}, por exemplo) possa ser seu pai. A ordenação das variáveis é então de suma importância no K2, já que uma ordenação inadequada irá certamente interferir negativamente no resultado do aprendizado da rede Bayesiana. O método Pred(x_i, Z) retorna os nós anteriores a x_i no conjunto ordenado Z.

A função g foi construída com base na função P(B_S, D), que faz o cálculo da probabilidade de uma estrutura de rede Bayesiana ser adequada, descrita na Equação 1 abaixo.

$$P(B_S, D) = P(B_S) \cdot \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}! \tag{1}$$

Levando P(B_S, D) ao seu máximo, se consegue isolar a parte da função que efetivamente leva P(B_S, D) ao seu máximo, como mostrado na Equação 2. Essa parte foi chamada de função g.

$$\max_{B_S}[P(B_S, D)] = \prod_{i=1}^{n} \max_{\pi_i} \left[P(\pi_i \rightarrow x_i) \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}! \right] \tag{2}$$

Isolando a parte da equação 2 que leva P(B_S, D) ao seu máximo se obtém a função g, descrita na equação 3.

$$g(x_i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \cdot \prod_{k=1}^{r_i} N_{ijk}! \tag{3}$$

O K2 irá procurar os valores de x_i e π_i que acarretam o valor máximo de g; e levando g ao seu máximo acarreta levar a probabilidade da estrutura B_S ao seu máximo, resultando então, que B_S é a estrutura mais adequada dentre as possíveis para representar a base de dados utilizada. O pseudocódigo a seguir é uma representação do algoritmo K2 como dado em [11].

1. **Algoritmo** K2;
2. {Entrada: Um conjunto de n nós ordenados Z, um limite superior u para o número de pais que um nó pode ter, e uma amostra D contendo m casos.}
3. {Saída: a estrutura de uma rede Bayesiana.}
4. **para** i := 1 **até** n **faça**
5. $\pi_i := \{ \}$;
6. $P_{old} := g(x_i, \pi_i)$;
7. OKToProceed := **true**;
8. **enquanto** OKToProceed **e** $| \pi_i | < u$ **faça**
9. z é o(s) nó(s) em $Pred(x_i, Z) - \pi_i$ que maximiza $g(x_i, \pi_i \cup \{z\})$;
10. $P_{new} := g(x_i, \pi_i \cup \{z\})$;
11. **se** $P_{new} > P_{old}$ **então**
12. $P_{old} := P_{new}$;
13. $\pi_i := \pi_i \cup \{z\}$;
14. **senão** OKToProceed := **false;**
15. **fim** {enquanto};
16. **fim** {para};

Algoritmo 1: K2

Na linha 9, $Pred(x_i, Z)$ é o conjunto dos nós que precedem o nó x_i em Z, conjunto das variáveis do problema. Devido à ordenação das variáveis, somente um dos nós que precedem o nó x_i, levando em conta a ordenação definida, pode ser seu pai.

O algoritmo itera sobre todos os nós em Z (o laço *for* da linha 4 à linha 17) fazendo uma busca heurística para encontrar o conjunto de nós pais do nó x_i ($\pi_i \cup \{z\}$) que maximiza a função g.

Na literatura existem diferentes implementações do K2. Normalmente a diferenciação se dá no trecho entre as linhas 11 a 14, como é mostrado a seguir, onde se define o critério para escolher qual conjunto de pais da variável x_i é o conjunto mais apropriado.

O código apresentado está usando o critério Hill Climbing, que busca sempre o maior valor para P_{new} e, conseqüentemente, para g. O critério Hill Climbing pode

levar g para um máximo local, que pode não ser o máximo global. Esta é uma característica deste critério. Entre outras variações do K2 podem-se utilizar os métodos de busca *branch-and-bound*, TABU e o *simulated annealing*, por exemplo, como mostrado em [47].

Para o aprendizado de parâmetros, o K2 utiliza o cálculo de freqüência relativa para estimar as probabilidades associadas a cada nó. Assim, o parâmetro numérico de um nó x_i de uma rede Bayesiana é a probabilidade condicional desse nó x_i assumir um valor v_{ik}, dado que seus nós pais (π_i) estão no estado w_{ij} ($P(x_i = v_{ik} \mid \pi_i = w_{ij})$), e será denotado por Θ_{ijk}. O símbolo ξ denota o conjunto de hipóteses descritas neste capítulo.

Então pode-se calcular o valor da função $E[\Theta_{ijk} \mid D, B_S, \xi]$, que é o valor esperado de Θ_{ijk}, dados a amostra D, a estrutura de rede Bayesiana B_S e o conjunto de hipóteses ξ. O valor de $E[\Theta_{ijk} \mid D, B_S, \xi]$ é calculado usando a fórmula da Equação 4.

$$E\left[\theta_{ijk} \middle| D, B_S, \xi\right] = \frac{N_{ijk} + 1}{N_{ij} + r_i} \tag{4}$$

Para fazer esse cálculo, é necessário que a estrutura da rede Bayesiana (B_S) tenha sido definida anteriormente.

2.2. Algoritmo IC

O algoritmo IC (Inductive Causation) foi proposto por Pearl e Verma [38] em 1991. É um algoritmo simples e, diferentemente do K2, utiliza a premissa de testes de independência condicional ou, mais especificamente, testes de d-separação, para aprender a estrutura da rede Bayesiana.

Testes de d-separação fazem uma medida, através de métodos estatísticos, do grau de influência de uma variável sobre outra, dado um conjunto de dados. Com base nesse grau de influência pode-se dizer se duas variáveis são condicionalmente independentes ou não.

O IC faz teste de independência condicional em todas as variáveis, e utiliza um conceito simples para montar a estrutura. Se, para dois nós A e B, existe um conjunto de variáveis, não contendo A e B, tal que o teste de independência

condicional para esses dois nós, é bem sucedido, ou seja, eles são independentes, esses dois nós A e B não devem ter um arco os conectando.

Naturalmente, se para dois nós A e B todos os testes de independência condicional falharem, deverá existir um arco entre eles.

Ao fazer esse teste sobre todos os pares de variáveis pode-se construir uma estrutura de rede Bayesiana onde os nós independentes não estão conectados. É importante notar que, eventualmente, o IC não consegue construir uma estrutura de rede Bayesiana totalmente orientada, tendo então uma estrutura de rede Bayesiana com alguns arcos sem orientação. Neste caso, deve-se utilizar algum artifício externo (auxílio de um especialista do domínio ou ordenação prévia das variáveis, por exemplo) para orientar todos os arcos e ter a rede completamente definida. Em [38] se define o algoritmo como sendo:

1. **Algoritmo** IC;
2. {Entrada: um grafo G com nós totalmente desconectados}.
3. **repita**
4. Seja x um nó do grafo, y outro nó do grafo, x ≠ y, tal que a dupla (x, y) não tenha sido instanciada anteriormente;
5. S(x,y) := {};
6. dSepara := falso;
7. **repita**
8. Instancie S(x,y) com um conjunto de nós do grafo que não tenha sido instanciado antes e que não contenha os nós x e y.
9. dSepara := testaDSeparacao(x, y, S(x,y));
10. **Até que** dSepara = verdadeiro ou |S(x,y)| > |G|;
11. **Se** dSepara = falso **então** adicione um arco entre x e y, sem orientação;
12. **Até que** todas as duplas possíveis tenham sido instanciadas.
13. **repita**
14. Seja x um nó do grafo, y outro nó do grafo e z outro nó do grafo, x ≠ y ≠ z, tal que x e y estejam conectados a z, mas x e y não estejam conectados e a tripla (x, y, z) não tenha sido instanciado anteriormente;
15. **Se** z não estiver contido em S(x, y) **então** oriente os arcos x – z e y – z na forma x → z ← y;
16. **Até que** todas as triplas possíveis tenham sido instanciadas
17. **repita**
18. Seja x um nó do grafo, y outro nó do grafo e z outro nó do grafo, x ≠ y ≠ z, tal que x e y não estejam conectados, x → z, y – z e a tripla (x, y, z) não tenha sido instanciado anteriormente;
19. Oriente o arco y – z no formato y ← z.
20. **Até que** todas as triplas possíveis tenham sido instanciadas
21. núcleo(P) := {};
22. **repita**
23. Seja x um nó do grafo, y outro nó do grafo e z outro nó do grafo, x ≠ y ≠ z, tal que x e z não estejam conectados, x → y, y → z e a tripla (x, y, z) não tenha sido instanciado anteriormente;
24. Adicione o arco y → z ao conjunto núcleo(P);
25. **Até que** todas as triplas possíveis tenham sido instanciadas

Algoritmo 2: IC

No algoritmo, a função testaDSeparacao(x, y, S(x,y)) testa se as variáveis x e y são condicionalmente independentes dado o conjunto S(x,y). O resultado desse procedimento é a subestrutura chamada de núcleo(P) em que todo arco direcionado é verdadeiro para a afirmação “X é causa direta de Y”. Esses relacionamentos são chamados de causas genuínas.

Os arcos que não foram direcionados nesse procedimento deverão ser direcionados com a ajuda de um especialista do domínio.

O IC não faz aprendizado de parâmetros da rede Bayesiana. Se for preciso fazê-lo pode-se utilizar o cálculo de freqüência relativa, similar ao cálculo do K2 [11] para aprendizado de parâmetros. Existem outras formas de se realizar esta etapa do aprendizado [36].

2.2.1. Exemplo do aprendizado supervisionado de estrutura de rede Bayesiana usando o algoritmo IC

Segue o conjunto de independências condicionais, obtidas através de testes de d-separação e que será usado como parâmetro de entrada para o algoritmo IC. A sintaxe A ⫫ B | {C} denota que o nó A é condicionalmente independente do nó B dado o conjunto {C}.

- A ⫫ D | {B}
- A ⫫ E | {C}
- B ⫫ C | {A}
- B ⫫ E | {D}
- C ⫫ D | {B, E}

O conjunto de nós {A, B, C, D, E} inicialmente forma o grafo G mostrado na Figura 3.

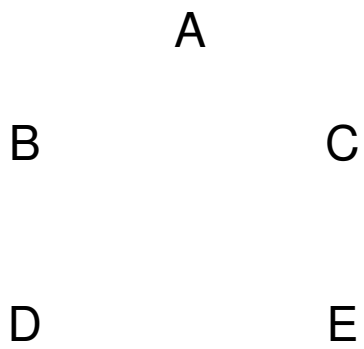

Figura 3: Grafo G inicialmente.

Após a identificação das independências condicionais apresentadas acima e a conseqüente criação de arcos entre os nós condicionalmente dependentes, têm-se o grafo G como mostrado na Figura 4.

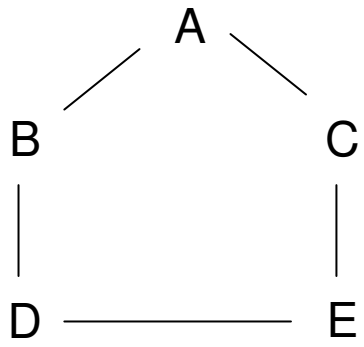

Figura 4: Grafo G depois de identificadas as independências condicionais.

Depois de aplicada a ordenação dos arcos do algoritmo IC, o grafo G fica na forma descrita pela Figura 5.

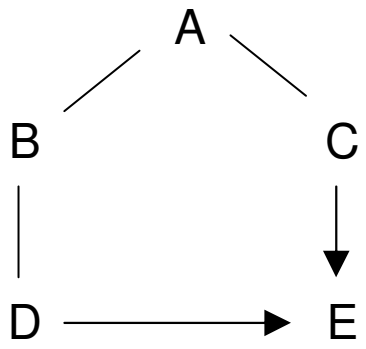

Figura 5: Grafo G depois de aplicada a ordenação de arcos do algoritmo IC.

Como o algoritmo IC não conseguiu ordenar totalmente o grafo G então se pode chamar um especialista para completar a ordenação do grafo. Depois da ação do especialista, o grafo G ordenado pode ficar na forma descrita pela Figura 6.

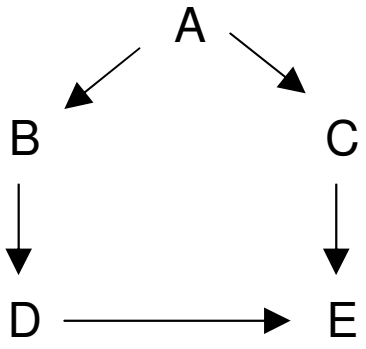

Figura 6: Grafo G ordenado.

2.3. Algoritmo PC

O algoritmo PC foi proposto em 1993 por Spirtes, Glymour e Scheines [45]. É um algoritmo de aprendizado de estrutura de rede Bayesiana baseado em testes de independência condicional, ou, mais especificamente, testes de d-separação, assim como o IC [38]. O nome do algoritmo foi formado a partir das iniciais do primeiro nome de Peter Spirtes e Clark Glymour, PC.

O PC pode ser abordado como uma variação do IC; a variação se dá na forma de montar a estrutura da rede Bayesiana. Enquanto o IC inicia a estruturação da rede Bayesiana a partir de nós não conectados, o PC inicia a estruturação a partir de uma estrutura em que todo nó está conectado a todos os outros nós do grafo, exceto ele mesmo.

O teste de d-separação testa se duas variáveis X e Y são independentes, dado um conjunto de variáveis (Z, por exemplo). Através de testes estatísticos pode-se determinar se X e Y são independentes, dado Z.

Colocando isso no contexto do PC, tem-se que se duas variáveis A e B da rede são independentes (dado um conjunto de variáveis), então A e B não podem estar conectadas na rede Bayesiana.

O PC, na forma como foi proposto, necessita de dois conjuntos de entrada, passados como parâmetro. O conjunto Adjacências(C,X) contém os vértices adjacentes a X no grafo C. O conjunto Sepset(X,Y) contém os vértices que d-separam X e Y.

Esses conjuntos são usados na orientação dos arcos no algoritmo, ou seja, originalmente no PC não está prevista a implementação de rotinas de cálculos de d-separação, embora o algoritmo as utilize.

Para abastecer o PC com as entradas que ele necessita pode-se usar uma rotina que faça testes de d-separação em um conjunto de dados e retorne as dependências e independências condicionais do conjunto.

Segue o algoritmo PC adaptado do original em [45].

1. **Algoritmo** PC;
2. {Entrada: um grafo C com nós totalmente desconectados, um conjunto Adjacências(C,X), para cada nó (X) do grafo, contendo os vértices adjacentes a X no grafo C, um conjunto Sepset(X,Y), para todos os possíveis pares ordenados (X, Y) de nós do grafo C, contendo os nós que d-separam X e Y.}.
3. **repita**
4. Seja x um nó do grafo C;
5. **repita**
6. Seja y um nó do grafo C, y ≠ x e não conectado a x;
7. Conecte os nós x e y (x – y);
8. **Até que** todos os nós diferentes de x estejam conectados a x;
9. **Até que** o grafo C esteja inteiramente conectado;
10. **repita**
11. Seja x um nó do grafo C, y outro nó do grafo e z outro nó do grafo, x ≠ y ≠ z, tal que x e y estejam conectados a z, mas x e y não estejam conectados e a tripla (x, y, z) não tenha sido instanciada anteriormente;
12. **Se** z não estiver contido em Sepset(x, y)
13. **então** oriente os arcos x – z e y – z na forma x –> z <– y;
14. **Até que** todas as triplas possíveis tenham sido instanciadas
15. **repita**
16. Seja x um nó do grafo;
17. Seja y um nó do grafo, não igual a x;
18. Seja z um nó do grafo, não igual a x e a y;
19. Tal que não exista um arco entre x e y, exista um arco orientado entre x e z, exista um arco não orientado entre y e z , não exista nenhum arco orientado para z exceto x –> z e a tripla (x, y, z) não tenha sido instanciada anteriormente;
20. Oriente o arco y – z no formato y <– z.
21. **Até que** todas as triplas possíveis tenham sido instanciadas;

Algoritmo 3: PC

O algoritmo até aqui descrito é o PC original. Existe uma variante dele chamada PC*, na qual se aplicam duas melhorias em relação ao PC original para diminuir o esforço computacional do algoritmo na formação dos conjuntos SepSet(X, Y).

No PC original os conjuntos SepSet(X, Y) são parâmetros de entrada do algoritmo. O algoritmo PC* vai então agregar essa funcionalidade de formação dos conjuntos SepSet(X, Y) e apresentar otimizações nessa nova funcionalidade.

A primeira melhoria apresentada no PC* se baseia na idéia de que se as variáveis A e B são independentes dado Adjacências(C,A) ou Adjacências(C,B), então elas são independentes dado um subconjunto de Adjacências(C,A) ou Adjacências(C,B). Dessa forma se repete menos vezes o processo de formação do conjunto SepSet(X, Y).

A idéia da segunda melhoria do PC* é que quanto mais rápido forem retiradas as ligações não efetivas entre nós do grafo C, menos iterações terão que ser feitas na formação do conjunto SepSet(X, Y), e as iterações que tiverem de ser feitas tendem a ser mais curtas, já que retirando ligações o tamanho do conjunto Adjacências(C,A) diminui.

Para aplicar esse conceito, foram apresentadas algumas heurísticas para determinar quais seriam as primeiras variáveis a ser analisadas:

1. Teste primeiro os pares de variáveis que são menos dependentes probabilisticamente, ou seja, que têm mais chance de serem independentes. Isso pode ser verificado através de testes de correlação parcial, para variáveis contínuas, procurando as variáveis com menor correlação parcial. Pode ser verificado também através de teste de χ^2 (qui-quadrado), no caso de variáveis discretas, procurando as variáveis com menor valor para χ^2. As variáveis restantes são selecionadas seguindo um critério a ser definido (ordenação de variáveis, ordem alfabética, dentre outras).

2. Para uma variável A, primeiro teste as variáveis que são menos dependentes probabilisticamente de A, em relação ao conjunto das variáveis mais dependentes probabilisticamente de A. Para descobrir quais variáveis devem ser testadas, use os mesmos testes da heurística descrita no item 1.

Os arcos que não foram direcionados tanto no PC quanto no PC* deverão ser direcionados com a ajuda de um especialista. O PC e o PC* não fazem o aprendizado de parâmetros da rede Bayesiana. Se for preciso fazê-lo pode-se utilizar o cálculo de freqüência relativa, similar ao cálculo do K2 [11] para aprendizado de parâmetros. Existem outras formas de se realizar esta etapa do aprendizado [36].

2.4. Classificação e Propagação de Evidências

Uma rede Bayesiana pode ser usada para fazer classificação de dados. O objetivo da classificação é utilizar o "conhecimento" presente em um modelo (rede Bayesiana, por exemplo) para identificar o padrão ao qual uma dada instância pertence. Neste sentido, quando uma rede Bayesiana é induzida com o propósito de ser aplicada em tarefas de classificação, pode-se utilizar o índice de classificação correta para mensurar o quão adequada a rede Bayesiana em questão está em relação à amostra de dados. Para se fazer a classificação de dados, utilizando-se uma rede Bayesiana, utiliza-se a propagação de evidências.

A Seção 2.4.1 descreve o funcionamento da propagação de evidências, a Seção 2.4.2 descreve o funcionamento da classificação de dados e por fim a Seção 2.4.3 faz uma discussão sobre os parâmetros numéricos.

2.4.1. Propagação de Evidências

A propagação de evidências é um processo que permite avaliar as probabilidades *a posteriori* das variáveis da rede Bayesiana [25]. O processo de propagação de evidências começa com a definição dessas evidências. Define-se, por exemplo, que a variável $x_3 = 1$ e que $x_5 = 3$, com base em evidências. Essas evidências são propagadas através da rede Bayesiana, atualizando as crenças de todos os nós da rede.

Todos os nós têm associado a eles uma matriz de probabilidades, também chamadas de crenças. Essa matriz representa a probabilidade do nó assumir um valor v_k dado que seus pais estão no estado j. Ao se alimentar a rede Bayesiana com as evidências, atualizam-se todas essas matrizes de probabilidades, a começar pelo nó ao qual a evidência se refere. Esse nó então envia mensagens para os nós que são ou pais ou filhos dele, atualizando as matrizes deles, que então enviam mensagens para seus pais ou filhos, exceto aquele que enviou a mensagem para ele, e assim sucessivamente até que todos os nós da rede Bayesiana sejam atualizados a partir da evidência.

A mensagem que um nó envia para seu filho se chama π. A mensagem que um nó envia para seu pai se chama λ. Toda a atualização das crenças dos nós da rede Bayesiana é feita com base nessas mensagens. Há dois tipos básicos de influência entre nós de uma rede Bayesiana: as influências causais (π), que são das causas para os sintomas, e as influências diagnósticas (λ), que são dos efeitos para as causas [25].

A propagação de evidências utiliza algum algoritmo para fazer o cálculo das matrizes dos nós. Existem vários desses algoritmos, porém, nesse trabalho só foi utilizado o algoritmo desenvolvido por Lauritzen e Spiegelhalter chamado PPTC (Probability Propagation in Trees of Clusters) [26].

2.4.2. Classificação de Dados

Antes de explicar o funcionamento da classificação de dados são necessárias algumas definições:

- Variável classe: é a variável que será classificada.
- Conjunto de teste: amostra de dados que será usada para fazer a classificação.
- Índice de classificação correta (ICC): porcentagem de acerto do processo de classificação.

Considere um conjunto de teste formado por 3 registros e descrito por 5 atributos (Tabela 1) e a rede Bayesiana da Figura 7. Considere ainda que a variável

classe seja a variável B da rede Bayesiana RB. A classificação ocorre da seguinte maneira:

- Para cada registro do conjunto de teste, se define o estado de cada variável da rede Bayesiana RB, exceto a variável classe, com o valor presente no registro atual;
- É feita a propagação das evidências na rede Bayesiana RB, inferindo assim o valor da variável classe;
- Compara-se o valor da variável classe com o valor inferido. Se forem iguais, então se incrementa o contador de sucessos;
- No final do processo se divide o contador de sucessos pelo número de registros, obtendo assim o índice de classificação correta (ICC).

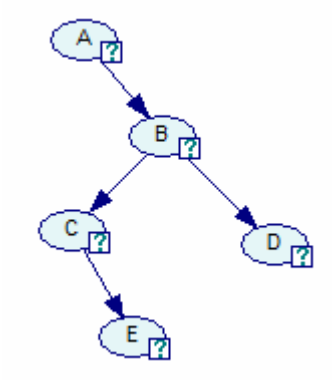

Figura 7: Rede Bayesiana RB.

Tabela 1: Conjunto de teste.

#	A	B	C	D	E
1	0	1	1	0	1
2	1	0	1	1	0
3	0	1	0	1	0

Para o conjunto de teste representado na Tabela 1, o funcionamento é o seguinte:

Primeiro registro – instancia-se A=0, C=1, D=0 e E=1. Ao fazer a propagação de evidências, se infere que o valor de B é 1, que é igual ao valor de B no registro, então é um sucesso. Incrementa-se o contador de sucessos.

Segundo registro – instancia-se A=1, C=1, D=1 e E=0. Ao fazer a propagação de evidências, se infere que o valor de B é 1, diferente do valor de B no registro, então é uma falha.

Terceiro registro – instancia-se A=0, C=0, D=1 e E=0. Ao fazer a propagação de evidências, se infere o valor de B como sendo 1, igual ao valor de B no registro, então é um sucesso. Incrementa-se o contador de sucessos.

Depois de terminado o processamento do conjunto de teste, divide-se o contador de sucessos pelo número de registros (2/3), obtendo assim o ICC de 66,66% para o conjunto de testes processado. Segue abaixo a representação em pseudocódigo do algoritmo de um classificador Bayesiano.

```
1.  Algoritmo classificador;
2.  {Entrada: Um conjunto de teste ct; uma rede Bayesiana RB; o índice da
    variável classe indClasse; o numero de nós da rede NumeroNos}
3.  {Saída: a porcentagem de acerto do conjunto de teste (ICC).}
4.  para cada registro de ct faça
5.     para i:= 0 até NumeroNos faça
6.        contLinha := contLinha + 1;
7.        se i <> indClasse então
8.           atualizaEvidencia(RB, i, ct(i));
9.     fim {para};
10.    atualizaCrença(RB);
11.    maisProvavel := getMaisProvavel(RB, indClasse);
12.    se maisProvavel = ct(indClasse) então
13.       nroAcerto := nroAcerto + 1;
14. fim {para};
15. retorne nroAcerto / contLinha;
```

Algoritmo 4: Classificador Bayesiano.

No pseudocódigo do algoritmo descrito em Algoritmo 4 há três métodos não definidos: atualizaEvidencia, atualizaCrença e getMaisProvavel. Esses métodos estão relacionados com a propagação de evidências e serão definidos a seguir:

O método atualizaEvidencia define o estado do nó i da rede Bayesiana RB para o estado n. Nesse método não há nenhuma propagação de evidências, somente a definição dessas evidências.

O método atualizaCrença faz a propagação das evidências especificadas anteriormente com o método atualizaEvidencia.

O método getMaisProvavel retorna o estado mais provável para o nó i da rede Bayesiana RB depois de feita a propagação de evidências dentro da rede Bayesiana.

A classificação de dados pode ser utilizada para avaliar o quão adequada a rede Bayesiana está ao conjunto de dados classificado. Mas não necessariamente o conjunto de dados classificado tem as mesmas distribuições probabilísticas do problema a que a rede Bayesiana se refere. Ou seja, um índice de classificação correta de 100% pode indicar uma rede Bayesiana muito bem adaptada ao conjunto de dados classificado, mas não necessariamente a rede Bayesiana estará tão bem adaptada a um conjunto de dados com outra distribuição de probabilidade.

Levando isso ao domínio do aprendizado incremental, onde a distribuição probabilística dos dados pode ser alterada ao longo do tempo, tem outro significado. A classificação de um conjunto de dados passa a ter um sentido temporal também. Afinal o conjunto de dados classificado é uma amostra da base de dados no instante t1, por exemplo.

Então o índice de classificação correta representa a adequação da rede Bayesiana ao conjunto de dados classificado, que não necessariamente tem a mesma distribuição probabilística da base de dados. Esse problema é mais freqüente em bases de dados pequenas, já que nesses casos o conjunto de teste também é pequeno, e nesse caso é maior a chance do conjunto de testes não ter a mesma distribuição probabilística da base de dados. Uma tática utilizada para minimizar este problema é chamada de validação cruzada, e consiste em gerar vários conjuntos de teste e de treinamento a partir de uma mesma base de dados. Usa-se o conjunto de treinamento para fazer o aprendizado de estrutura e parâmetros da rede Bayesiana e então se classifica o respectivo conjunto de teste. É feita a média dos índices de classificação correta (MICC) e então se passa a considerar o MICC em vez do ICC. Com isso se reduz as discrepâncias

entre os conjuntos de testes, e assim o MICC fica mais fiel à capacidade de generalização da rede Bayesiana.

Quanto ao problema temporal, a solução é analisar constantemente o índice de classificação correta. E através de técnicas de aprendizado incremental de redes Bayesianas tentar ajustar a rede Bayesiana à base de dados ao longo do tempo.

2.4.3. Relação entre Parâmetros Numéricos e ICC

Em [15], Drudzel e van der Gaag discutem sobre a necessidade de se ter parâmetros numéricos precisos na rede Bayesiana, ou seja, o impacto dos parâmetros numéricos sobre a rede Bayesiana. Eles acabam por concluir, empiricamente, que a estrutura da rede Bayesiana é muito mais importante que seus parâmetros numéricos.

Quanto aos parâmetros numéricos, são citados, em [15] estudos que sistematicamente variam os parâmetros numéricos de uma rede Bayesiana e examinam o seu comportamento, de acordo com seu uso (predição, classificação, agrupamento, etc.). Nesses estudos há casos em que grandes variações nos parâmetros numéricos de uma rede Bayesiana acabam não influenciando o comportamento da rede Bayesiana. Do mesmo modo, em outros estudos grandes variações nos parâmetros numéricos acabam influenciando bastante o comportamento da rede.

Dessa forma se conclui em [15] que não se pode afirmar qual é a relevância dos parâmetros numéricos para a rede Bayesiana de forma geral. Cada rede Bayesiana terá seu nível de sensibilidade quanto aos parâmetros numéricos.

Essa conclusão pode parecer conflitante com o resto dessa seção, afinal se os parâmetros numéricos, ou crenças, ou probabilidade *a priori* não influem tanto na probabilidade *a posteriori*, então pode não ser interessante utilizar os parâmetros numéricos.

Porém não se pode desprezar a possível contribuição dos parâmetros numéricos no ICC. Cada base de dados tem uma sensibilidade aos parâmetros numéricos. Pode ser que para uma base de dados BD1 os parâmetros numéricos tenham até mais impacto no ICC do que a estrutura da rede Bayesiana.

Nas Seções 5.2 e 6.3, de discussão dos experimentos realizados, é discutida a relação entre os parâmetros numéricos de cada base de dados utilizada e seu respectivo ICC.

Capítulo 3. Aprendizado Incremental

A idéia de aprendizado incremental provém da observação de que a maior parte do conhecimento humano pode ser visto como um processo gradativo de formação de conceitos ou como a habilidade humana de incorporar conhecimento proveniente de novas experiências em estruturas de conhecimento já formadas [17].

O aprendizado humano pode então ser caracterizado como incremental, na medida que os conceitos são formados ao longo do tempo à luz de novas informações. Já o aprendizado de máquina é melhor se feito em batelada, pois para os processos tradicionais de aprendizado é melhor ter todas as informações disponíveis no momento do aprendizado.

Dessa forma o aprendizado incremental de máquina vem como uma forma de simular o processo de aprendizado humano no processo de aprendizado de máquina. Então é preciso propor novos processos de aprendizado ou adaptar os processos existentes para que eles possam operar de forma incremental com a mesma eficiência, ou melhor, que na forma de batelada.

Atualmente, a grande motivação para se fazer o aprendizado incremental vem, principalmente, das empresas e seus enormes bancos de dados [40]. O tamanho das bases de dados disponíveis e suas taxas de crescimento hoje em dia requerem um esforço computacional consideravelmente grande, o que praticamente inviabiliza a atualização constante da base de conhecimento, se for feito do modo tradicional, usando algoritmos de aprendizado em batelada (não incrementais).

Algoritmos de aprendizado incremental são estudados na comunidade de aprendizado de máquina desde a metade da década de 80, quando se focou em algoritmos de *clustering*, como o trabalho de Fisher [17]. O trabalho de Gennari [21] apresenta um resumo dos algoritmos de clustering incremental (considerados os precursores do aprendizado incremental) feitos até 1989.

Alguns algoritmos de aprendizado de máquina podem ser inerentemente incrementais dado que não constroem modelos, ou ainda se utilizam de modelos fixos para representar o conhecimento contido em um conjunto de dados. Como exemplo, pode-se citar o KNN [33], para aprendizado baseado em instâncias, o

Naïve-Bayes [19], essencialmente um classificador, que pode ser considerado um algoritmo de aprendizado incremental, embora somente para os parâmetros numéricos da rede Bayesiana.

Um algoritmo de aprendizado é considerado incremental, segundo Langley [29], se ele processa um conjunto de treinamento por vez, não reprocessa qualquer conjunto de treinamento anterior e retém somente uma estrutura de conhecimento em memória.

Já segundo Domingos e Hulten [12] e [13], que propuseram uma outra definição de algoritmo de aprendizado incremental, visando adequar a definição de Langley para tratar bases de dados que podem crescer indefinidamente, um algoritmo de aprendizado, para ser considerado incremental, deve seguir as seguintes restrições:

- O algoritmo deve processar um registro em um tempo pequeno e constante;
- O algoritmo deve ser capaz de construir um modelo usando no máximo uma leitura dos dados;
- O algoritmo deve usar uma quantidade fixa de memória principal, independentemente do número de registros que o algoritmo tenha processado;
- O algoritmo deve construir um modelo usável em qualquer momento, ao contrário de somente fazê-lo quando terminar de processar os registros;
- O algoritmo deve produzir um modelo que seja equivalente ou quase idêntico ao modelo que seria obtido usando um algoritmo de aprendizado não incremental.

No campo de aprendizado incremental de redes Bayesianas, não existem ainda muitos trabalhos. Dentre os mais relevantes, pode-se citar o trabalho de J. Roure [40], que será descrito na Seção 4.2, o trabalho de Lam e Bacchus [28] e o trabalho de Friedman [18], que será descrito na Seção 4.1.

Existe uma diferença primordial entre os algoritmos de aprendizado incremental utilizados em classificadores Naïve-Bayes, aprendizado baseado em

instâncias, clustering e os algoritmos de aprendizado incremental de redes Bayesianas.

Os algoritmos de aprendizado utilizados em classificadores, como o Naïve-Bayes [19] por exemplo, aprendizado baseado em instâncias, como o KNN [33], e clustering, como o C-means [4], são baseados em relações entre dados [40]. Então quando um dado novo precisa ser incorporado à estrutura, se atualiza os parâmetros da rede, no caso do Naïve-Bayes, ou se adiciona o dado novo à base de conhecimento, no caso do KNN, ou se faz um estudo para determinar à qual cluster o dado pertence e, então, se incorpora o dado novo na estrutura, no caso do C-means.

O aprendizado de redes Bayesianas, por outro lado, se baseia em relações entre variáveis [40]. Portanto um dado novo que deve ser incorporado à rede pode acarretar não só em atualizações dos parâmetros da rede, mas também em uma atualização de sua estrutura.

Portanto o aprendizado incremental de redes Bayesianas é mais complexo do que o aprendizado incremental dos outros classificadores. E assim sendo algumas restrições das definições de algoritmo incremental apresentadas anteriormente não podem ser aplicadas totalmente aos algoritmos de aprendizado incremental de redes Bayesianas. A definição de Langley [29] é muito restritiva, assim como a definição de Domingos e Hulten [12] e [13]. No desenvolvimento de algoritmos de aprendizado incremental de redes Bayesianas tenta-se adequar o algoritmo a essas definições, porém tendo em mente que elas não foram propostas para algoritmos de aprendizado de redes Bayesianas.

A definição de algoritmo de aprendizado incremental de redes Bayesianas, segundo Friedman e Goldszmidt [18], diz que, para um algoritmo de aprendizado de redes Bayesianas ser considerado incremental, a cada iteração, o algoritmo processa um novo dado u_n e, então, produz a próxima hipótese S_{n+1}. Essa hipótese é então usada para fazer a tarefa requisitada (predição, diagnóstico, classificação, etc.) no próximo novo dado u_{n+1}, que por sua vez é utilizado para produzir a próxima hipótese e assim por diante. O algoritmo pode gerar um modelo novo depois que um número k de dados tenha sido processado.

Quando se adota uma política de aprendizado incremental, normalmente se leva em conta os custos computacionais dos processos envolvidos. No caso das redes Bayesianas, deve-se considerar então os custos do aprendizado da

estrutura e do aprendizado dos parâmetros numéricos. Como já visto, o aprendizado da estrutura da rede Bayesiana é um processo de elevado esforço computacional; já a atualização dos parâmetros numéricos da rede Bayesiana, é um processo que tem esforço computacional menor.

3.1. Proposta de Friedman e Goldszmidt

Em [18], Friedman e Goldszmidt propõem três estratégias para aprendizado incremental no contexto de aprendizado de redes Bayesianas. As duas primeiras exploram extremos do aprendizado incremental, e a terceira é um meio-termo entre elas.

A primeira estratégia (chamada Naïve) explora o extremo da qualidade, e consiste em guardar estatísticas de aprendizado para todas as redes possíveis e, ao chegar um dado novo, atualizar essas estatísticas e escolher, dentre as redes possíveis, a melhor. É uma estratégia não implementável devido ao seu esforço computacional elevado. Mas, teoricamente, a sua utilização traria sempre a melhor rede possível.

A segunda estratégia (chamada MAP) explora o extremo do desempenho, em detrimento da qualidade. Consiste em aprender uma estrutura (E_{t1}) de rede com os dados disponíveis naquele momento (t1) e considerar essa estrutura representativa para os dados disponíveis. Com a chegada de novos dados (em um momento t2), utiliza-se a estrutura (E_{t1}) anterior como passo inicial, e a partir dela se aprende, considerando que a estrutura E_{t1} representa os dados disponíveis em t1, outra estrutura (E_{t2}), usando os dados novos e as estatísticas presentes na estrutura E_{t1}.

Nessa estratégia as redes aprendidas, após algumas iterações, convergem para um modelo e, a partir desse ponto, os dados novos não mais influenciam a estrutura da rede, o que, segundo Friedman, é um problema, já que, dessa forma, dados novos que não necessariamente seguem o modelo estabelecido não acarretam uma mudança no modelo, e desse modo o modelo deixaria de ser um modelo atualizado. Lam e Bacchus, em [28] propõem uma técnica semelhante.

A terceira estratégia (chamada Incremental) é um meio-termo entre as duas primeiras. Consiste em se aprender uma estrutura de rede Bayesiana e a partir

dela fazer variações, como adicionar ou retirar um relacionamento ou inverter sua direção. As variações da rede aprendida formam o conjunto de redes vizinhas.

Então com a chegada de dados novos se atualiza a rede atual e as redes do conjunto de vizinhos. As redes do conjunto de vizinhos e a rede atual são comparadas e a melhor rede é escolhida como a atual.

Para as três estratégias de Friedman, o conceito de melhor rede Bayesiana é definido pela melhor pontuação da rede em relação a uma função de pontuação, como, por exemplo, a função g do K2, o MDL ou o BDe [18].

As duas primeiras estratégias da proposta de Friedman e Goldszmidt têm como objetivo demarcar as fronteiras do aprendizado incremental, relativas à qualidade e desempenho. Demarcando essas fronteiras pôde-se desenvolver uma estratégia que esteja dentro desses limites, e que, portanto, mescle características tanto de qualidade quanto de desempenho. Essa estratégia desenvolvida foi a terceira estratégia, a Incremental, que é uma estratégia que gera redes Bayesianas com mais qualidade que a segunda estratégia, mas com menos qualidade do que a primeira estratégia. Da mesma forma, gera redes Bayesianas com melhor desempenho do que a primeira estratégia, mas com menos do que a segunda estratégia. Essa estratégia é a mais adequada das estratégias de Friedman e Goldszmidt para ser posta em produção.

3.2. Proposta de J. Roure

Em [40], Roure propõe um método de aprendizado incremental de estrutura de rede Bayesiana baseado em duas heurísticas. Essas duas heurísticas podem ser adaptadas a vários algoritmos de aprendizado de rede Bayesiana em batelada, desde que usem a busca hill-climbing. No trabalho de Roure o método foi aplicado a quatro algoritmos: K2[11], CL[10], B[5] e HCMC[7] e [8].

A primeira heurística é chamada TOCO (*Traversal Operators in Correct Order*). Alguns pré-requisitos precisam ser cumpridos para se usar essa heurística.

1. Operadores devem ser definidos. Os operadores possíveis são: adicionar um arco, retirar um arco ou inverter o sentido de um arco. Deve-se definir então um subconjunto desses operadores. Nesse trabalho não se usou operações sobre nós, somente sobre arcos.
2. A função que definirá a qualidade da rede Bayesiana deve ser escolhida. A função g do K2 ou o MDL são exemplos de funções de qualidade nesse caso.
3. O caminho que o algoritmo percorre (adiciona arco entre X_1 e X_4, retira arco entre X_3 e X_8, inverte arco entre X_2 e X_5) para chegar à estrutura ideal durante o aprendizado deve ser armazenado.

A heurística TOCO funciona com base em funções de continuidade de qualidade. A cada operação feita no grafo é feita uma pontuação do grafo resultante. Fazendo um gráfico “Operação X Qualidade”, é possível estimar uma curva que descreve a qualidade do grafo em relação ao caminho de operações feitas.

Pela característica da busca *hill-climbing*, uma operação só será feita no grafo se ela for melhor que a anterior. Graficamente então a curva de Operação X Qualidade será crescente.

A finalidade da heurística TOCO é saber se é necessário acionar a atualização da estrutura da rede Bayesiana, ou seja, saber se a estrutura atual da rede é adequada para os dados novos.

Com a chegada de novos dados, repete-se o caminho feito para a rede atual nos dados novos. Se, em uma dada operação aplicada aos dados novos, a qualidade do grafo for menor que a qualidade da operação anterior, é necessário acionar a atualização da estrutura do grafo, segundo a heurística TOCO.

A segunda heurística é chamada de RSS (*Reduced Search Space*). Ela se utiliza dos mesmos pré-requisitos da heurística TOCO. Essa heurística armazena, a cada operação no caminho que o algoritmo percorre até a estrutura mais adequada, as k operações mais próximas da atual.

É importante ressaltar que a heurística RSS complementa a heurística TOCO, atuando para reduzir o espaço de busca para a alteração na rede Bayesiana. Alteração essa que se viu necessária através do uso da heurística

TOCO. Dessa forma, quando for necessário atualizar a estrutura da rede Bayesiana, será usada alguma das operações armazenadas pela heurística RSS.

A premissa básica dessa heurística é que, se for considerado que a estrutura da rede Bayesiana atual é adequada para os dados armazenados até aquele momento, então ela estará próxima da estrutura adequada para os dados novos. Dessa forma, se dá preferência na análise às estruturas mais próximas da estrutura atual.

Essas duas heurísticas são aplicáveis a virtualmente todo algoritmo que faça operações sobre grafos e que use uma função para determinar a qualidade da rede Bayesiana.

Capítulo 4. AD-Trees

A AD-Tree foi proposta no artigo de Moore e Lee [34]. AD-Trees são um tipo de kd-trees [3] e [35]. O k de kd-trees é o número máximo de dimensões pelos quais cada nó da árvore pode se expandir. Cada dimensão representa um valor do domínio de cada nó ou um outro nó relacionado ao nó que está sendo analisado. Pode-se considerar então que cada dimensão é um caminho que pode ser percorrido na árvore. O A de AD-Trees significa *All* (todas). Então AD-Trees são um tipo de kd-trees que se expandem por todas as dimensões possíveis, ou seja, para cada nó do grafo se expande uma dimensão para cada um dos valores de seu domínio, e para cada nó do grafo se expande uma dimensão para cada um dos outros nós do grafo que se relacionam com este nó.

A AD-Tree é um estrutura de representação de dados, e o objetivo dela é armazenar contagens de freqüência para uma base de dados. É possível fazer consultas nessa estrutura, e através do uso dessas consultas se consegue otimizar bastante o tempo de processamento de um algoritmo de aprendizado de redes Bayesianas, como se verá adiante.

Outra motivação para seu uso é que essa estrutura é mais adequada para representar dados em um algoritmo incremental, já que, segundo as definições de algoritmo incremental de Langley [29] e Domingos e Hulten [12] e [13] o algoritmo deve processar uma vez somente cada registro. Utilizando a AD-Tree pode-se usar essa leitura do dado que é permitida para construir a AD-Tree e então passar a utilizar essa estrutura para extrair informações.

AD-Trees são árvores esparsas de tamanho variável. A AD-Tree tem dois possíveis tipos de nó, o nó *Vary* e o nó *ADnode*. Cada nó *ADnode* da árvore terá um inteiro e ponteiros para outros nós *Vary*. Cada nó *Vary* terá somente ponteiros para outros nós *ADnode*. Portanto cada nó tem tamanho pequeno. Porém o número de nós é dependente do número de variáveis da base de dados e do tamanho do domínio de cada variável. Se o tamanho da base de dados for grande com variáveis de domínios grandes, então o tamanho da AD-Tree será grande, com muitos nós de cada tipo.

O uso de AD-Trees é particularmente interessante para algoritmos de aprendizado de redes Bayesianas operando em bases de dados com atributos de

domínio discretos que usem o método freqüentista para contagem de instâncias da base de dados, como o K2 ou o ABC, onde ele será efetivamente utilizado.

O uso da AD-Tree é muito mais eficiente do que fazer varreduras completas na base de dados. Ao invés de se analisar, registro por registro, o número de registros da base de dados que atendem a uma consulta, apenas se acessa um nó *Adnode* relativo a essa consulta e o número de registros que atendem a essa consulta estará incluso neste nó. Considerando-se o número de registros da base de dados como sendo n e o número de atributos da base de dados como sendo r, a varredura completa da base de dados teria n x r operações, enquanto que a mesma consulta usando a AD-Tree teria no máximo r operações.

Nesta seção será apresentada a teoria da AD-Tree. Para exemplos práticos do uso da AD-Tree veja a Seção 6.1 – AD-Tree, onde é descrito como foi implementada a AD-Tree nesse trabalho, e, além disso, é mostrado como funciona a AD-Tree na prática.

Seguem abaixo algumas definições importantes para a AD-Tree.

Consulta

Podem ser feitas consultas (ou *queries*) à AD-Tree. As consultas são no formato a_1=1; a_2=3; onde a_1 e a_2 são variáveis da base de dados e 1 ou 3 são valores do domínio da variável em questão. O formato de uma consulta é então: <variável da base de dados>=<valor do domínio da variável>. Se alguma variável da base de dados não for especificada na consulta então ela assume o valor a_n=* (qualquer valor).

Contagem

A contagem (*count*) de uma consulta é um inteiro representando o número de registros na base de dados que satisfaz as condições impostas pela consulta. É denotado por C(consulta).

Tabelas de Contingência

Cada conjunto de atributos da base de dados tem uma tabela de contingência (*contingency table*) associada a ele. Para o conjunto de atributos a_1, a_2,..., a_n, a tabela de contingência associada a ele é denotada por ct(a_1, a_2,..., a_n). A tabela de contingência tem um registro para cada possível combinação dos valores das variáveis a_1, a_2,..., a_n. A linha correspondente a a_1=1, a_2=3,..., a_n=1 armazena o *count* C(a_1=1, a_2=3,..., a_n=1). A Figura 8 mostra exemplos de tabelas de contingência.

Uma tabela de contingência condicional (*conditional contingency table*) é denotada por ct($a_{i(1)}$,..., $a_{i(n)}$ | $a_{j(1)}$ = u_1,..., $a_{j(p)}$=u_p). Essa tabela é a tabela de contingência das variáveis à esquerda de | que satisfazem a condição imposta pela consulta à direita de |.

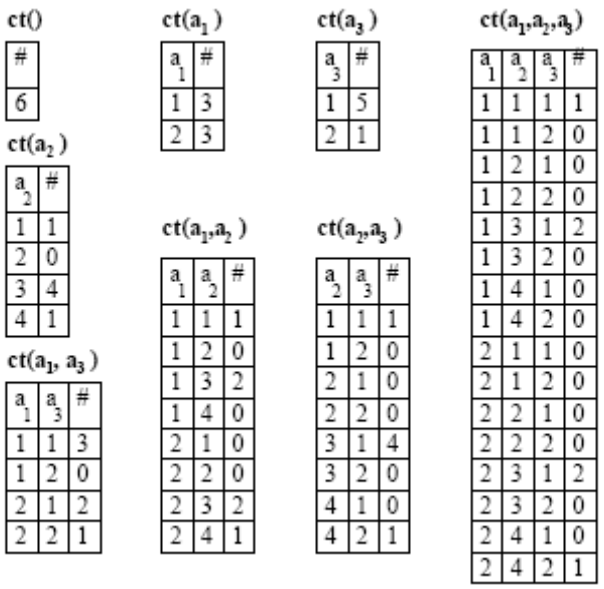

Figura 8: Tabelas de contingência.

AD-Tree

A AD-Tree é a estrutura de dados que será usada para representar todos os possíveis *counts* da base de dados. A proposta inicial não apresenta nenhuma tática de redução de uso de memória, portanto é a mais completa. Um exemplo de AD-Tree é mostrado na Figura 9 a seguir.

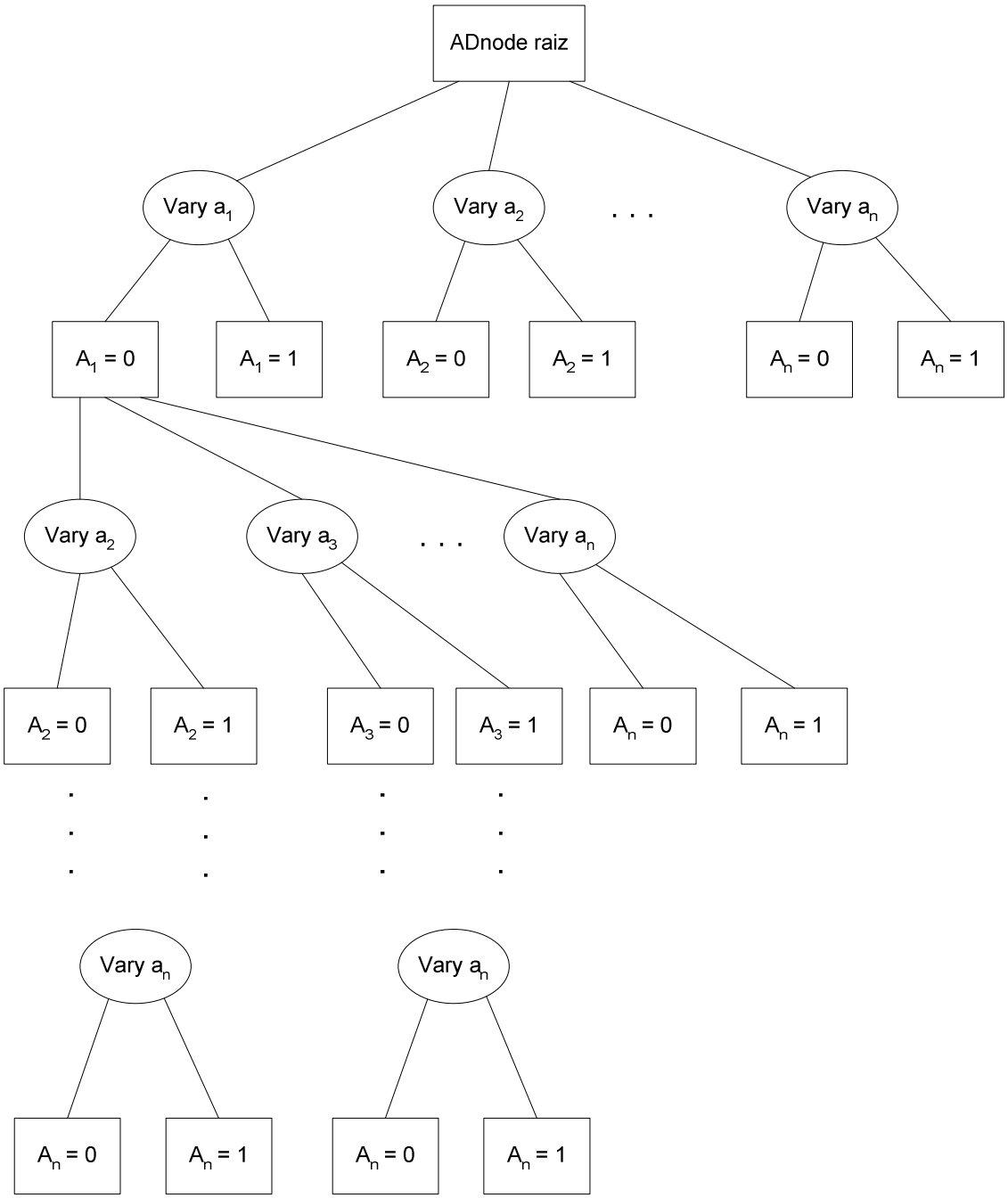

Figura 9: Exemplo de uma AD-Tree.

Como mencionado anteriormente, na AD-Tree existem dois tipos de nós. O nó *ADnode* é representado por um retângulo e é formado por ponteiros para nós filhos *Vary* e também o *count* a que esse nó se refere. O nó *Vary* é representado por uma elipse e consiste em ponteiros para nós filhos *ADnode*.

Cada nó *ADnode* representa uma consulta e armazena o *count* dessa consulta. O nó *Vary* a_j filho de um nó *Adnode* q_i tem um nó filho q_n (*ADnode*) para cada um dos valores do domínio da variável a_j. O k-ésimo filho q_k (*ADnode*) do nó

Vary a_j representa a mesma consulta do nó pai q_i (*ADnode*) de *Vary* aj, com a restrição adicional $a_j = k$.

O número máximo de nós (n_{max}) de uma AD-Tree é especificado pela Equação 5, onde *M* é o número de variáveis na base de dados e r_i é o tamanho do domínio da variável i. Desse total de nós, possivelmente vários podem ser eliminados. Para isso existem três técnicas para reduzir o tamanho da AD-Tree.

$$n_{\max} = \prod_{i=1}^{M} (r_i + 1) \tag{5}$$

Primeira Técnica – Eliminando os nós com *count*=0

Segundo essa técnica, ao encontrar um nó com *count*=0 não se instancia um *ADnode* para ele. Ao invés disso, coloca-se o valor nulo (NULL) em seu lugar. Desse modo o *ADnode* que tem *count*=0 e os nós filhos desse nó são cortados.

Isso pode ser feito, pois se um nó tem *count*=0, todos seus filhos são especificações da consulta que resultou o *count*=0, portanto também resultarão em *count*=0.

Essa técnica não implica em nenhuma alteração na complexidade de se recuperar uma informação na AD-Tree. A única alteração que precisa ser feita é uma validação do valor do ponteiro para *ADnode*. Se algum dos ponteiros para *ADnode* que façam parte da consulta for nulo então já se retorna o valor do *count* como 0.

Segunda técnica – Eliminando os nós com o valor mais comum

Essa técnica diz que, se após a redução da *AD-Tree* ao se cortar os nós com *count*=0 o tamanho da *AD-Tree* ainda estiver muito grande, pode-se reduzir ainda mais a *AD-Tree* ao se cortar os nós com o valor mais comum de cada variável, chamado de MCV. No lugar do nó com o *count* relativo ao MCV de cada variável, armazena-se o valor nulo (NULL). O restante dos nós é feito da mesma forma.

Porém essa técnica só pode ser aplicada para *AD-Trees* onde o valor do *count* relativo ao MCV de cada variável possa ser conseguido através da combinação dos valores dos *counts* dos outros nós da AD-Tree. A combinação dos valores dos *counts* dos outros nós da AD-Tree é feito através de uma função recursiva.

Essa técnica elimina vários nós da rede, mas em troca a performance das consultas na *AD-Tree* é diminuída por conta da função recursiva para determinar o valor do *count* do MCV de cada variável.

Terceira técnica – Leaf-Lists

Essa técnica diz que, se um nó *ADnode* tem *count* menor do que um valor especificado R_{min}, então não compensa expandir a sub-árvore relativa a esse nó. Ao invés disso, se armazena ponteiros para os registros na base de dados em que a consulta desse nó é atendida.

Essa técnica também reduz bastante o tamanho da AD-Tree, porém em compensação se tem que armazenar a base de dados em memória. Além disso, também é necessário implementar funções para processar os registros da base de dados e a partir deles conseguir o *count* da consulta desejada.

Um aspecto negativo do uso das AD-Trees é que o tempo para criar a AD-Tree é grande, assim como o tempo de se atualizar uma AD-Tree a partir de dados novos. Tem-se então de se pesquisar técnicas que diminuam o tempo de confecção da AD-Tree.

Por isso, indica-se o uso da AD-Tree para bases de dados grandes, com mais de 100 mil registros [34]. Porque para bases menores, o tempo de se criar a AD-Tree pode ser maior do que fazer a varredura na base de dados.

Capítulo 5. Proposta de Algoritmo de Aprendizado Incremental Ingênuo de Parâmetros (AIP)

Como já visto anteriormente, o aprendizado incremental de redes Bayesianas é um tema relativamente novo e, desta forma, ainda não se tem muitos resultados divulgados na literatura [41]. Alguns trabalhos, entretanto, já mostraram que bons resultados podem ser obtidos quando se busca adaptar métodos de aprendizado Bayesiano tradicionais para que eles possam atuar como um processo de aprendizado incremental [1], [20], [27], [41] e [42]. Nestas aplicações o aprendizado de estrutura é realizado de maneira incremental.

Já o objetivo inicial deste trabalho de pesquisa foi verificar qual o impacto da atualização somente dos parâmetros numéricos da rede Bayesiana, isto é, verificar se a atualização somente dos parâmetros numéricos traz algum ganho em índices de classificação correta (ICC) para a rede Bayesiana atualizada, e analisar qual o ganho de desempenho conseguido, considerando o esforço computacional realizado.

Como o aprendizado de parâmetros exige um esforço computacional bem menor que o aprendizado de estrutura, realizar o aprendizado incremental apenas dos parâmetros numéricos traz uma economia considerável de tempo e recursos computacionais para o processo de mineração de dados.

Existem alguns trabalhos sobre aprendizado incremental de parâmetros; esses trabalhos, entretanto, seguem outra linha e, dentre eles, pode-se destacar os trabalhos de Spiegelhalter & Lauritzen [44], Buntine [6] e Lauritzen [30]. Esses trabalhos seguem a linha de atualizar os parâmetros da rede Bayesiana, sendo que a estrutura dessa rede Bayesiana é definida com a ajuda de um especialista. Os parâmetros dessa rede devem seguir uma distribuição de probabilidade específica, como a normal ou Dirichlet, ou seja, eles trabalham com o enfoque Bayesiano dos parâmetros numéricos. Os métodos propostos nesses artigos atualizam as curvas de probabilidade, ajustando a média e a variância da distribuição de probabilidade.

Na proposta aqui apresentada (o algoritmo AIP) seguiu-se o enfoque freqüentista para os parâmetros numéricos que, ao invés de usar distribuições de probabilidade para os parâmetros numéricos, utiliza a sua freqüência relativa.

É importante ressaltar que não se está afirmando que o enfoque freqüentista é melhor do que o enfoque Bayesiano. O enfoque freqüentista, entretanto, é mais simples de ser implementado e mais adequado quando não se tem qualquer conhecimento *a priori* do comportamento das variáveis da base de dados e, devido a sua simplicidade, é normalmente empregado em tarefas de mineração de dados. Além disso, como mostrado em [49], assumir uma distribuição de probabilidade (Gaussiana, por exemplo) para uma variável da rede Bayesiana sem estar certo da validade dessa distribuição de probabilidade, pode implicar resultados ruins na classificação ou na inferência, usando essa rede Bayesiana. Por outro lado, quando se tem uma base de dados suficientemente grande, a freqüência relativa pode ser usada para estimar a distribuição de probabilidade dos nós da rede Bayesiana [30].

De fato, o enfoque Bayesiano é mais adequado para representar os parâmetros numéricos de uma rede Bayesiana, porém ele não pode ser usado em toda rede Bayesiana, já que não se pode afirmar que todas as variáveis seguem alguma distribuição de probabilidade que possa ser identificada com facilidade. Desse modo o enfoque freqüentista é mais abrangente, já que a freqüência relativa pode ser aplicada a qualquer variável discreta quando há um conjunto de observações relativo a esta variável.

O uso do enfoque Bayesiano para os parâmetros numéricos também envolve um nível maior de complexidade do que o uso do enfoque freqüentista, tanto no aprendizado dos parâmetros quanto na sua atualização e, especialmente, na propagação das evidências.

A proposta do AIP teve como foco a investigação do impacto da atualização dos parâmetros numéricos de uma rede Bayesiana, usando o enfoque freqüentista, na classificação de dados usando essa rede. Para tanto, o AIP utiliza os seguintes passos metodológicos: inicialmente deve-se realizar o aprendizado de estrutura como num processo tradicional de aprendizado de máquina; à medida que o volume de dados aumenta, a estrutura da rede Bayesiana é mantida e apenas os parâmetros numéricos vão sendo atualizados. Assim, é possível atualizar o conhecimento armazenado na base de conhecimento sem um grande esforço computacional. Esta abordagem é adequada em domínios de aplicação onde inicialmente existe uma grande quantidade de dados disponível. Neste sentido, tomando-se como base uma grande amostra de dados para se

construir a estrutura da rede Bayesiana, pode-se, com maior probabilidade, induzir uma rede inicial que representa de maneira adequada a distribuição de probabilidade que rege o comportamento das variáveis do problema em questão. Assim, a atualização dos parâmetros numéricos tende a ser suficiente para ajustar a rede às pequenas variações presentes nos novos dados.

Ao invés de se considerar sempre todos os dados disponíveis para a atualização dos parâmetros numéricos, a cada nova execução podem ser guardados apenas os dados relevantes para o aprendizado de parâmetros das próximas execuções. Entretanto, na implementação atual, não estão sendo salvos apenas dados relevantes. Neste sentido, o conjunto de dados completo está sendo utilizado. Na proposta final (algoritmo ABC – veja Capítulo 6), entretanto, serão salvas apenas as estatísticas relevantes ao aprendizado, levando-se em conta a estrutura da rede. Então para uma variável A, guarda-se os dados relativos à variável A e às suas variáveis pais. Pode-se guardar esses dados numa estrutura como a AD-tree [34].

Na implementação atual, quando o algoritmo recebe novos dados, a partir da segunda execução, ele carrega os dados armazenados nas execuções anteriores na memória, e os utiliza, em conjunto com os novos dados da execução atual, para atualizar os parâmetros da rede Bayesiana.

O Algoritmo 5 apresenta uma descrição do algoritmo AIP em pseudocódigo:

```
1. Algoritmo AIP
2. Se for a primeira execução então
3.     Aprende_estrutura();
4. Senão
5.     Carrega_dados_execuções_anteriores();
6. Fim se
7. Aprende_parâmetros();
8. Grava_arquivo_rede();
9. Guarda_dados_parâmetros_em_arquivo();
```

Algoritmo 5: Algoritmo AIP

Para a implementação do primeiro protótipo do AIP foi utilizado o algoritmo K2 [11] de aprendizado de estrutura e de parâmetros de rede Bayesiana. Desta forma, os métodos Aprende_estrutura() e Aprende_parâmetros() são os métodos para aprendizado de estrutura e aprendizado de parâmetros do K2.

O AIP foi testado empiricamente em problemas de classificação utilizando-se quatro bases de dados: Alarm [2], com 37 variáveis e 30000 casos, Asia [31], com 8 variáveis e 15000 casos, Credit [14], com 12 variáveis e 15000 casos, e Engine fuel system [14], com 9 variáveis e 15000 casos. Nessas bases foram testados vários cenários, que serão descritos na próxima seção.

A base de dados Alarm [2] armazena dados sobre uma UTI clínica e o objetivo é armazenar casos onde a UTI deve ser alarmada ou não. A base de dados Asia [31] armazena dados sobre casos de tuberculose e câncer de pulmão e tenta estabelecer uma relação entre essas doenças e uma visita à Ásia. A base de dados Credit [14] armazena dados sobre perfis de pessoas que podem ou não receber crédito do banco. A base Engine Fuel System [14] armazena dados sobre diagnósticos em motores.

5.1. Simulações e Resultados com o AIP

Os experimentos realizados com o algoritmo proposto (AIP) tiveram como objetivo avaliar o índice de classificação correta (ICC) das redes aprendidas e o tempo computacional gasto pelo método para fazer o aprendizado da rede Bayesiana. Vale notar que o tempo computacional não é uma medida de extrema precisão, mas pode ser útil para ilustrar o esforço computacional exigido pelos algoritmos. Todos os experimentos realizados neste trabalho utilizaram um computador pessoal com processador Pentium 4 HT 3.2 MHz, com 1024 MB de memória RAM. Este equipamento não executou nenhum outro aplicativo durante a execução dos experimentos. Os experimentos têm como objetivo:

- Verificar se um conjunto inicial com poucos casos influi negativamente no desempenho da rede Bayesiana[4] quando utilizada para classificação;
- Verificar a diferença no tempo de execução entre a execução do K2 em batelada e do AIP.
- Verificar qual o impacto da atualização dos parâmetros no desempenho das redes Bayesianas, quando usadas para classificação, em comparação com o desempenho da rede Bayesiana, gerada através do K2 em batelada, usada para classificação.

Nos testes foram usados subconjuntos de dados selecionados aleatoriamente de cada base de dados, e foi mantida, nestes subconjuntos, a distribuição de probabilidade da base original para a variável classe. Além disso, como os métodos de aprendizado incremental, em geral, tendem a ser sensíveis à ordem da chegada dos dados, todos os cenários de todas as bases de dados foram gerados aleatoriamente 10 vezes e os resultados mostrados nas tabelas (disponíveis no Apêndice I) mostram a média e o desvio padrão das 10 execuções. Para uma análise de todos os resultados (todas as 10 tabelas para cada cenário de cada base de dados acesse http://www.dc.ufscar.br/~estevam/AIP.html).

Pode-se notar que os cenários, mostrados na Tabela 2 abaixo, foram projetados para evidenciar o quanto o tamanho do conjunto de dados inicial influencia na média do índice de classificação correta total. O primeiro cenário relativo a cada base de dados é de aprendizado não incremental (batelada).

[4] É importante notar que, assim como nos outros trabalhos sobre aprendizado incremental de redes Bayesianas já citados neste texto, as bases de dados aqui consideradas não possuem influência temporal na distribuição de suas variáveis, ou seja, a distribuição de um conjunto de dados inicial deve ser a mesma presente em qualquer um dos novos dados que sejam fornecidos.

Tabela 2: Tabela dos cenários utilizados nos experimentos.
CI: Conjunto Inicial de dados, NC500: Numero de Conjuntos de aprendizado incremental de 500 registros.

Tabela de cenários								
Base de dados	Alarm		Ásia		Credit		Engine Fuel System	
Tamanho da base de dados	30.000		15.000		15.000		15.000	
Tamanho do conjunto de teste	10.000		5.000		5.000		5.000	
	CI	NC500	CI	NC500	CI	NC500	CI	NC500
Primeiro cenário	20.000	0	10.000	0	10.000	0	10.000	0
Segundo cenário	10.000	20	5.000	10	5.000	10	5.000	10
Terceiro cenário	5.000	30	3.000	14	3.000	14	3.000	14
Quarto cenário	3.000	34	1.000	18	1.000	18	1.000	18
Quinto cenário	1.000	38	-	-	-	-	-	-

Os cenários foram testados da seguinte forma:

- Para o primeiro cenário de cada base de dados, foi feito somente o aprendizado da estrutura da rede Bayesiana e de seus parâmetros numéricos em batelada utilizando o conjunto inicial de dados. Em seguida a rede Bayesiana gerada foi usada para classificar o conjunto de teste, obtendo o índice de classificação correta (ICC). A média das 10 execuções de cada cenário forma a média do índice de classificação correta (MICC) para aquele cenário.
- Para os outros cenários de cada base de dados, usa-se o conjunto inicial de dados para fazer o aprendizado da estrutura da rede Bayesiana e de seus parâmetros numéricos. A partir daí usa-se o AIP para atualizar os parâmetros numéricos da rede Bayesiana gerada, utilizando os conjuntos de aprendizado incremental, um de cada vez. Neste sentido, os conjuntos de aprendizado incremental simulam dados novos que vão se tornando disponíveis à medida que o tempo passa.

- Para fazer a comparação justa com o aprendizado em batelada, a cada nova execução do AIP também é executado o aprendizado em batelada com os dados disponíveis até aquele momento. Ou seja, quando o AIP processar o quarto conjunto de aprendizado incremental, por exemplo, o aprendizado em batelada é executado usando o conjunto inicial de dados mais os quatro conjuntos de aprendizado incremental já processados pelo AIP. A cada nova execução do AIP ou do aprendizado em batelada a rede Bayesiana resultante é usada para classificar o conjunto de teste, obtendo assim o índice de classificação correta (ICC). A média das 10 execuções de cada cenário forma a média do índice de classificação correta (MICC) para aquele cenário.
- A cada cenário testado faz-se a média dos MICC, obtendo a média do índice de classificação total (MICCT). Cada medida, tanto de tempo quanto de MICC, vem acompanhado de seu respectivo desvio padrão.
- Para cada execução de cada cenário se utiliza um conjunto de teste, e esse conjunto é classificado utilizando a rede Bayesiana gerada pelo aprendizado em batelada e pela rede Bayesiana gerada pelo AIP, se possível. Esses conjuntos de teste são gerados aleatoriamente, então ao fim das 10 execuções de cada cenário, quando se obtém o MICC, não necessariamente o MICC final do aprendizado em batelada de cada cenário será igual. Seria igual se os conjuntos de teste de todos os cenários fossem iguais, mas não se pode garantir isso.

Foram usados os seguintes cenários de teste:

Alarm – variável classificada: Catechol

- Primeiro cenário (somente batelada) – Tabela 15
    - 20.000 casos (aprendizado de estrutura e parâmetros)
    - 10.000 casos (conjunto de teste)

- Segundo cenário (incremental e batelada) – Tabela 16
    - 10.000 casos (aprendizado de estrutura e parâmetros)
    - 20 conjuntos de 500 casos (aprendizado de parâmetros)
    - 10.000 casos (conjunto de teste)

- Terceiro cenário (incremental) – Tabela 17
    - 5.000 casos (aprendizado de estrutura e parâmetros)
    - 30 conjuntos de 500 casos (aprendizado de parâmetros)
    - 10.000 casos (conjunto de teste)

- Quarto cenário (incremental) – Tabela 18
    - 3.000 casos (aprendizado de estrutura e parâmetros)
    - 34 conjuntos de 500 casos (aprendizado de parâmetros)
    - 10.000 casos (conjunto de teste)

- Quinto cenário (incremental) – Tabela 19
    - 1.000 casos (aprendizado de estrutura e parâmetros)
    - 38 conjuntos de 500 casos (aprendizado de parâmetros)
    - 10.000 casos (conjunto de teste)

A Tabela 3 a seguir mostra um resumo dos experimentos feitos com a base Alarm.

Tabela 3: Resumo cenários – Base de dados Alarm.
MICCT: Média do Índice de Classificação Correta Total; T: Tempo médio, em segundos (s), e DP: Desvio Padrão.

Base de Dados Alarm				
	Incremental		Batelada	
Cenário	MICCT±DP	T±DP	MICCT±DP	T±DP
Conjunto inicial 20000 casos	-	-	96,97 ± 0,13	149,43 ± 0,90
Conjunto inicial 10000 casos	90,80 ± 9,22	99,71 ± 0,80	90,36 ± 9,36	2.414,54 ± 115,50
Conjunto inicial 5000 casos	90,80 ± 9,22	67,46 ± 1,19	93,01 ± 7,62	3.049,63 ± 436,69
Conjunto inicial 3000 casos	90,59 ± 9,25	54,36 ± 1,61	94,37 ± 6,23	3.260,28 ± 70,75
Conjunto inicial 1000 casos	88,60 ± 9,12	39,16 ± 2,76	89,86 ± 2,30	2.892,09 ± 1,57

Os experimentos feitos com a base Alarm [2] evidenciaram o ganho de desempenho, relativo ao tempo computacional gasto, obtido ao se fazer o aprendizado incremental, mesmo sendo ele ingênuo (como ocorre no AIP). O tempo necessário para o cenário com o menor conjunto de aprendizado de estrutura (1000 casos, quinto cenário) foi significativamente menor do que o tempo necessário para se fazer o mesmo cenário usando o aprendizado em batelada. Em números, o tempo de processamento do quinto cenário usando o aprendizado em batelada foi aproximadamente 74 vezes maior do que o tempo de processamento do quinto cenário usando o AIP. A diferença entre o tempo de processamento de cada cenário usando o aprendizado em batelada e o AIP diminuiu para aproximadamente 24 vezes no segundo cenário. O tempo de processamento do cenário usando AIP diminuiu progressivamente com a diminuição do conjunto de aprendizado de estrutura (conjunto inicial).

Quanto aos índices de classificação correta obtidos nos cenários, fica evidente a dependência do AIP em relação ao aprendizado de estrutura da rede Bayesiana feito usando o conjunto de aprendizado. Dentro de cada cenário, nas execuções em que houve um bom aprendizado de estrutura o AIP conseguiu manter o bom ICC. Já nas execuções em que esse aprendizado não foi bom, o AIP não conseguiu melhorar significativamente o ICC. Analisando o aprendizado em batelada, percebe-se que os conjuntos de dados que geram ICC ruim no AIP também geram ICC ruim no aprendizado em batelada. Porém ao longo do tempo, na medida em que mais dados ficam disponíveis ao aprendizado em batelada, o ICC melhora, ficando uniformemente bom no final do cenário.

De modo geral, percebe-se que há uma troca de desempenho em relação ao tempo computacional por ICC. O ICC dos cenários utilizando o AIP varia em torno do ICC obtido com o aprendizado em batelada usando o conjunto de aprendizado, com tempo computacional pequeno. Portanto percebe-se que o AIP é altamente dependente da estrutura da rede Bayesiana. Já o ICC dos cenários utilizando o aprendizado em batelada é melhor, porém com tempo computacional bem maior.

Asia – base original – 15.000 casos – variável classificada: Tuberculosis Or Cancer

- Primeiro cenário (batelada) – Tabela 20
    - 10.000 casos (aprendizado de estrutura e parâmetros)
    - 5.000 casos (conjunto de teste)

- Segundo cenário (incremental) – Tabela 21
    - 5.000 casos (aprendizado de estrutura e parâmetros)
    - 10 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

- Terceiro cenário (incremental) – Tabela 22
    - 3.000 casos (aprendizado de estrutura e parâmetros)
    - 14 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

- Quarto cenário (incremental) – Tabela 23
    - 1.000 casos (aprendizado de estrutura e parâmetros)
    - 18 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

A Tabela 4 a seguir mostra um resumo dos experimentos feitos com a base Asia.

Tabela 4: Resumo cenários – Base de dados Ásia.
MICCT: Média do índice de Classificação correta Total; T: Tempo médio, em segundos (s), e DP: Desvio Padrão.

Base de Dados Asia				
	Incremental		Batelada	
Cenário	MICCT±DP	T±DP	MICCT±DP	T±DP
Conjunto inicial 10000 casos	-	-	99,98 ± 0,00	0,50 ± 0,18
Conjunto inicial 5000 casos	99,98 ± 0,01	1,57 ± 0,54	99,98 ± 0,01	4,10 ± 0,38
Conjunto inicial 3000 casos	99,98 ± 0,01	1,51 ± 0,46	99,98 ± 0,01	4,84 ± 0,41
Conjunto inicial 1000 casos	99,97 ± 0,02	2,17 ± 1,11	99,97 ± 0,02	4,97 ± 1,08

Os experimentos realizados com a base de dados Asia [31] foram bem sucedidos em relação aos índices de classificação correta obtidos. O MICC se manteve estável ao longo dos cenários, mesmo usando conjuntos de aprendizado de tamanho diferente para aprendizado de estrutura de rede Bayesiana.

Quanto ao tempo de processamento, o tempo de processamento obtido usando o AIP foi inferior ao tempo de processamento obtido usando o aprendizado em batelada. Não foi uma diferença tão grande como na base Alarm, mas isso decorre do fato do tamanho da base de dados ser menor, tanto no número de casos quanto no número de variáveis.

Os experimentos com a base de dados Ásia foram favoráveis ao AIP, já que o tempo de processamento obtido com o AIP foi menor, com MICC equivalente ao obtido com o aprendizado em batelada.

Credit – base original – 15.000 casos – variável classificada: Credit Worthiness

- Primeiro cenário (batelada) – Tabela 24
  - 10.000 casos (aprendizado de estrutura e parâmetros)
  - 5.000 casos (conjunto de teste)

- Segundo cenário (incremental) – Tabela 25
  - 5.000 casos (aprendizado de estrutura e parâmetros)
  - 10 conjuntos de 500 casos (aprendizado de parâmetros)
  - 5.000 casos (conjunto de teste)

- Terceiro cenário (incremental) – Tabela 26
    - 3.000 casos (aprendizado de estrutura e parâmetros)
    - 14 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

- Quarto cenário (incremental) – Tabela 27
    - 1.000 casos (aprendizado de estrutura e parâmetros)
    - 18 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

A Tabela 5 abaixo mostra um resumo dos experimentos feitos com a base Credit.

Tabela 5: Resumo cenários – Base de dados Credit.
MICCT: Média do índice de Classificação correta Total; T: Tempo médio, em segundos (s), e DP: Desvio Padrão.

Base de Dados Credit				
	Incremental		Batelada	
Cenário	MICCT±DP	T±DP	MICCT±DP	T±DP
Conjunto inicial 10000 casos	-	-	65,34 ± 0,75	3,51 ± 0,06
Conjunto inicial 5000 casos	65,58 ± 0,76	4,36 ± 0,78	65,58 ± 0,76	25,81 ± 0,69
Conjunto inicial 3000 casos	68,16 ± 4,33	3,73 ± 1,13	65,28 ± 1,91	30,69 ± 1,01
Conjunto inicial 1000 casos	73,15 ± 0,04	3,43 ± 0,95	67,69 ± 3,20	32,50 ± 0,89

De todas as bases, a que teve resultados mais favoráveis ao AIP foi a base Credit [14]. O MICC dos cenários usando o AIP aumentou à medida que o tamanho do conjunto de aprendizado diminuiu, enquanto o tempo computacional dos cenários usando o AIP diminuiu à medida que o tamanho do conjunto de aprendizado diminuiu.

O interessante é que o AIP conseguiu manter o ICC obtido com o aprendizado em batelada usando o conjunto de aprendizado ao longo de cada cenário. O mesmo não ocorreu nos cenários quando foi utilizado o aprendizado em batelada. Neles o ICC diminuiu à medida que mais casos ficaram disponíveis

para aprendizado. Isto pode significar a presença de ruídos nos novos dados que foram chegando para a realização do aprendizado.

Engine Fuel System – base original – 15.000 casos – variável classificada: Fuel Filters

- Primeiro cenário (batelada) – Tabela 28
    - 10.000 casos (aprendizado de estrutura e parâmetros)
    - 5.000 casos (conjunto de teste)

- Segundo cenário (incremental) – Tabela 29
    - 5.000 casos (aprendizado de estrutura e parâmetros)
    - 10 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

- Terceiro cenário (incremental) – Tabela 30
    - 3.000 casos (aprendizado de estrutura e parâmetros)
    - 14 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

- Quarto cenário (incremental) – Tabela 31
    - 1.000 casos (aprendizado de estrutura e parâmetros)
    - 18 conjuntos de 500 casos (aprendizado de parâmetros)
    - 5.000 casos (conjunto de teste)

A Tabela 6 abaixo mostra um resumo dos experimentos feitos com a base Engine Fuel System.

Tabela 6: Resumo cenários – Base de dados Engine Fuel System
MICCT: Média do índice de Classificação correta Total; T: Tempo médio, em segundos (s), e DP: Desvio Padrão.

Base de Dados Engine Fuel System				
	Incremental		Batelada	
Cenário	MICCT±DP	T±DP	MICCT±DP	T±DP
Conjunto inicial 10000 casos	-	-	99,66 ± 0,01	0,48 ± 0,10
Conjunto inicial 5000 casos	99,57 ± 0,03	1,96 ± 0,52	99,57 ± 0,03	4,57 ± 0,58
Conjunto inicial 3000 casos	99,62 ± 0,03	2,35 ± 0,99	99,62 ± 0,03	5,60 ± 0,74
Conjunto inicial 1000 casos	99,63 ± 0,02	2,47 ± 0,78	99,63 ± 0,03	5,80 ± 0,62

Os experimentos realizados com a base de dados Engine fuel system [14] apresentou um comportamento equivalente aos experimentos feitos com a base Asia. Ou seja, os índices de classificação nos cenários utilizando o AIP se mantiveram relativamente constantes e o desempenho do aprendizado incremental, relativo ao tempo computacional, foi melhor ao longo dos cenários quando o AIP foi utilizado.

5.2. Discussão sobre os resultados obtidos com o AIP

Em relação à discussão sobre a influência dos parâmetros numéricos na obtenção do ICC, apresentada em [15] e na Seção 2.4, nos experimentos realizados neste trabalho se constata que cada base tem o seu nível de sensibilidade à atualização dos parâmetros numéricos da rede Bayesiana. Entre as quatro bases, a base Engine fuel system tem a menor sensibilidade à atualização dos parâmetros numéricos, enquanto a base Alarm tem a maior sensibilidade. Este comportamento pode estar sendo influenciado também pelo tamanho do conjunto de dados inicial (utilizado para a construção da estrutura da rede Bayesiana), pois, a base Alarm possui 37 variáveis (que podem assumir 2 ou mais valores) e assim, existem mais de 2^{37} instâncias (registros na base de dados) possíveis para esta base, ou seja, uma amostra de 10.000 instâncias não é significativa para esta base. Já para uma base menor como a Engine (com apenas 9 variáveis binárias) uma amostra com 10.000 instâncias é bem mais

significativa. Assim, o comportamento similar ocorrido com as duas menores bases (Ásia e Engine Fuel System) pode ter ocorrido por este motivo.

Os resultados apresentados na seção anterior podem ser utilizados para uma interpretação inicial sobre o comportamento do método AIP proposto (principalmente quando se considera o aspecto de ICCs). Assim, a comparação justa sobre o tempo computacional dentro de cada cenário é feita com a execução de cada cenário usando o AIP e o aprendizado em batelada paralelamente. Para facilitar a interpretação desta argumentação, imagine-se o segundo cenário da base de dados Alarm como descrito abaixo:

- 10.000 casos (aprendizado de estrutura e parâmetros)
- 20 conjuntos de 500 casos (aprendizado de parâmetros)
- 10.000 casos (conjunto de teste)

Este cenário está simulando uma situação em que se tem 10.000 casos num momento t1 que podem ser utilizados para o aprendizado de estrutura. Com o passar do tempo, vão chagando novos dados. Assim, num momento t2 chegam mais 500 casos, em t3 chegam mais 500 casos e assim por diante até se atingir um momento t21 onde chegam os últimos 500 casos. Quando não se tem um mecanismo de aprendizado incremental, o procedimento para a incorporação dos novos casos ao modelo classificador é a execução do algoritmo de batelada a cada novo conjunto de dados que chega e isto torna o processo muito custoso computacionalmente.

A observação que pode ser feita da comparação entre os cenários é que o aprendizado incremental de redes Bayesianas se mostra promissor. Em relação ao tempo necessário para se fazer o aprendizado de todos os dados de um cenário específico, pode-se constatar uma relação inversa entre o número de variáveis da base de dados e a porcentagem de tempo necessário para a execução do aprendizado inicial, ou seja, quanto mais variáveis a base de dados tiver, maior será o tempo gasto para gerar a RB (estrutura e parâmetros numéricos) utilizando o conjunto de dados inicial.

Assim, considerando-se o tempo total do processo o tempo necessário para se fazer o aprendizado incremental será proporcionalmente menor. Isso é evidente se for levado em conta a complexidade de tempo do algoritmo K2 especificado em [11], O ($m\ u^2\ n^2\ r$), m sendo o número de registros na base de dados, u o limite de pais que uma variável possa ter, n o número de variáveis na

base de dados e r o tamanho do domínio de cada variável na base de dados. Ou seja, quando se tem mais variáveis na base de dados (n^2) o custo computacional do algoritmo será maior, aumentando a relevância do fator m. Assim sendo, ao diminuir m e dividir o restante de m em conjuntos de 500 casos, que tem um custo computacional baixo, se terá um ganho em performance em relação ao conjunto original de dados. O inverso é válido, ou seja, em uma base com número pequeno de variáveis, o ganho de performance tende a ser menor.

Quanto à classificação utilizando as redes Bayesianas aprendidas incrementalmente, pode-se constatar que a diminuição do conjunto de dados usado para aprendizado de estrutura pode, em alguns casos, ter efeitos benéficos. Na base Credit, por exemplo, a diminuição do conjunto de dados usado para aprendizado de estrutura foi benéfico. Uma possível explicação para isso pode ser que, ao diminuir os dados disponíveis para aprendizado de estrutura há uma tendência em se identificar os relacionamentos mais fortes entre variáveis, deixando alguns relacionamentos menos relevantes de fora da estrutura da rede Bayesiana. Por exemplo, a diferença entre a rede Bayesiana do primeiro cenário (R1) da base Credit e a rede Bayesiana do quarto cenário (R4) da base Credit é que em R4 não há a relação entre os nós Age e Credit Worthiness, ou seja, no aprendizado da rede Bayesiana R4 não se detectou no conjunto de aprendizado a relação entre essas variáveis, o que resultou na melhora na classificação de dados já mencionada. Mas, é importante ressaltar que, caso fosse possível se trabalhar com um conjunto de dados suficientemente grande (2.000.000, por exemplo) como conjunto inicial, dificilmente um conjunto menor (com 1.000, por exemplo) geraria uma estrutura de rede mais adequada. Ou seja, este comportamento ocorrido com a base Credit não pode ser generalizado e deve ser atribuído à amostra não significativa utilizada nos experimentos.

Já na base Alarm a diminuição do conjunto de dados usado para aprendizado de estrutura resultou, em alguns casos, em índices de classificação correta muito inferiores aos obtidos através de aprendizado de estrutura em batelada, o que teoricamente deveria ser a tendência natural.

Quando se analisa o impacto do aprendizado incremental de parâmetros nos cenários, pode-se dizer que não houve um comportamento uniforme em relação a isso. Houve cenários em que não houve variação no ICC, cenários em que houve

uma variação pequena e não freqüente em torno do ICC inicial e, finalmente, houve cenários em que o índice de classificação correta variou bastante.

Também não é possível fazer uma correlação entre o comportamento do cenário e a base de dados utilizada. Cenários utilizando a mesma base tiveram comportamentos diferentes. Isto mostra que, como a maioria dos métodos de aprendizado incremental, o AIP é sensível à ordem de chegada dos dados e ao tamanho do conjunto inicial. Os experimentos realizados para a análise empírica do método ABC (capítulo 6) permitirão uma conclusão mais consistente com relação à influência do tamanho do conjunto inicial, pois as bases terão um número maior de registros.

Portanto pode-se concluir que os resultados revelaram duas características do AIP. A primeira é que o impacto da atualização dos parâmetros numéricos não pode ser desconsiderado, pois se a base de dados for sensível à alteração dos parâmetros numéricos da rede Bayesiana, então há chance real de que a atualização dos parâmetros numéricos resulte em melhoria do ICC. A segunda é que o AIP tende a reduzir consideravelmente o tempo computacional exigido pelo processo de aprendizado.

Capítulo 6. Proposta de Algoritmo de Aprendizado Bayesiano em Camadas – ABC

A proposta de algoritmo de aprendizado incremental de parâmetros de rede Bayesiana – AIP (descrito no Capítulo 5) foi o primeiro método desenvolvido neste trabalho e foi muito útil permitindo explorar características do aprendizado incremental e fazer experimentos para verificar a influência dos parâmetros numéricos no conhecimento armazenado em uma rede Bayesiana.

A principal contribuição deste trabalho de mestrado, entretanto, é a proposta de um algoritmo mais completo de aprendizado incremental de redes Bayesianas chamado ABC (Aprendizado Bayesiano em Camadas). Nesse capítulo é detalhado este algoritmo.

O algoritmo ABC foi desenvolvido a partir dos conhecimentos adquiridos durante o desenvolvimento do algoritmo AIP e dos conhecimentos adquiridos durante a revisão bibliográfica feita durante o desenvolvimento do trabalho de pesquisa. O cerne do ABC é o aprendizado incremental de parâmetros numéricos e a terceira estratégia proposta por Friedman e Goldszmidt, a Incremental [18]. Da estratégia Incremental se aproveita a idéia de se utilizar um conjunto de redes vizinhas, formado a partir de alterações na rede Bayesiana inicial.

Como visto na Seção 4.1, o conceito que sustenta a idéia de gerar o conjunto de redes vizinhas é que se uma rede Bayesiana é adequada para representar um conjunto de dados, e assumindo que esses dados seguem uma distribuição de probabilidade qualquer, quando chegarem dados novos, seguindo a mesma distribuição de probabilidade, se essa rede não for mais adequada então a rede mais adequada estará próxima da estrutura da rede atual [18] e [40].

Já o conceito de aprendizado incremental de parâmetros numéricos se sustenta no conhecimento adquirido durante o desenvolvimento do algoritmo AIP. Durante esse período se identificou que o processo de aprendizado incremental de parâmetros numéricos é um processo computacionalmente leve e o eventual benefício de se fazer a atualização dos parâmetros numéricos não pode ser descartado, já que a ocorrência ou não desse benefício é dependente da base de dados e de sua sensibilidade à mudança dos parâmetros numéricos.

A idéia de camadas surgiu para que o processo completo de aprendizado da rede Bayesiana não seja efetuado sempre que novos dados estejam disponíveis. Em outras palavras, o ABC posterga a execução do aprendizado da estrutura da rede Bayesiana enquanto outras técnicas podem ser utilizadas para atualizar o conhecimento presente na rede. Esta idéia depende de se definir uma função de aptidão para medir a qualidade da rede Bayesiana gerada. Essa função é usada juntamente com um limiar de qualidade aceitável. Quando a função de aptidão de uma rede Bayesiana retornar um valor abaixo desse limiar então muda-se de camada. Cada camada tem uma ação específica de atualização do conhecimento, sendo que quanto maior a camada, maior é a complexidade da ação feita nessa camada.

A função de aptidão escolhida para medir a qualidade da rede Bayesiana gerada foi o índice de classificação correta (ICC) obtido quando se classifica o conjunto de teste usando a rede Bayesiana gerada. É uma função interessante, no sentido de que se usa o objetivo final, que é o ICC da rede, como guia do aprendizado da rede Bayesiana. Outra opção interessante seria usar o score Bayesiano da rede, obtido através da função g do algoritmo K2, por exemplo. Nesse caso o guia do aprendizado seria a qualidade do processo de aprendizado. A escolha do ICC como variável para medir a qualidade da rede Bayesiana gerada também levou em conta o contexto em que se quer aplicar este algoritmo, que é a Mineração de Dados.

No total o ABC possui cinco camadas. O ponto de entrada é a primeira camada, que é a camada com complexidade menor. O ponto de saída do algoritmo é a primeira camada ou a quinta camada, que é a camada com a complexidade maior. O fato do algoritmo sair na primeira camada indica sucesso, ou seja, a rede Bayesiana gerada apresentou ICC maior que o limite estabelecido, e se o algoritmo sair na quinta camada indica falha, indica que o algoritmo não conseguiu deixar a rede Bayesiana com ICC maior que o limite estabelecido.

A Figura 10 a seguir mostra o fluxograma do algoritmo ABC, usando o ICC como variável para medir a qualidade da rede Bayesiana gerada. A descrição da ação de cada camada virá após a figura.

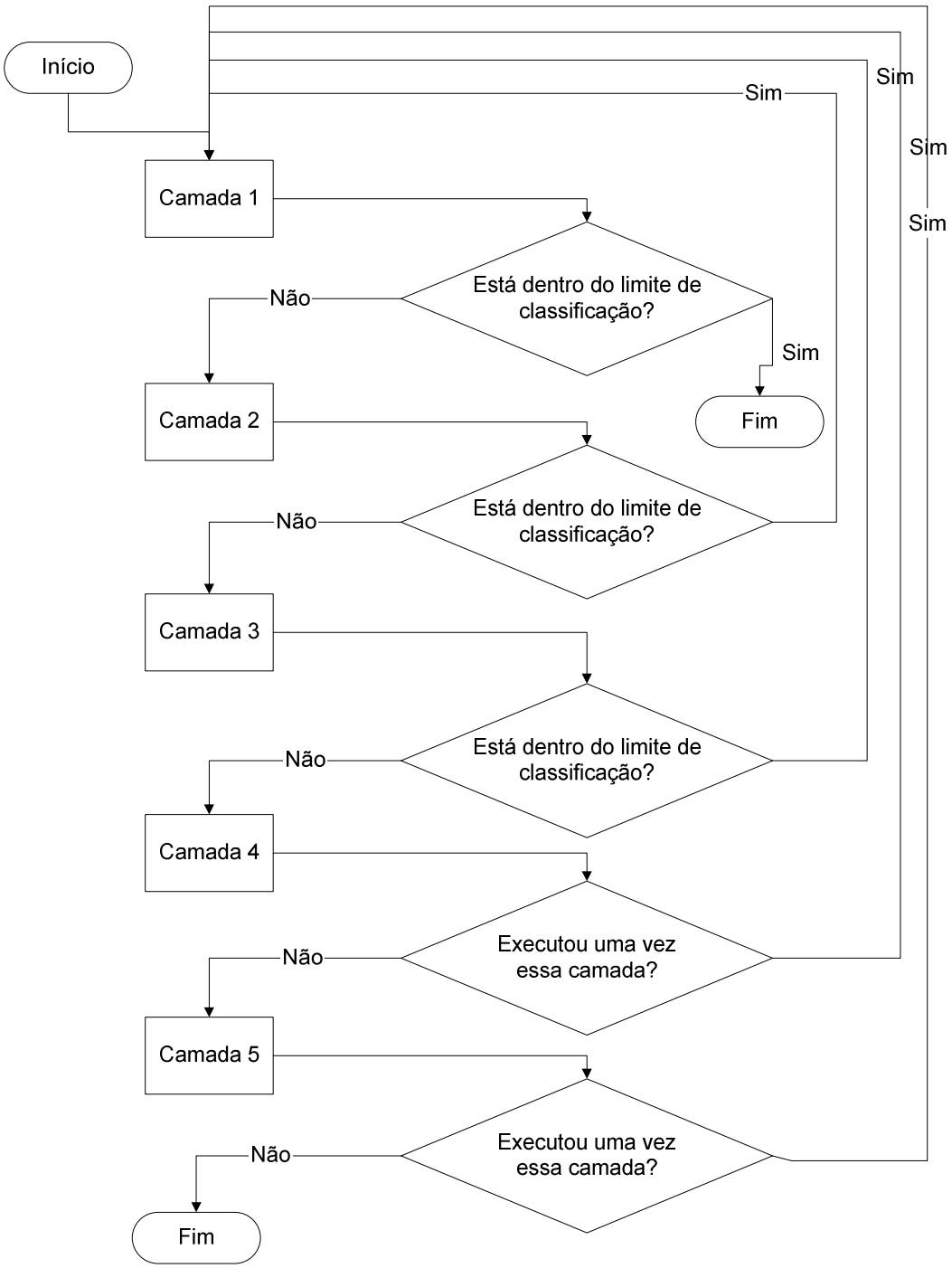

Figura 10: Fluxograma do algoritmo ABC.

Antes de descrever o funcionamento de cada camada são necessárias algumas definições:

- Os dados usados para aprendizado de estrutura ou de parâmetros numéricos de rede Bayesiana serão armazenados em uma AD-Tree [34].

- Existem quatro conjuntos de dados ao longo do algoritmo:
    - Conjunto de dados de entrada: conjunto de dados de entrada do algoritmo. A partir dos dados deste conjunto serão criados os outros conjuntos de dados descritos abaixo.
    - Conjunto de aprendizado: no início do algoritmo é o conjunto de dados que será usado para fazer o aprendizado da rede Bayesiana original. Após o início do algoritmo é o conjunto de dados que já foi usado em algum aprendizado da rede Bayesiana e que poderá ser usado novamente para fazer o aprendizado da rede Bayesiana se necessário.
    - Conjunto de atualização: conjunto de dados que será usado para fazer a atualização da rede Bayesiana. Depois que a atualização da rede Bayesiana for feita, os dados do conjunto de atualização são inseridos no conjunto de aprendizado.
    - Conjunto de teste: conjunto de dados utilizado para classificar a rede Bayesiana gerada.
- Existe uma AD-Tree ao longo da execução do algoritmo. Quando for necessário o aprendizado de estrutura e parâmetros numéricos da rede Bayesiana a AD-Tree armazenará o conjunto de aprendizado. Quando for necessária apenas a atualização dos parâmetros numéricos, a AD-Tree armazenará o conjunto de atualização.
- O conjunto de aprendizado é armazenado em disco[5] quando não é utilizado.
- O classificador utilizado para obter o ICC foi desenvolvido a partir da biblioteca de funções SMILE [14], de acordo com o Algoritmo 4 apresentado na Seção 2.4.2.
- O conjunto de redes vizinhas é o conjunto de redes Bayesianas gerado a partir de uma rede Bayesiana de referência. Para gerar o conjunto de redes vizinhas se aplica um operador à rede Bayesiana de referência. Existem três operadores possíveis: adicionar arco, remover arco e inverter arco. Não é utilizado neste trabalho a inserção ou remoção de variáveis da rede Bayesiana como operadores.

[5] Armazenar em disco refere-se, neste trabalho, a armazenar os dados em algum dispositivo de armazenagem física de dados.

- Como há sempre mais de uma rede Bayesiana na memória durante a execução do algoritmo, então é necessário definir a rede Bayesiana oficial, que será aquela a partir da qual será gerado o conjunto de redes vizinhas. No início do algoritmo, a rede oficial é a rede original, a primeira aprendida. Mas no decorrer do algoritmo ela pode ser outra, e então o conjunto de redes vizinhas pode ser gerado a partir de uma rede que é vizinha à original, se distanciando assim da rede original.

Segue a descrição do funcionamento de cada camada.

Primeira camada do algoritmo ABC

É a camada inicial. Se for a primeira execução, então os dados que chegarem a essa camada serão divididos entre conjunto de aprendizado e conjunto de teste. O conjunto de aprendizado será usado para fazer o aprendizado de estrutura e parâmetros numéricos da rede Bayesiana original.

Se não for a primeira execução então os dados que chegarem a essa camada serão inseridos no conjunto de teste.

Independentemente de qual execução for, o conjunto de teste será classificado usando a rede Bayesiana oficial. Se o ICC obtido com esta classificação ficar acima do limiar estabelecido, então o algoritmo é encerrado e a rede Bayesiana oficial é retornada como classificador, senão o algoritmo passa para a segunda camada.

Segue o pseudocódigo da primeira camada descrito pelo Algoritmo 6.

```
1.  Algoritmo camada1;
2.  {Entrada: Um conjunto de dados novos cjDados; a rede Bayesiana oficial;
    o conjunto de redes vizinhas; o conjunto de teste }
3.  se primeiraExecução então
4.     dividirAmostra(cjDados);
5.     aprendizadoRede();
6.  senão
7.     adicionarCjtoTeste(cjDados);
8.  fim {se};
9.  ICC := classificar();
10. se ICC > limiteClassificação então
11.    retorna sucesso;
12. senão
13.    retorna camada2();
14. fim {se};
```

Algoritmo 6: Camada 1 do algoritmo ABC.

Segunda camada do algoritmo ABC

Vale lembrar, que a segunda camada do ABC só será executada caso a rede Bayesiana oficial não apresente um ICC adequado. Assim, o objetivo da segunda camada é obter uma rede Bayesiana melhor que a oficial sem precisar executar o algoritmo de aprendizado. Neste sentido, na segunda camada, o conjunto de teste é classificado utilizando todas as redes Bayesianas do conjunto de redes vizinhas, exceto a rede oficial, uma por vez. Cada rede Bayesiana utilizada gera um ICC.

Se encontrar uma rede com ICC acima do limiar, o algoritmo volta para a primeira camada, e a rede Bayesiana que gerou este ICC passa a ser a rede oficial. Se nenhum dos ICC ficar acima do limiar, então o algoritmo passa para a terceira camada.

Segue o pseudocódigo da segunda camada descrito pelo Algoritmo 7.

```
1. Algoritmo camada2;
2. {Entrada: a rede Bayesiana oficial; o conjunto de redes vizinhas; o
   conjunto de teste }
3. para todas as redes Bayesianas do conjunto de redes vizinhas, exceto a
   rede oficial, faça
4.    ICC := classificar();
5.    se ICC > limiteClassificação então
6.       retorna camada1();
7. fim {para};
8. retorna camada3();
```

Algoritmo 7: Camada 2 do algoritmo ABC.

Terceira camada do algoritmo ABC

A terceira camada só será executada caso a rede Bayesiana oficial não tenha gerado um ICC adequado (acima do limiar) na primeira camada, e a segunda camada não tenha conseguido uma rede vizinha que gerasse um ICC acima do limiar. Neste caso, a terceira camada é acionada na tentativa de se encontrar uma rede Bayesiana "melhor" do que as já analisadas e, isto é feito através da atualização dos parâmetros numéricos (como o executado pelo AIP) da rede oficial e de suas redes vizinhas. Assim, na terceira camada, uma porção do conjunto de teste é usada para formar o conjunto de atualização.

O conjunto de atualização é então utilizado para atualizar os parâmetros numéricos da rede Bayesiana oficial e das redes do conjunto de redes vizinhas. Após esse processo, é feita a classificação do novo conjunto de teste, usando a rede Bayesiana oficial e as redes do conjunto de redes vizinhas, devidamente atualizadas. A cada rede utilizada é gerado um novo ICC. Se algum dos ICC ficar acima do limiar, o algoritmo volta para a primeira camada, e a rede Bayesiana que gerou o ICC acima do limiar passa a ser a rede oficial. Caso contrário, o algoritmo passa à quarta camada.

Segue o pseudocódigo da terceira camada descrito pelo Algoritmo 8.

```
1. Algoritmo camada3;
2. {Entrada: a rede Bayesiana oficial; o conjunto de redes vizinhas; o
   conjunto de teste }
3. gerarConjuntoAtualização();
4. para todas as redes Bayesianas do conjunto de redes vizinhas faça
5.    atualizarParametros();
6.    ICC := classificar();
7.    se ICC > limiteClassificação então
8.       retorna camada1();
9. fim {para};
10. retorna camada4();
```

Algoritmo 8: camada 3 do algoritmo ABC.

Quarta camada do algoritmo ABC

Assim como ocorre nas camadas anteriores, a quarta camada só é acionada quando as camadas anteriores não foram capazes de encontrar uma rede Bayesiana satisfatória para a classificação. Assim, na quarta camada um novo conjunto de redes vizinhas é gerado a partir da rede oficial atual. Os parâmetros numéricos das redes Bayesianas do conjunto de redes vizinhas são estimados usando o conjunto de atualização.

Após esse processo o conjunto de atualização é inserido no conjunto de aprendizado, sendo salvo em disco, e o algoritmo retorna à primeira camada. Se o algoritmo voltar à quarta camada sem ter sucesso em manter o índice de classificação do conjunto de teste dentro do limite pré-definido, o algoritmo passa à quinta camada.

Segue o pseudocódigo da quarta camada descrito pelo Algoritmo 9.

```
1. Algoritmo camada4;
2. {Entrada: a rede Bayesiana oficial; o conjunto de redes vizinhas; o
   conjunto de teste; o conjunto de atualização }
3. se SegundaExecuçãoQuartaCamada então
4.    retorna camada5();
5. senão
6.    gerarConjuntoRedesVizinhas();
7.    para todas as redes Bayesianas do conjunto de redes vizinhas faça
8.       atualizarParametros();
9.    fim {para};
10.   inserirCjtoAtualizaçãoCjtoAprendizado();
11.   retorna camada1();
12. fim {se};
```

Algoritmo 9: Camada 4 do algoritmo ABC.

Quinta camada do algoritmo ABC

A quinta camada é a última camada do ABC e, quando ela é atingida, o ABC considera que as estratégias de atualização da rede Bayesiana oficial não promoveram o desempenho necessário, e assim, um algoritmo não-incremental terá que ser usado. Ou seja, quando o ABC não consegue atualizar a rede Bayesiana de maneira satisfatória utilizando técnicas mais simples (e menos custosas computacionalmente) presentes nas camadas anteriores, parte-se para uma execução de aprendizado não-incremental.

Desta forma, na quinta camada o conjunto de aprendizado é carregado na memória e é utilizado para fazer o aprendizado em batelada de estrutura e parâmetros numéricos da nova rede Bayesiana oficial. Após isso, gera-se novamente o conjunto de redes vizinhas a partir da rede oficial, utilizando o conjunto de aprendizado para estimar os parâmetros numéricos das redes Bayesianas do conjunto de redes vizinhas.

Após esse processo o conjunto de aprendizado é armazenado em disco e o algoritmo retorna à primeira camada.

Se o algoritmo voltar à quinta camada sem ter sucesso em manter o índice de classificação do conjunto de teste acima do limiar pré-definido, o algoritmo é terminado, para não entrar em um loop infinito.

Segue o pseudocódigo da quinta camada descrito pelo Algoritmo 10.

```
1. Algoritmo camada5;
2. {Entrada: a rede Bayesiana oficial; o conjunto de redes vizinhas; o
   conjunto de teste; o conjunto de atualização }
3. se SegundaExecuçãoQuintaCamada então
4.    retorna falha;
5. senão
6.    carregarCjtoAprendizado();
7.    aprendizadoRede();
8.    gerarConjuntoRedesVizinhas();
9.    para todas as redes Bayesianas do conjunto de redes vizinhas faça
10.      atualizarParametros();
11.   fim {para};
12.   retorna camada1();
13. fim {se};
```

Algoritmo 10: Camada 5 do algoritmo ABC.

Alguns métodos presentes nos algoritmos acima não foram ainda especificados. Eles serão especificados a seguir:

adicionarCjtoTeste()

Método que adiciona registros do conjunto de dados de entrada no conjunto de teste.

aprendizadoRede()

Método que faz o aprendizado de estrutura e parâmetros numéricos da rede Bayesiana. Na implementação do ABC foi usado o algoritmo K2 [11] para fazer o aprendizado. Note-se que aqui nesse método se faz o aprendizado em batelada da rede Bayesiana.

atualizarParametros()

Método que faz a atualização dos parâmetros numéricos da rede Bayesiana. É baseado no algoritmo AIP, descrito no capítulo 5.

carregarCjtoAprendizado()

Método que carrega o conjunto de aprendizado, salvo em disco, para a memória. Os dados armazenados no disco são inseridos na AD-Tree.

dividirAmostra()

Método que divide o conjunto de dados de entrada entre o conjunto de aprendizado e o conjunto de teste. Cada registro do conjunto de dados de entrada será inserido em um dos dois conjuntos.

classificar()

Método que faz a classificação do conjunto de teste usando a rede Bayesiana oficial ou alguma das redes Bayesianas do conjunto de redes vizinhas.

gerarConjuntoRedesVizinhas()

Método que gera as redes Bayesianas do conjunto de redes vizinhas a partir da rede Bayesiana oficial.

Essas redes Bayesianas são geradas a partir de operações aplicadas à rede Bayesiana oficial. Os operadores existentes são: adicionar arco, remover arco e inverter arco. Podem-se configurar quais desses operadores estarão disponíveis para gerar o conjunto de redes vizinhas. Neste trabalho não foram usados a adição ou remoção de nós da rede Bayesiana.

inserirCjtoAtualizaçãoCjtoAprendizado()

Método que insere o conjunto de atualização, que está na AD-Tree, em memória, e o insere no conjunto de aprendizado, em disco. Para fazer isso, carrega-se o conjunto de aprendizado na memória e se faz a junção dos dois conjuntos, e então se salva o novo conjunto de aprendizado em disco.

Como dito anteriormente, o ponto de entrada do algoritmo ABC é a primeira camada, e o ponto de saída é a primeira ou quinta camada. Existe um controle na quinta camada que impede que o algoritmo fique executando indefinidamente. Assim, se a quinta camada for executada duas vezes o algoritmo é encerrado. A expectativa é que o algoritmo não precise chegar muitas vezes à quinta camada, já que as camadas anteriores têm ações que podem ser suficientes para adaptar a rede Bayesiana aos novos dados. As situações em que a execução da quinta camada seria necessária são:

- O conjunto de dados inicial é pequeno, e seus dados não são representativos para o problema em questão. Nesse caso, o conjunto de aprendizado usado para fazer o primeiro aprendizado da rede Bayesiana oficial e das redes Bayesianas do conjunto de redes vizinhas não é adequado para fazer essa tarefa.
- Houve uma mudança grande de perfil do problema em questão. Por exemplo, a ocorrência de um terremoto em uma área em que se estuda a expectativa de vida. Nesse caso, o conjunto de teste teria uma distribuição de probabilidade diferente da distribuição de probabilidade do conjunto de aprendizado original.

Para os casos em que a distribuição de probabilidade do conjunto de teste é pouco diferente da distribuição de probabilidade do conjunto de aprendizado, as ações da primeira à quarta camada conseguem adaptar a rede Bayesiana atual aos dados novos.

Pode-se notar que o ABC é um método incremental de aprendizado incremental de redes Bayesianas. Neste sentido, à medida que se faz necessário, é incrementada uma etapa (camada) no processo de aprendizado incremental. Assim, busca-se procrastinar ao máximo a execução do algoritmo de batelada, e por conseqüência minimizar o esforço computacional do processo, sem permitir a deterioração dos resultados da classificação.

6.1. Detalhes da Implementação

Essa seção se destina a mostrar alguns detalhes da implementação do algoritmo ABC que tornaram viáveis os experimentos efetuados. Vale lembrar que o ABC é destinado a problemas nos quais o conjunto de dados tende a crescer indefinidamente e, assim, uma implementação que não busque otimizar a utilização de memória pode inviabilizar a sua utilização. Por este motivo, alguns aspectos relevantes da implementação são aqui descritos.

O algoritmo ABC foi modelado em UML, e essa modelagem se encontra no endereço http://www.dc.ufscar.br/~estevam/ABC.zip, junto com os executáveis utilizados.

A implementação do ABC foi feita em C++, utilizando a ferramenta Microsoft Visual C, e na implementação foram utilizados recursos da MFC (Microsoft Foundation Classes).

6.1.1. O uso de AD-Tree para representação de conhecimento

A implantação da AD-Tree no algoritmo ABC se deu em quatro experimentos, sempre visando o menor tamanho da AD-Tree em memória. O primeiro e segundo experimentos não foram bem sucedidos, já o terceiro e quarto experimentos foram bem sucedidos, e ambos podem ser utilizados. O objetivo de se mostrar todos os quatro experimentos (mesmo os dois iniciais que não são adequados para domínios de aplicação muito grandes) é mostrar ao leitor os caminhos já tentados e que não trouxeram bons resultados, assim podem ser evitados.

O critério para avaliar os experimentos foi tentar montar a AD-Tree usando a base de dados Alarm [2], que é uma base de dados com muitas variáveis, várias delas não binárias. São 37 variáveis no total. O número possível de nós da AD-Tree, de acordo com a Equação 5, para essa base de dados resulta em 2,13986E+21 possíveis nós. Ou seja, se cada nó da AD-Tree ocupar um byte, o que é irreal, a AD-Tree poderia vir a ocupar 2 bilhões de terabytes. Então é preciso otimizar muito a AD-Tree para que ela venha a ocupar um tamanho que seja razoável.

O ambiente em que a AD-Tree foi testada é um computador com aproximadamente 400 megabytes de memória disponível.

Primeiro experimento

O primeiro experimento foi a implementação da AD-Tree como especificado em [34], na primeira otimização proposta por Moore e Lee. A otimização consiste em não armazenar ADnodes com *count*=0. No lugar desses ADnodes se coloca um ponteiro para nulo (NULL).

Segue na Figura 11 a representação de uma AD-Tree no primeiro experimento:

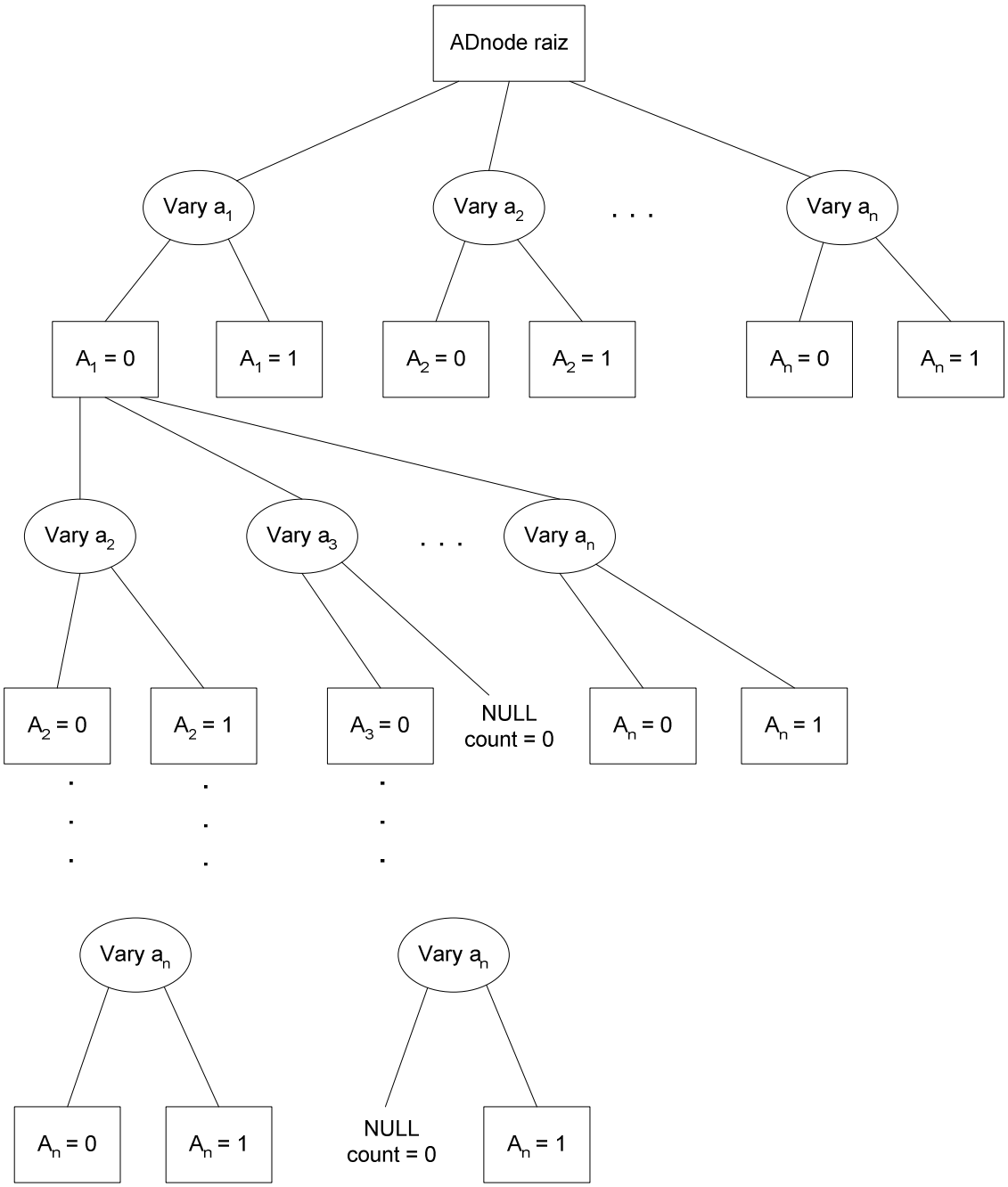

Figura 11: AD-Tree no primeiro experimento.

Em testes, não foi possível montar essa AD-Tree usando a base de dados Alarm. O número de nós da AD-Tree foi muito grande, e o tamanho da AD-Tree foi maior que 400 megabytes.

Segundo experimento

O segundo experimento foi baseado em mais uma otimização sobre a primeira otimização proposta por Moore e Lee em [34], implementada no primeiro experimento. A otimização é baseada no algoritmo K2 [11], que é usado nesse

trabalho. Nele há um limite de pais que cada nó pode ter. Como a busca sobre a AD-Tree é feita com base nos pais de cada nó mais o nó em si, então a altura máxima da AD-Tree para o K2 seria no máximo número de pais mais um. Essa otimização reduz muito o tamanho da AD-Tree, já que sem essa tática a AD-Tree seria expandida por completo. Para o seguinte contexto: usando a base de dados Alarm, com limite de pais igual a quatro, a altura máxima da AD-Tree seria reduzida de 37 para 5.

Segue a representação de uma AD-Tree no segundo experimento e no contexto especificado acima na Figura 12.

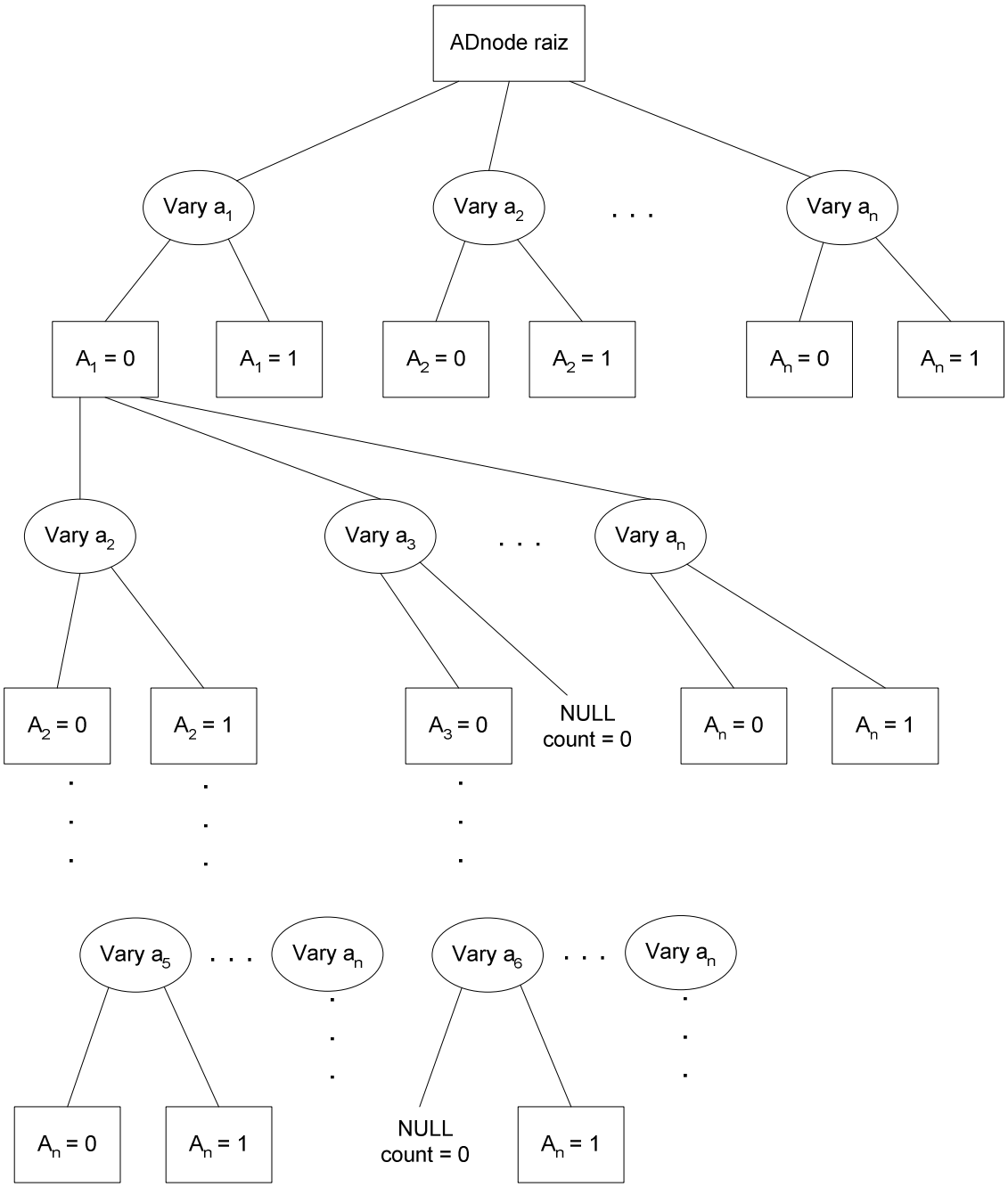

Figura 12: AD-Tree no segundo experimento e para um limite de pais=4.

Em testes, não foi possível montar essa AD-Tree usando a base de dados Alarm. O número de nós da AD-Tree foi menor do que o do primeiro experimento, mas mesmo assim foi muito grande, e por isso o tamanho da AD-Tree foi maior que 400 megabytes.

Terceiro experimento

O terceiro experimento foi uma otimização mais radical. A otimização por altura da AD-Tree do segundo experimento não foi suficiente, então no terceiro experimento se tentou fazer a otimização por largura da AD-Tree.

O problema é que para fazer a otimização por largura da AD-Tree é preciso alterar o funcionamento das pesquisas feitas na AD-Tree. Tem que se alterar o funcionamento da consulta quando uma variável assume o valor a_n=*, que na AD-Tree original seria o valor "don't care" (não interessa). Antes de explicar a alteração feita será explicado o funcionamento da consulta da AD-Tree original. Segue abaixo na Figura 13 a representação do funcionamento de uma consulta na AD-Tree original, para a consulta a_3=1, a_6=0, a_8=1:

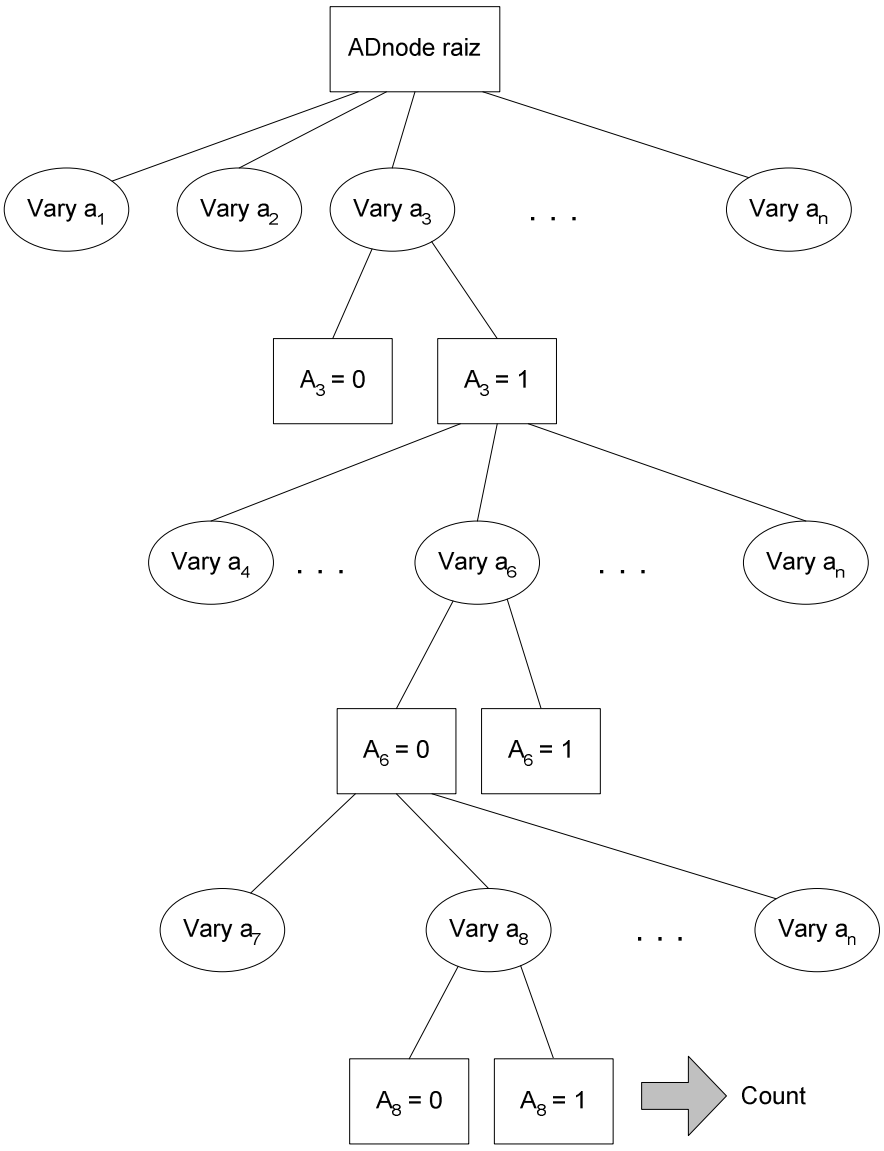

Figura 13: Consulta na AD-Tree original.

A otimização feita consistiu em não mais criar todos os nós Vary possíveis para cada nó ADnode da AD-Tree. Assim para o nó a_j se cria somente o nó Vary a_{j+1}. A exceção é o nó ADnode raiz, que continua tendo todos os nós Vary possíveis. Dessa forma cada ramo da AD-Tree será expandida nó por nó até chegar ao nó a_n, que é o nó folha. A Figura 14 mostra a representação de uma AD-Tree no terceiro experimento.

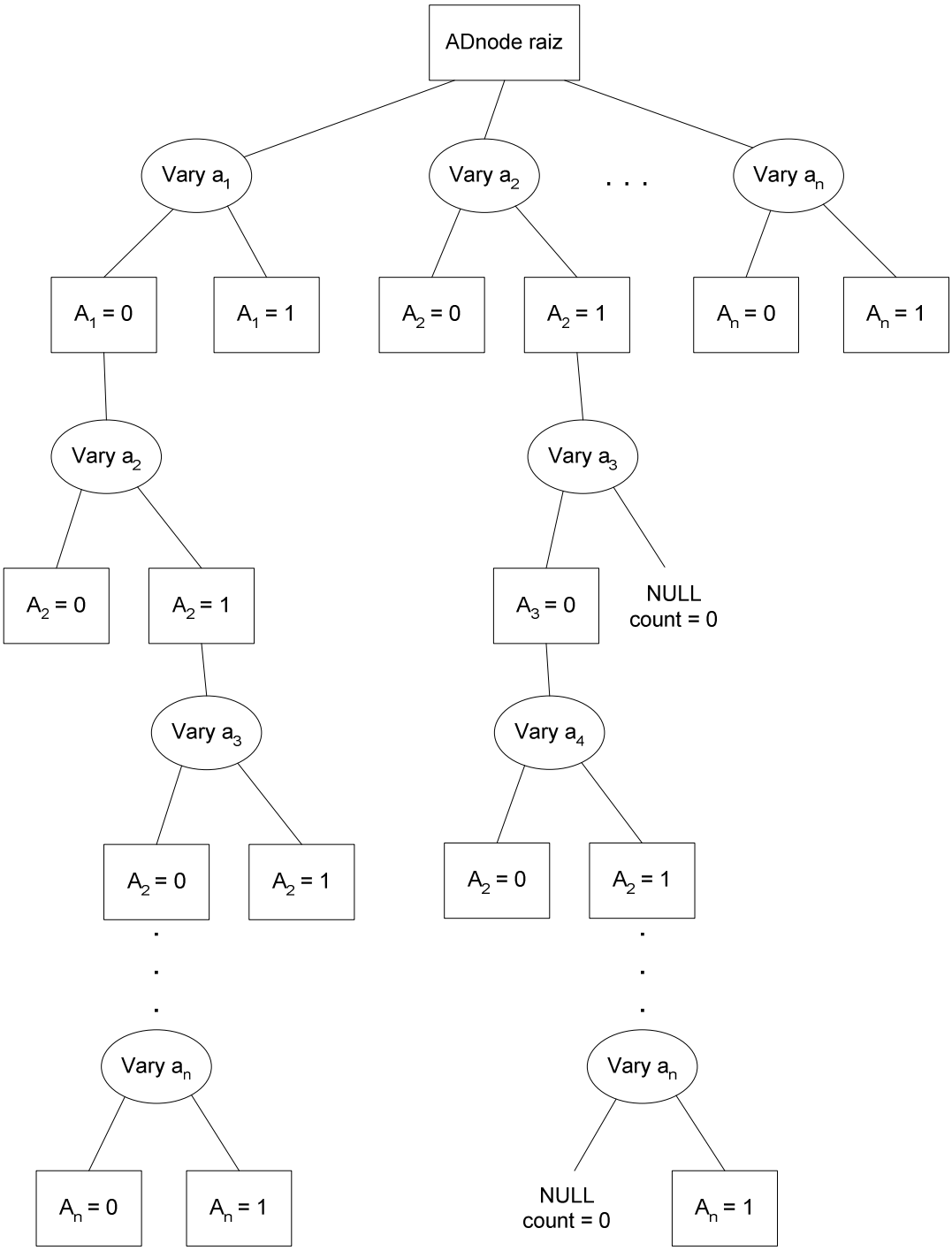

Figura 14: AD-Tree no terceiro experimento.

Da forma como é feita a AD-Tree no terceiro experimento não se consegue mais fazer a consulta a_3=1, a_6=0, a_8=1 da forma como foi mostrado na Figura 13, pois já não se consegue acessar diretamente o ADnode a_6 a partir do ADnode a_3, assim como não se consegue acessar diretamente o ADnode a_8 a partir do ADnode a_6. Para fazer essa consulta, tem que se tratar a consulta de outra forma.

Na consulta a_3=1, a_6=0, a_8=1, embora não se especifique, estão presentes os nós a_4, a_5 e a_7, na forma a_3=1, a_4=*, a_5= *, a_6=0, a_7=*, a_8=1. Os nós anteriores a a_3 e posteriores a a_8 também estão na consulta, porém tanto na AD-Tree original quanto na AD-Tree no terceiro experimento eles não são relevantes para a consulta.

Para fazer a consulta a_3=1, a_6=0, a_8=1 e obter o mesmo resultado que se encontra fazendo a mesma consulta na AD-Tree original, é preciso alterar a forma como se trata os nós a_4, a_5 e a_7. Na AD-Tree original esses nós são ignorados, e se acessa somente os nós que tenham algum valor válido na consulta. Ou seja, o valor '*' para um nó significa nenhum, e o nó que tenha o valor '*' será ignorado na consulta. Já na AD-Tree no terceiro experimento o valor '*' significa todos. Assim quando um nó tiver o valor '*' a consulta se expandirá por todos os possíveis valores para aquele nó. A Figura 15 mostra a representação do funcionamento da consulta a_3=1, a_6=0, a_8=1 na AD-Tree no terceiro experimento:

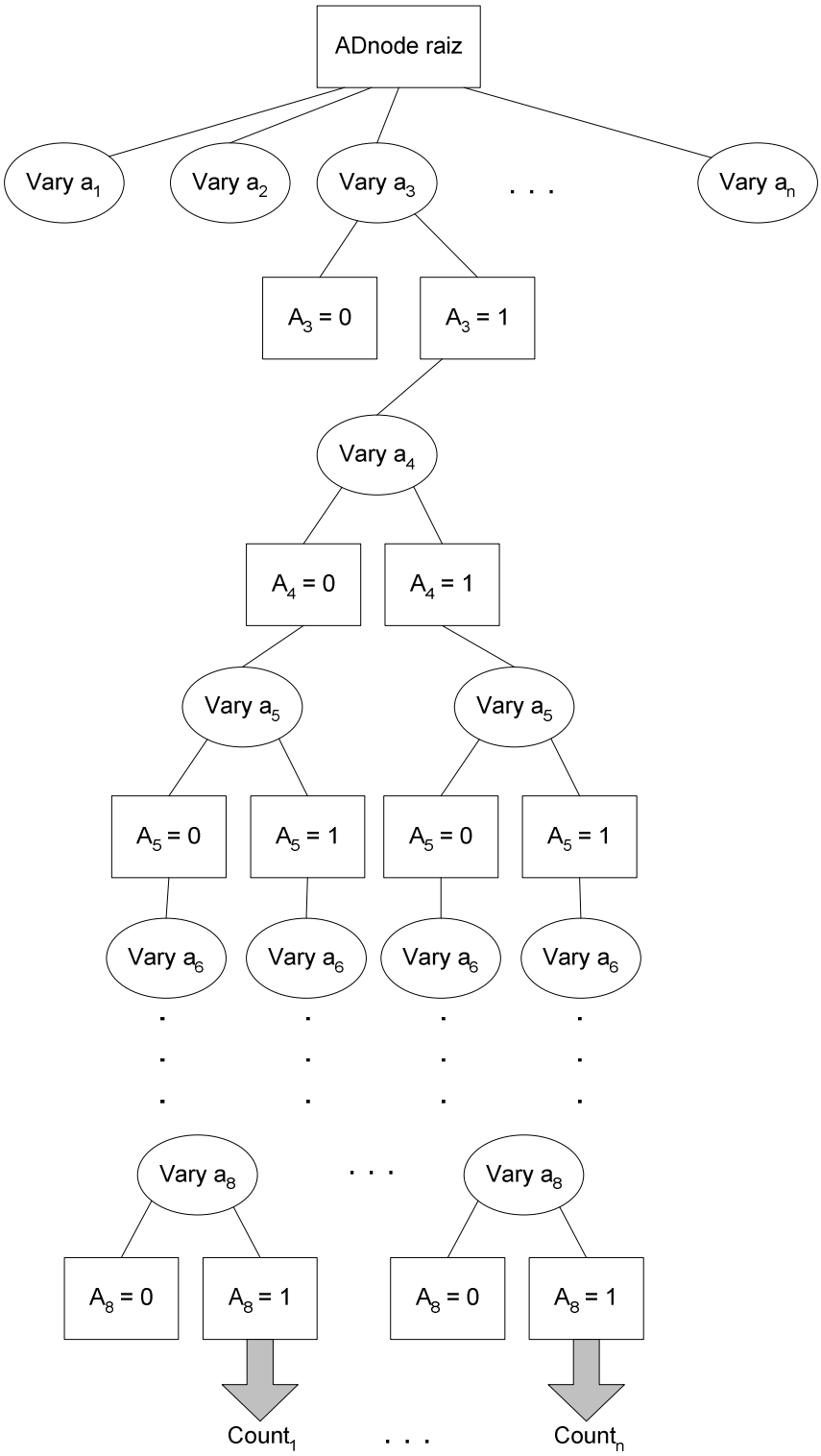

Figura 15: Consulta na AD-Tree no terceiro experimento.

O *count* da consulta acima é calculado segundo a Equação 6 abaixo:

$$count_{total} = \sum_{i=1}^{n} count_i \tag{6}$$

A AD-Tree no terceiro experimento, por não se expandir totalmente por todos os nós possíveis, talvez não possa ser chamada de AD-Tree. Porém por ela se expandir por todos os nós no nó raiz, ela tampouco pode ser chamada de kd-tree [3], [35]. Seria no caso uma "AKD-Tree", já que inicialmente ela se expande por todos os nós possíveis, e depois ela se expande tendo k=1.

A escolha de se expandir inicialmente a árvore por todos os nós possíveis vem de duas causas:

- Durante a implementação do algoritmo ABC viu-se que mais da metade das consultas feitas são de um nó somente. Por exemplo, a_5=0. Nesse caso, acessa-se diretamente o nó a_5 a partir do nó raiz e se busca o *count* da consulta com custo mínimo.
- Como a consulta é expandida por todos os nós que tenham valor igual a '*', o tamanho da busca pode ser muito grande. Tendo todos os possíveis nós disponíveis inicialmente, pode-se acessar diretamente o primeiro nó com valor diferente de '*' na consulta, otimizando o tempo da consulta.

A primeira causa, de que mais da metade das consultas feitas são de um nó somente, se deve a natureza hill-climbing do algoritmo K2 [11], sobre o qual foi implementado o ABC. Nele antes de se adicionar algum pai a um nó, tenta-se adicionar todos os outros nós antes. Isso gera um grande volume de consultas com somente um nó.

Os testes com a AD-Tree no terceiro experimento usando a base de dados Alarm com cem mil casos foram bem sucedidos. Durante os testes se conseguiu montar a AD-Tree e durante o processo foram gerados aproximadamente quatro milhões de nós ADnode e mais quatro milhões de nós Vary.

Porém os testes com a AD-Tree no terceiro experimento usando a base de dados Alarm com um milhão de casos não foram bem sucedidos. O tamanho da AD-Tree gerada ficou acima dos 400 megabytes de memória disponíveis.

Apesar de ser a mesma base de dados Alarm nos dois testes, o número de nós com a base de dados de um milhão de casos foi bem maior do que com a base de dados de cem mil casos. O número de nós da AD-Tree não é influenciado diretamente pelo número de casos da base de dados, mas sim pelo número de diferentes combinações de valores dos nós. Então quanto maior o tamanho da base de dados, maior a probabilidade de haver mais combinações de valores. Porém se houvesse uma base de dados de um milhão de casos com somente uma combinação de valores dos nós, o tamanho da AD-Tree seria bem pequeno.

Para uma base de dados com menos variáveis, ou com variáveis com domínios menores, a AD-Tree no terceiro experimento é uma boa solução, já que ela faz uma boa mescla entre tamanho da árvore em memória e performance das consultas. Porém para bases de dados com muitas variáveis e com variáveis com domínios grandes essa AD-Tree pode não ser uma boa alternativa já que nessas situações ela tende a ocupar muito espaço em memória.

Quarto experimento

O quarto experimento foi a mais radical, e o objetivo dela foi conseguir montar a AD-Tree usando a base de dados Alarm com um milhão de casos, que foi a situação em que a AD-Tree no terceiro experimento falhou.

A otimização consiste em eliminar todos os ramos da árvore, exceto o primeiro. O nó raiz continua se expandindo por todos os nós possíveis, para poder tratar as consultas em que só há um nó com valor diferente de '*'. Porém depois desse primeiro nível somente o ramo da árvore que parte do primeiro nó (a_1) é expandido. A Figura 16 mostra uma representação da AD-Tree no quarto experimento.

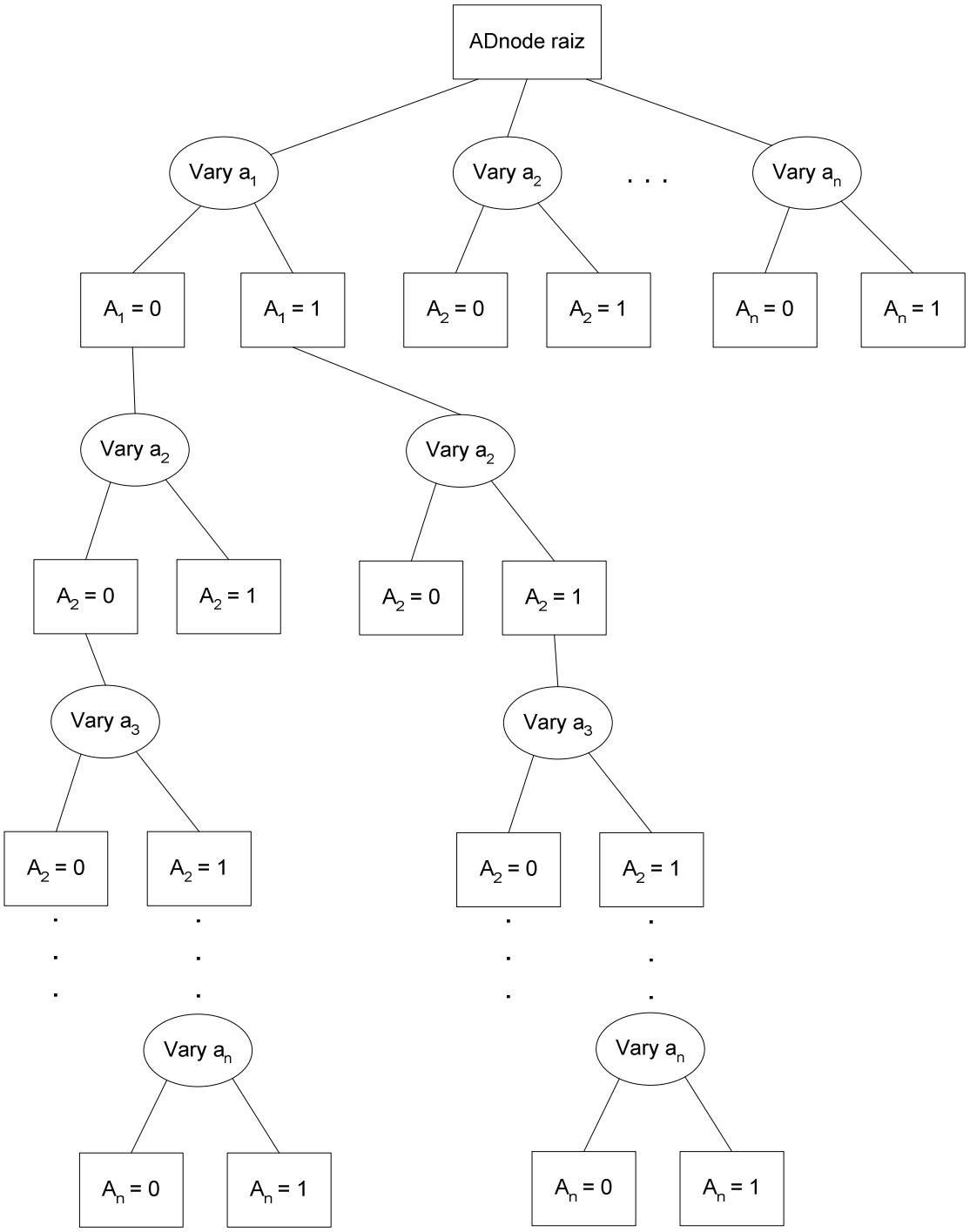

Figura 16: AD-Tree no quarto experimento.

Por ter somente o primeiro ramo da árvore expandido, as consultas feitas na AD-Tree no quarto experimento que tenham mais de um nó com valor diferente de '*' sempre começam a partir do nó a_1. Isso piora a performance das consultas, caracterizando então uma situação de troca entre espaço da árvore em memória e performance da consulta. A consulta a_3=1, a_6=0, a_8=1 tem de ser feita da

seguinte maneira: a_1=*, a_2=*, a_3=1, a_4=*, a_5=*, a_6=0, a_7=*, a_8=1. A Figura 17 mostra a representação dessa consulta.

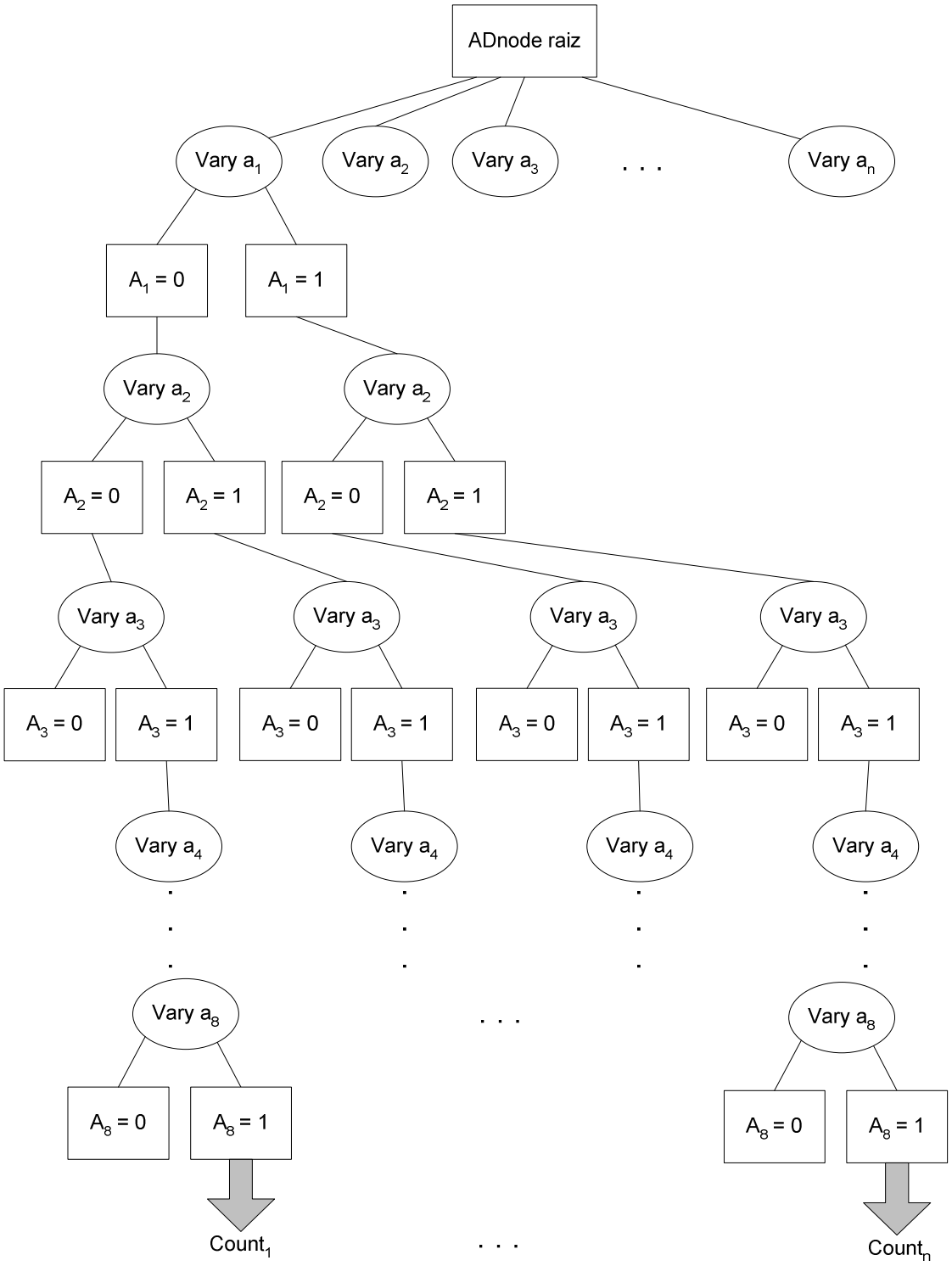

Figura 17: Consulta na AD-Tree no quarto experimento.

O valor do *count* da consulta é calculado segundo a Equação 6. Se a AD-Tree no terceiro experimento pode ser chamada de “AKD-Tree”, a AD-Tree no quarto experimento pode ser chamada de “AKD-Tree reduzida”.

Os testes feitos com a AD-Tree no quarto experimento usando a base de dados Alarm foram bem sucedidos, tanto para a base de dados Alarm com cem mil casos quanto para a base de dados Alarm com um milhão de casos.

Para efeito de comparação, os testes com a AD-Tree no terceiro experimento usando a base de dados Alarm com cem mil casos geraram aproximadamente quatro milhões de nós ADnode e mais quatro milhões de nós Vary. Já os testes com a AD-Tree no quarto experimento usando a mesma base de dados Alarm com cem mil casos geraram aproximadamente 280 mil nós ADnode mais 250 mil nós Vary, uma redução de 16 vezes no número de nós da árvore.

Já os testes com a AD-Tree no quarto experimento com a base de dados Alarm com um milhão de casos ocupou aproximadamente trinta megabytes de memória, ante mais de 400 megabytes nos testes com a AD-Tree no terceiro experimento.

A performance das consultas no entanto piorou. Isso se refletiu no tempo computacional exigido para fazer o aprendizado de estrutura e parâmetros numéricos da rede Bayesiana (o que já era esperado).

6.1.2. Classificador

O classificador utilizado nos experimentos não foi otimizado pois não era o foco do problema. Porém constatou-se durante a execução dos testes que o tempo gasto na classificação do conjunto de teste é bem grande, e por vezes até maior do que o tempo gasto no próprio aprendizado da rede Bayesiana.

Por essas razões se separou o tempo de aprendizado da rede Bayesiana e o tempo de classificação do conjunto de teste. O tempo de classificação do conjunto de teste não foi desconsiderado, apenas foi separado.

Por essa razão se pensou em um algoritmo de classificação otimizado, mas não houve tempo hábil de implementá-lo. Mas a idéia do algoritmo será colocada aqui, para que se implemente num trabalho futuramente.

O princípio é o seguinte: um registro que será classificado tem o valor de cada variável como sendo $a_1=v_1$, $a_2=v_2$,..., $a_n=v_n$. Durante a classificação se estima o valor da variável classe como sendo $a_m=v_m$. Assim sendo, se houver 20 registros com o mesmo valor do registro classificado anteriormente, haverá 20 vezes o mesmo resultado ($a_m=v_m$).

Então a idéia é agrupar os registros por valor de registro, e então classificar uma vez somente cada grupo. Após a classificação se compara o valor da variável classe com o valor obtido na classificação, e se somará os acertos para aquele grupo aos acertos da base de dados, e o número de registros daquele grupo ao número de registros da base de dados.

Para agrupar os registros por valor pode-se usar estrutura como a AD-Tree, por exemplo. Ela seria usada de forma diferente da usada durante o aprendizado de rede Bayesiana, evidentemente. As consultas que seriam feitas na AD-Tree seriam consultas com todas as variáveis da base de dados exceto a variável classe com seu respectivo valor, o conjunto dos valores formando o valor de registro que caracteriza o grupo de registros. Essa consulta retornaria o numero de registros do grupo de registros. Ao alterar essa consulta colocando a variável classe na consulta e como valor da variável classe o valor inferido pela classificação para aquele grupo de registros se obtém o número de acertos para aquele grupo de registros.

Uma otimização interessante que pode ser feita a esse classificador já otimizado é fazer os grupos de registros e a AD-Tree somente com as variáveis do Markov Blanket [24] da variável classe. Se for possível haver mudança de estrutura da rede Bayesiana ou mudança de variável classe talvez seja interessante fazer a AD-Tree com todas as variáveis da base de dados.

Isso pode ser feito pois como as variáveis do Markov Blanket são as que mais influenciam a variável classe, e como todas elas têm valor vindo da base de dados, pode-se desconsiderar as outras variáveis da base de dados que não fazem parte do Markov Blanket da variável classe. O Algoritmo 11 descreve o pseudocódigo do algoritmo classificador otimizado.

```
1.  Algoritmo classificadorOtimizado;
2.  {Entrada: uma rede Bayesiana RB, uma AD-Tree com os dados da base
    de dados ADTree, o índice da variável classe indClasse, uma lista com
    os valores de cada variável dos grupos de registros listaGrupos, o
    numero de nós da rede NumeroNos}
3.  {Saída: o índice de classificação correta ICC}
4.  para cada linha de listaGrupos faça
5.     grupo := getLinhaListaGrupos();
6.     para i:= 0 até NumeroNos faça
7.        se i <> indClasse então
8.           atualizaEvidencia(RB, i, grupo(i));
9.     fim {para};
10.    queryNroRegistros := montaQueryRegistro(ADTree, grupo);
11.    nroRegistros := query(ADTree, queryNroRegistros);
12.    contRegistros := contRegistros + nroRegistros;
13.    atualizaCrença(RB);
14.    maisProvavel := getMaisProvavel (RB, indClasse);
15.    queryNroAcertos := montaQueryAcerto(ADTree, grupo, maisProvavel);
16.    nroAcertos := query(ADTree, queryNroAcertos);
17.    contAcerto := contAcerto + nroAcertos;
18. fim {para};
19. retorne nroAcerto / contLinha;
```

Algoritmo 11: Classificador otimizado.

No pseudocódigo acima existem alguns métodos sem especificação os quais são especificados abaixo:

- **getLinhaListaGrupos**(): retorna os valores de cada nó para o grupo atual de registros.
- **atualizaEvidencia**(): define o estado do nó i da rede Bayesiana RB para o estado presente em grupo(i). Não há nesse método nenhuma propagação de evidências, somente a especificação destes.

- **montaQueryRegistro**(): monta a consulta que será feita na AD-Tree para retornar o numero de registros do grupo de acordo com os valores das variáveis presentes em grupo.
- **query**(): faz a consulta passada por parâmetro na AD-Tree e retorna o *count* resultante da consulta.
- **atualizaCrença**(): faz a propagação das evidências especificadas anteriormente com o método atualizaEvidencia.
- **getMaisProvavel**(): retorna o estado mais provável para o nó i da rede Bayesiana RB depois de feita a propagação de evidências dentro da rede Bayesiana.
- **montaQueryAcerto**(): monta a consulta que será feita na AD-Tree para retornar o numero de acertos do grupo de acordo com os valores das variáveis presentes em grupo e com o valor inferido pela classificação, presente em maisProvavel.

A expectativa é que esse algoritmo classificador otimizado tenha uma performance muito superior à alcançada com o classificador utilizado no ABC. Se isso acontecer o desempenho do próprio algoritmo ABC será melhorado, pois ele utiliza em larga escala o classificador.

6.2. Simulações e Resultados em experimentos de aprendizado usando o ABC e o K2 – uma análise comparativa

Os experimentos realizados com o algoritmo ABC tiveram foco em alto volume de dados, pois é o nicho do algoritmo ABC e do aprendizado incremental de redes Bayesianas. Para pequenos volumes de dados não é necessário o aprendizado incremental, pois o aprendizado em batelada consegue resultados melhores com pouco acréscimo de tempo computacional. Os experimentos foram feitos com no máximo um milhão de registros.

Assim como nos experimentos feitos com o algoritmo AIP, nos experimentos feitos com o algoritmo ABC se analisou o ICC (índice de classificação correta) obtido com a rede Bayesiana gerada pelo algoritmo e o tempo computacional demandado para gerar essa rede. Dessa forma, pode-se fazer uma comparação

entre qualidade da rede Bayesiana gerada e tempo levado para chegar a esse resultado.

As bases de dados utilizadas nos experimentos foram as mesmas dos experimentos feitos com o algoritmo AIP, ou seja:

- Alarm [2], com 37 variáveis, variável classificada: Intubation;
- Asia [31], com 8 variáveis, variável classificada: Smoking;
- Credit [14], com 12 variáveis, variável classificada: Credit Worthiness;
- Engine Fuel System [14], com 9 variáveis, variável classificada: Fuel Filters and Bypass Valves.

Para cada base de dados foram criados dois cenários, um chamado estático e outro chamado dinâmico. Cada cenário foi executado utilizando o algoritmo ABC e o algoritmo K2, ambos usando a AD-Tree. O cenário estático é organizado da seguinte forma:

- Utilizando a rede Bayesiana original de cada base de dados e o software Genie [14] são gerados dez conjuntos de dados de 100.000 registros.
- As execuções feitas com o algoritmo ABC são feitas utilizando esses conjuntos de dados.
- A cada execução do algoritmo ABC é criado um conjunto de dados com o conjunto de dados utilizado naquela execução do algoritmo ABC mais todos os conjuntos de dados utilizados anteriormente, na ordem em que foram utilizados.
- As execuções feitas com o algoritmo K2 são feitas utilizando esses conjuntos de dados. Ou seja, enquanto o ABC na sua sexta execução é executado com um conjunto de 100.000 registros, o K2 é executado com um conjunto de 600.000 registros.

O conjunto estático simula uma situação onde se inicia com uma base de dados de 100.000 registros e com o passar do tempo novas informações vão chegando (novas bases de dados com mais 100.000 registros), mas a distribuição de probabilidade que governa os dados é sempre a mesma. Ou seja, tanto o

primeiro conjunto de dados quanto o décimo são gerados a partir do mesmo modelo de distribuição de probabilidade.

O cenário dinâmico também utiliza dez conjuntos de dados de 100.000 registros, porém eles são gerados de forma diferente:

- O primeiro conjunto de dados de 100.000 registros é gerado a partir de uma rede Bayesiana original;
- É feita então uma alteração na estrutura da rede Bayesiana original, e então é gerado outro conjunto de dados de 100.000 registros;
- É feita uma alteração (pouco significativa) nos parâmetros numéricos de um nó da rede Bayesiana anterior, e são gerados mais 100.000 registros;
- É feita uma nova alteração nos parâmetros numéricos no mesmo nó da rede Bayesiana anterior, porém agora uma alteração mais significativa, e mais 100.000 registros são gerados.

O segundo, o terceiro e o quarto passo são repetidos três vezes e assim, tem-se ao final 10 conjuntos de dados contendo 100.000 registros cada um. Chamando o conjunto de dados gerado a partir da rede Bayesiana original de “O”, o conjunto de dados gerado a partir da alteração na estrutura da rede Bayesiana original de “E” e o conjunto de dados gerado a partir da alteração nos parâmetros numéricos da rede Bayesiana original de “P”, tem-se que a seqüência de geração dos conjuntos de dados é: O-E-P-P-E-P-P-E-P-P.

As três alterações estruturais na rede Bayesiana são feitas sempre da seguinte forma: uma inserção de um arco entre dois nós da rede Bayesiana, uma remoção de um arco entre dois nós da rede Bayesiana e uma inversão de um arco entre dois nós da rede Bayesiana. É importante ressaltar que as alterações não devem se anular, por exemplo, a inserção do arco não pode ser feita entre dois nós que tiveram o arco entre eles removido num passo anterior. Não há uma regra para a ordem em que as alterações são feitas. Outro ponto importante é que se procurou fazer as alterações estruturais e as alterações de parâmetros numéricos em nós do Markov Blanket da variável classe. Quando não foi possível fazer a alteração em um nó do Markov Blanket da variável classe se procurou fazer a alteração em um nó que esteja no Markov Blanket de um nó do Markov

Blanket da variável classe. Nas alterações de inserção de arco e inversão de arco é necessário especificar parâmetros numéricos para o nó filho depois da operação.

Assim, pode-se notar que o cenário dinâmico representa um domínio onde a distribuição de probabilidade que rege as variáveis do problema se altera com o passar do tempo (são dinâmicas). Ou seja, o objetivo deste cenário é simular um problema onde o comportamento das variáveis se altera com o passar do tempo e, por isso, a rede Bayesiana original pode não ser mais adequada para classificar os dados do décimo conjunto de dados. Assim, pretende-se forçar o ABC a trabalhar em condições extremas para verificar a efetividade das camadas propostas.

Pode-se concluir então que ao simular mudanças na distribuição probabilística da base de dados, a alteração estrutural da rede Bayesiana representa uma alteração grande na distribuição. Já as alterações nos parâmetros numéricos da rede Bayesiana representam mudanças mais leves na distribuição probabilística da rede Bayesiana, em graus diferentes. Além disso, a primeira alteração nos parâmetros numéricos é mais leve, e a segunda é moderada.

Os conjuntos de dados foram gerados utilizando o software Genie [14]. Os conjuntos de dados de 100.000 registros são utilizados nos experimentos com o algoritmo ABC, e os conjuntos de dados utilizados nos experimentos com o algoritmo K2 são gerados da mesma forma que no cenário estático, ou seja, na quinta execução do ABC é usado um conjunto de 100.000 registros, enquanto que na quinta execução do K2 é usado um conjunto de 500.000 registros, formado pelos conjuntos já executados pelo ABC anteriormente.

Nos dois cenários, ao se executar o sexto conjunto de dados de 100.000 registros, por exemplo, com o algoritmo ABC, se executa o mesmo conjunto de dados de 100.000 registros mais os cinco conjuntos anteriores, ou seja, um conjunto de dados de 600.000 registros com o algoritmo K2. Ao final do experimento, enquanto o algoritmo ABC é executado com o décimo conjunto de dados de 100.000 registros, o algoritmo K2 é executado com um conjunto de dados de um milhão de registros. O objetivo é fazer a comparação justa entre o algoritmo ABC e o algoritmo K2, já que o ABC, por ser baseado um algoritmo

incremental, guarda estatísticas das execuções anteriores, enquanto que o K2 não.

Seguem abaixo a descrição do cenário dinâmico de cada base de dados junto com os resultados dos experimentos feitos com cada base de dados. As tabelas de cada cenário estão no Apêndice II. Nas tabelas foi separado o tempo gasto com o algoritmo ABC do tempo gasto com a classificação. Isso foi feito para que se pudesse comparar de forma mais justa os cenários, já que a classificação é muito utilizada pelo algoritmo ABC e o classificador utilizado nos testes não é otimizado. O tempo de classificação também serve para reforçar a necessidade de se implementar o classificador otimizado.

O limite de classificação para cada cenário executado pelo algoritmo ABC foi definido procurando sempre deixar esse limite abaixo do menor ICC gerado pelo algoritmo K2.

Alarm – variável classificada: Intubation

A Tabela 7 descreve a forma como foi feito o cenário dinâmico da base de dados Alarm:

Tabela 7: Cenário dinâmico Alarm.

RB	Tipo de alteração	Alteração
1	rede Bayesiana original	nenhuma
2	Estrutura	remoção arco Intubation -> VentLung
3	Parâmetros numéricos	alteração do nó VentAlv quando pai VentLung=0
4	Parâmetros numéricos	alteração do nó VentAlv quando pai VentLung=0
5	Estrutura	inversão arco Intubation -> Shunt
6	Parâmetros numéricos	alteração do nó PulmEmbolus, sem pais
7	Parâmetros numéricos	alteração do nó PulmEmbolus, sem pais
8	Estrutura	inserção arco FiO2 -> Intubation
9	Parâmetros numéricos	alteração do nó KinkedTube, sem pais
10	Parâmetros numéricos	alteração do nó KinkedTube, sem pais

Foram executados os seguintes cenários com a base de dados Alarm:

- Estático ABC, limite de classificação = 90% - Tabela 32;
- Estático K2 - Tabela 32;
- Dinâmico ABC, limite de classificação = 90% - Tabela 33;
- Dinâmico K2 - Tabela 33.

A Tabela 8 resume os resultados dos experimentos conduzidos.

Tabela 8: Resumo cenários Alarm – ABC.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Alarm			
Cenário	Tapr ± DP	Tclass ± DP	MICCT ± DP
Estático ABC	877,92 ± 9,30	1.971,67 ± 19,02	98,58 ± 0,03
Estático K2	8.866,55 ± 112,37	2.338,40 ± 12,58	98,59 ± 0,04
Dinâmico ABC	926,59 ± 9,31	27.572,38 ± 15.075,89	95,06 ± 2,22
Dinâmico K2	12.318,85 ± 168,83	2.609,52 ± 54,36	97,16 ± 1,72

Os experimentos com a base de dados Alarm evidenciam principalmente a questão do tempo. No tempo gasto para o aprendizado, desconsiderando o tempo gasto com a classificação, pode-se ver claramente a vantagem do uso do algoritmo ABC sobre o algoritmo K2. A diferença foi de 10 vezes no cenário estático e 13 vezes no cenário dinâmico. Já no tempo gasto na classificação fica evidente a necessidade de se otimizar o classificador utilizado nos experimentos. O tempo de 27 mil segundos, na média, gasto na classificação no cenário dinâmico com o algoritmo ABC fala por si só.

Esse tempo grande gasto na classificação não ocorre no cenário estático com o algoritmo ABC. Acontece isso porque no cenário estático não foi necessário acionar alguma camada exceto a primeira. Já no cenário dinâmico com o algoritmo ABC foi necessário acionar a segunda camada. E nessa camada se utiliza em larga escala a classificação, e daí vem a causa do tempo grande gasto na classificação.

Se por um lado a execução da segunda camada do algoritmo ABC gerou esse tempo grande gasto na classificação, por outro lado ele foi bem sucedido no seu objetivo primário, que é encontrar uma rede Bayesiana que tenha um ICC acima do limiar definido. Analisando os resultados obtidos nos experimentos, vê-se que a execução da segunda camada gerou uma melhora de 2 a 5% no ICC (2,5% na média). Essa melhora se deve a escolha de uma rede Bayesiana do conjunto de redes vizinhas. O interessante é que as redes escolhidas foram redes

que foram geradas através de alterações sobre os nós do Markov Blanket da variável classe.

Analisando os resultados de ICC versus tempo gasto no aprendizado, percebe-se que o algoritmo ABC tem sucesso em manter o ICC acima do limiar definido, aliás bem próximo do ICC obtido com o algoritmo K2, utilizando para isso um tempo de processamento bem menor, no tocante às tarefas de aprendizado.

Asia – variável classificada: Smoking

A Tabela 9 descreve como foi feito o cenário dinâmico da base de dados Ásia:

Tabela 9: Cenário dinâmico Ásia.

RB	Tipo de alteração	Alteração
1	rede Bayesiana original	nenhuma
2	Estrutura	remoção arco Smoking -> Bronchitis
3	Parâmetros numéricos	alteração do nó Lung Cancer, quando pai Smoking=Smoker
4	Parâmetros numéricos	alteração do nó Lung Cancer, quando pai Smoking=Smoker
5	Estrutura	inserção arco Visit To Asia -> Lung Cancer
6	Parâmetros numéricos	alteração do nó Visit to Asia, sem pais
7	Parâmetros numéricos	alteração do nó Visit to Asia, sem pais
8	Estrutura	inversão do arco Visit to Asia -> Tuberculosis
9	Parâmetros numéricos	alteração do nó Tuberculosis, sem pais
10	Parâmetros numéricos	alteração do nó Tuberculosis, sem pais

Foram executados os seguintes cenários com a base de dados Ásia:

- Estático ABC, limite de classificação = 52% - Tabela 34;
- Estático K2 - Tabela 34;
- Dinâmico ABC, limite de classificação = 52% - Tabela 35;
- Dinâmico K2 - Tabela 35.

A Tabela 10 resume os resultados dos experimentos conduzidos com a base de dados Ásia.

Tabela 10: Resumo cenários Ásia – ABC.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Ásia			
Cenário	Tapr ± DP	Tclass ± DP	MICCT ± DP
Estático ABC	263,81 ± 44,75	93,12 ± 10,92	66,62 ± 0,10
Estático K2	1.748,60 ± 47,55	201,82 ± 2,37	66,75 ± 0,16
Dinâmico ABC	267,92 ± 1,59	87,57 ± 0,17	66,84 ± 5,10
Dinâmico K2	1.713,64 ± 192,39	263,64 ± 30,05	60,84 ± 5,37

Os experimentos com a base Ásia podem ser considerados como muito bem sucedidos. O algoritmo ABC gastou menos tempo tanto nas tarefas de aprendizado como nas tarefas de classificação nos dois cenários, sendo que no cenário estático o ICC do algoritmo ABC ficou bem próximo do ICC obtido com o algoritmo K2, e no cenário dinâmico o ICC obtido com o algoritmo ABC foi maior do que o obtido com o algoritmo K2.

O tempo gasto na classificação nos cenários usando o algoritmo ABC foi menor do que o tempo gasto na classificação nos cenários usando o algoritmo K2 porque não foi necessário executar alguma camada exceto a primeira. Nesse caso, o fato do algoritmo ABC obter melhores ICCs do que o algoritmo K2 se deve ao aprendizado feito na primeira execução do algoritmo ABC, utilizando o primeiro conjunto de dados de 100.000 registros. A análise que pode ser feita é que os dados presentes nos segundo e terceiro conjuntos têm uma distribuição de probabilidade diferente da presente no primeiro conjunto. E depois desses dois conjuntos, do quarto conjunto para frente, a distribuição de probabilidade presente no primeiro conjunto volta, possibilitando que a rede Bayesiana aprendida na primeira execução se mantenha adequada para representar a distribuição de probabilidade da base de dados.

De modo análogo, a presença desses dois conjuntos (o segundo e o terceiro) no aprendizado da rede Bayesiana quando foi usado o algoritmo K2 piorou a qualidade da rede Bayesiana aprendida, pois esses dois conjuntos atuam no sentido de induzir uma distribuição de probabilidade diferente da existente na base de dados. Esse efeito provocado por esses dois conjuntos persiste até a sexta ou sétima execução, quando então há dados suficientes para mascarar os efeitos produzidos pelo segundo e terceiro conjuntos.

De todo modo, é muito positivo que o algoritmo ABC tenha conseguido um ICC 6% maior do que o obtido com o algoritmo K2.

Credit – variável classificada: Credit Worthiness

A Tabela 11 descreve como foi feito o cenário dinâmico da base de dados Credit:

Tabela 11: Cenário dinâmico Credit.

RB	Tipo de alteração	Alteração
1	rede Bayesiana original	nenhuma
2	Estrutura	remoção arco Ratio of Debts -> Credit Worthiness
3	Parâmetros numéricos	alteração do nó Reliability, quando pai Payment History=Excellent
4	Parâmetros numéricos	alteração do nó Reliability, quando pai Payment History=Excellent
5	Estrutura	inserção arco Income -> Credit Worthiness
6	Parâmetros numéricos	alteração do nó Age, sem pais
7	Parâmetros numéricos	alteração do nó Age, sem pais
8	Estrutura	inversão arco Future Income -> Credit Worthiness
9	Parâmetros numéricos	alteração do nó Income, sem pais
10	Parâmetros numéricos	alteração do nó Income, sem pais

Foram executados os seguintes cenários com a base de dados Credit:

- Estático ABC, limite de classificação = 70% - Tabela 36;
- Estático K2 - Tabela 36;
- Dinâmico ABC, limite de classificação = 70% - Tabela 37;
- Dinâmico K2 - Tabela 37.

A Tabela 12 resume os resultados dos experimentos conduzidos com a base de dados Credit.

Tabela 12: Resumo cenários Credit – ABC.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Credit			
Cenário	Tapr ± DP	Tclass ± DP	MICCT ± DP
Estático ABC	338,42 ± 11,01	158,73 ± 1,41	72,19 ± 0,09
Estático K2	1.834,54 ± 27,40	403,17 ± 4,45	72,19 ± 0,13
Dinâmico ABC	404,99 ± 30,93	1.107,09 ± 77,38	71,26 ± 0,63
Dinâmico K2	1.903,70 ± 34,46	516,78 ± 5,32	71,73 ± 0,41

Nos experimentos com a base de dados Credit novamente o algoritmo ABC obteve ICCs próximos aos obtidos com o algoritmo K2 em menos tempo, no tocante às tarefas de aprendizado.

O tempo gasto na classificação no cenário dinâmico com o algoritmo ABC foi grande, o que indica que outra camada do algoritmo ABC foi executada. Nesse cenário isso ocorreu em dois pontos, na execução com o segundo conjunto de 100.000 registros e na execução com o quinto conjunto de 100.000 registros. Na execução com o segundo conjunto, a segunda camada foi executada e foi bem sucedida, encontrando uma rede Bayesiana como ICC acima do limiar estabelecido.

Essa rede Bayesiana continuou com ICC maior que o limiar estabelecido até a execução com o quinto conjunto. Então foi necessário buscar outra rede Bayesiana. O interessante aqui é que nesse ponto foi executada a terceira camada. A segunda camada foi executada antes, e não conseguiu encontrar nenhuma rede Bayesiana com ICC acima do limiar estabelecido. Então a terceira camada foi executada, atualizando os parâmetros numéricos das redes Bayesianas, tanto a oficial como as que estão dentro do conjunto de redes vizinhas. Após a atualização foi encontrada uma rede Bayesiana com ICC acima do limiar estabelecido, o que significa efetivamente que a atualização dos parâmetros numéricos da rede Bayesiana surtiu efeito, aumentando o ICC obtido com aquela rede Bayesiana.

Engine Fuel System – variável classificada: Fuel Filters and Bypass Valves

A Tabela 13 descreve como foi feito o cenário dinâmico da base de dados Engine Fuel System:

Tabela 13: Cenário dinâmico Engine Fuel System.

RB	Tipo de alteração	Alteração
1	rede Bayesiana original	nenhuma
2	Estrutura	remoção arco Fuel Filters -> Fuel Pressure Low
3	Parâmetros numéricos	alteração do nó Fuel Delivery System, sem pais
4	Parâmetros numéricos	alteração do nó Fuel Delivery System, sem pais
5	Estrutura	inserção arco Fuel Filters -> Fuel Pressure High
6	Parâmetros numéricos	alteração do nó Load Test Fuel Pressure, quando pai Fuel Filters=defective
7	Parâmetros numéricos	alteração do nó Load Test Fuel Pressure, quando pai Fuel Filters=defective
8	Estrutura	inversão arco Fuel Filters -> Fuel Pressure Drop
9	Parâmetros numéricos	alteração do nó Inpection Fuel Sight Glasses, quando pai Fuel Delivery System=defective
10	Parâmetros numéricos	alteração do nó Inpection Fuel Sight Glasses, quando pai Fuel Delivery System=defective

Foram executados os seguintes cenários com a base de dados Engine Fuel System:

- Estático ABC, limite de classificação = 90% - Tabela 38;
- Estático K2 - Tabela 38;
- Dinâmico ABC, limite de classificação = 90% - Tabela 39;
- Dinâmico K2 - Tabela 39.

A Tabela 14 descreve os resultados dos experimentos conduzidos com a base de dados Engine Fuel System.

Tabela 14: Resumo cenários Engine Fuel System – ABC.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Engine Fuel System			
Cenário	Tapr ± DP	Tclass ± DP	MICCT ± DP
Estático ABC	231,31 ± 6,19	103,87 ± 0,56	99,97 ± 0,01
Estático K2	1.677,92 ± 200,81	255,56 ± 46,59	99,97 ± 0,01
Dinâmico ABC	247,13 ± 7,95	104,29 ± 0,58	99,83 ± 0,10
Dinâmico K2	1.104,60 ± 8,33	152,36 ± 2,03	99,93 ± 0,03

Nos experimentos com a base de dados Engine Fuel System não foi necessário executar outra camada que não fosse a primeira. Então tanto o tempo de aprendizado como o tempo de classificação gastos pelo algoritmo ABC nos

cenários ficou abaixo dos tempos gasto pelo algoritmo K2 nos mesmos cenários, e o ICC obtido pelos dois algoritmos foi equivalente.

6.3. Discussão dos resultados obtidos com o algoritmo ABC

Para validar os resultados, é importante observar dois aspectos relevantes e decorrentes da mesma origem, a sensibilidade do algoritmo à ordem de chegada dos dados.

O primeiro aspecto é a precisão do modelo obtido com o ABC em relação ao modelo obtido com o algoritmo de aprendizado de redes Bayesianas em batelada, usando os mesmo dados.

O algoritmo de aprendizado em batelada não sofre os efeitos da ordem de chegada dos dados, já que ele considera todos ao mesmo tempo. Já o ABC sofre esses efeitos, e então o ideal é que, ao final do processo de aprendizado, o modelo obtido com o algoritmo de aprendizado incremental em camadas se aproxime o máximo possível do modelo obtido com o algoritmo de aprendizado em batelada. Quanto mais próximos os modelos forem, melhor será a avaliação do algoritmo de aprendizado incremental em camadas, em relação à precisão do modelo obtido.

O outro aspecto é a capacidade do ABC rever a sua rede Bayesiana e fazer eventuais correções nela à medida que chegam dados novos que não seguem o modelo especificado pela rede Bayesiana obtida com o algoritmo incremental.

Segundo [32] e [43], o aprendizado incremental é intrinsecamente hill-climbing, já que ele não tem todos os dados disponíveis durante o aprendizado, e portanto ele não pode fazer uma busca pelo máximo global e garantir que esse máximo seja realmente global em outros momentos. O máximo global em um dado momento t1 pode ser somente um máximo local no momento t2, com mais dados disponíveis.

Como é uma busca hill-climbing, o aprendizado incremental precisa de mecanismos para tratar casos em que a rede Bayesiana atual do algoritmo (o seu máximo local) não seja adequada para eventuais dados novos que cheguem ao algoritmo. Existem várias formas de se fazer esse mecanismo, como, por exemplo, a heurística TOCO de Roure [40] ou a proposta de algoritmo ABC. Se esse mecanismo for eficiente, a rede Bayesiana gerada pelo algoritmo será

semelhante à rede Bayesiana obtida por um algoritmo de aprendizado em batelada.

Portanto, o primeiro aspecto é uma implicação do segundo, porém as avaliações se dão de forma diferente. O primeiro aspecto avalia o resultado do algoritmo, enquanto o segundo avalia a eficácia do processo. Ou seja, um aspecto foca no resultado enquanto o outro foca no processo.

Considerando esses dois aspectos apresentados, pode-se definir os experimentos realizados com o algoritmo ABC como sendo bem sucedidos. Por um lado, os ICCs obtidos com o algoritmo ABC ficaram próximos aos obtidos com o algoritmo K2, sendo que o algoritmo ABC conseguiu gerar ICCs superiores aos gerados pelo algoritmo K2 em alguns casos.

Por outro lado, viu-se durante a realização dos experimentos que a idéia do conjunto de redes vizinhas é válida. Através desse conceito foi possível adaptar a rede Bayesiana oficial quando o ICC por ela gerado não estava acima do limiar estabelecido. Dessa forma a rede Bayesiana oficial esteve sempre adequada aos dados novos.

Então, resumindo, quando o ICC gerado pela rede Bayesiana não ficou acima do limiar estabelecido, o algoritmo foi capaz de contornar essa situação, adaptando a rede Bayesiana. Com essa adaptação a rede Bayesiana passou a gerar novamente ICCs acima do limiar estabelecido.

Um ponto importante a ser discutido é: o algoritmo ABC se encaixa na definição de algoritmo de aprendizado incremental? Vejamos:

Como dito no Capítulo 4, segundo Langley [29], um algoritmo é dito incremental se ele processa um conjunto de treinamento por vez, não reprocessa qualquer conjunto de treinamento anterior e retém somente uma estrutura de conhecimento em memória.

Segundo essa definição, o algoritmo ABC não seria incremental, já que é previsto no algoritmo que ele processe novamente um conjunto de treinamento se necessário, e, além disso, ele armazena mais que uma estrutura de conhecimento em memória, no conjunto de redes vizinhas. Vale dizer que a maioria das propostas de aprendizado incremental para redes Bayesianas não são incrementais segundo essa definição.

Segundo Domingos e Hulten [12] e [13], um algoritmo é dito incremental se ele seguir as seguintes restrições:

1) O algoritmo deve processar um registro em um tempo pequeno e constante;
2) O algoritmo deve ser capaz de construir um modelo usando no máximo uma leitura dos dados;
3) O algoritmo deve usar uma quantidade fixa de memória principal, independentemente do número de registros que o algoritmo tenha processado;
4) O algoritmo deve construir um modelo usável em qualquer momento, ao contrário de somente fazê-lo quando terminar de processar os registros;
5) O algoritmo deve produzir um modelo que seja equivalente ou quase idêntico ao modelo que seria obtido usando um algoritmo de aprendizado não incremental.

O algoritmo ABC segue as restrições 2, 3, 4 e 5. As restrições 2 e 3 só são possíveis graças ao uso da AD-Tree no algoritmo ABC. Já a restrição 4 é plenamente atendida, pois o algoritmo ABC depende de ter um modelo usável a todo momento para poder funcionar. Verificou-se durante os experimentos que a restrição 5 é atendida.

A restrição 1 não é plenamente atendida. Ela será atendida se o ICC gerado quando utilizado o registro processado ficar acima do limiar estabelecido. Caso contrário não se pode dizer que o tempo de processamento do registro será pequeno nem que será constante.

6.4. Trabalhos Futuros

Durante o desenvolvimento da pesquisa aqui descrita se viu a oportunidade de desenvolver alguns trabalhos para complementar o trabalho aqui desenvolvido.

O primeiro e mais evidente é o desenvolvimento do classificador otimizado descrito na Seção 6.1. Durante a execução dos testes com o algoritmo ABC ficou clara a necessidade de otimizar a parte do processo referente à classificação do conjunto de teste.

Outro trabalho futuro interessante é referente ao conjunto de redes vizinhas do algoritmo ABC. Durante os testes se viu que na maioria das vezes em que foi executada a segunda camada se escolheu uma rede Bayesiana que foi criada sobre nós do Markov Blanket da variável classe. Seria interessante então que o conjunto de redes vizinhas fosse criado preferencialmente com operações sobre os nós do Markov Blanket da variável classe. Se o número de redes Bayesianas que puder ser criado a partir de operações sobre os nós do Markov Blanket da variável classe for menor que o número de redes configurado para o conjunto de redes vizinhas então depois de esgotadas as operações sobre os nós do Markov Blanket da variável classe passa-se a gerar as redes Bayesianas com operações sobre os nós que não estejam no Markov Blanket da variável classe.

O terceiro trabalho futuro interessante é mudar a forma como se usa o limite de classificação do algoritmo ABC. Em vez de se usar o limite de classificação de forma absoluta, como se fez nesse trabalho, se usaria o limite de classificação de forma relativa. Na forma atual, se o ICC (índice de classificação correta) obtido com a rede Bayesiana oficial ficar abaixo do limite, de 90% por exemplo, se ativa a próxima camada do algoritmo. Na nova forma sugerida aqui, se alimentaria o algoritmo ABC com o ICC obtido com a rede Bayesiana original daquela base de dados, e então se analisaria o ICC das redes Bayesianas geradas de forma relativa. Se a diferença entre o ICC da rede Bayesiana original e o ICC da rede Bayesiana gerada ficar acima de um intervalo definido se ativaria a próxima camada do algoritmo.

Capítulo 7. Conclusão

O aprendizado incremental de redes Bayesianas é uma área de interesse para pesquisa acadêmica e de muito potencial de uso para as empresas, por vários motivos, entre eles:

- O crescimento constante das bases de dados. Fazer aprendizado em batelada dessas bases pode se tornar proibitivo computacionalmente;
- O aspecto temporal do aprendizado incremental de redes Bayesianas. Uma vez que usando o aprendizado incremental se terá sempre a rede Bayesiana mais atualizada;
- O aspecto adaptativo do aprendizado incremental de redes Bayesianas. Com o aprendizado incremental vai se adaptando a rede Bayesiana ao conjunto total de dados, diminuindo assim o risco de uma base de dados não aleatória produzir uma rede Bayesiana tendenciosa.
- O tempo necessário para fazer o aprendizado incremental de redes Bayesianas é menor do que o tempo do aprendizado em batelada, possibilitando assim o seu uso em aplicações onde tempo é crucial (web-mining, por exemplo).

Os resultados obtidos com o algoritmo ABC mostraram que esse método é completo e que ele consegue gerar bons resultados, sendo o algoritmo ABC em si um método que leva em conta um aspecto qualitativo do aprendizado, reforçando o foco nos resultados. Neste sentido as principais contribuições deste trabalho foram:

- A definição e implementação de dois métodos de aprendizado incremental: o AIP e o ABC.
    - O AIP possui como base um princípio simples e se mostra adequado em problemas onde há condições para um bom aprendizado de estrutura de rede Bayesiana, o que implica na disponibilidade de uma base de dados grande e significativa o suficiente para isso.
    - Já o ABC é um método mais completo, que consegue, se necessário, alterar a estrutura e os parâmetros da rede Bayesiana para adaptar ela aos registros processados.
- A definição e implementação de duas formas de otimização de estruturas AD-Tree.
    - A primeira é chamada “AKD-Tree”, e é uma estrutura de tamanho otimizado, mas que privilegia a performance das consultas feitas na estrutura.
    - Já a segunda otimização é chamada “AKD-Tree reduzida” e privilegia o tamanho da estrutura e por isso tem performance de consultas pior do que a da AKD-Tree, porém seu tamanho é mínimo.

Bibliografia

[1] Anderson, J.R., Matessa, M. Explorations of an incremental, Bayesian algorithm for categorization. *Machine Learning*, 9(4):275-308, 1992.

[2] Beinlich, I., Suermondt, H., Chavez, R., Cooper, G. The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks. *Proceedings of the Second European Conference on Artificial Intelligence and medicine*, 689-693, 1992.

[3] Bentley, J.L. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9): 509-517, 1975.

[4] Bezdek, J.C. Pattern recognition with fuzzy objective function algorithms. Plenum Press, New York, 1981.

[5] Buntine, W. Theory refinement on Bayesian networks. *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, 52-60, 1991.

[6] Buntine, W. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159-225, 1994.

[7] Castelo, R. A Discrete Acyclic Digraph Markov Model in Data Mining. PhD thesis, Faculteit Wiskunde en Informatica, Univeriteit Utrecht, 2002.

[8] Castelo, R., Kocka, T. On inclusion-driven learning of Bayesian networks. *Journal of Machine Learning Research*, 4:527-574, 2003.

[9] Castillo, E., Gutierrez, J., Hadi, A. Expert Systems and Probabilistic Network Models. Springer-Verlag, New York, 1997.

[10] Chow, C.K., Liu, C.N. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14(3):462-467, 1968.

[11] Cooper, G., Herskovits, E. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9(4):309-347, 1992.

[12] Domingos, P., Hulten, G. Catching up with the data: Research issues in mining data streams. *Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2001.

[13] Domingos, P., Hulten, G. A general framework for mining massive data streams. *Journal of Computational and Graphical Statistics*, 12(4):945-949, 2003.

[14] Druzdzel, M.J. SMILE: Structural Modeling, Inference and Learning Engine and GeNIe: A development environment for graphical decision-theoretic models. *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 902-903, 1999.

[15] Druzdzel, M.J., van der Gaag, L.C. Building probabilistic networks: where do the numbers come from?. *IEEE Transactions on Knowledge and Data Engineering*, 12(4):481-486, 2000.

[16] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. From data mining to knowledge discovery: an overview. *Advances in Knowledge Discovery and Data Mining*, AAAI Press, Menlo Park, CA, 1-34, 1996.

[17] Fisher, D.H. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2(2):139-172, 1987.

[18] Friedman, N., Goldszmidt, M. Sequential update of Bayesian network structure. *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, 165-174, 1997.

[19] Friedman, N., Geiger, D. Goldszmidt, M. Bayesian network classifiers. *Machine Learning*, 29(2-3):131–163, 1997.

[20] Gama, J., Castillo, G. Adaptive Bayes. *Proceedings of the 8th Ibero-American Conference of Artificial Intelligence*, 765-774, 2002.

[21] Gennari, J.H., Langley, P., Fisher, D. Models of incremental concept formation. *Artificial Intelligence*, 40(1-3):11-61, 1989.

[22] Hruschka Jr, E.R., Ebecken, N.F.F. Variable ordering for Bayesian networks learning from data. *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation*, Vienna, Austria, 2003.

[23] Hruschka Jr, E.R. et al. Bayesian networks for imputation in classification problems. *Journal of Intelligent Information Systems*, 2007.

[24] Hruscka Jr, E.R. Imputação Bayesiana no contexto da mineração de dados. PhD. Thesis, COPPE-Universidade Federal do Rio de Janeiro, 2003.

[25] Hruscka Jr, E.R. Propagação de Evidências em Redes Bayesianas: Diagnóstico sobre Doenças Pulmonares, MSc. Thesis, CIC-Universidade de Brasília, 1997

[26] Huang, C., Darwiche, A. Inference in belief networks: a procedural guide. *International Journal of Approximate Reasoning*, 15(3), 225-263, 1996.

[27] Hulten, G., Domingos, P. Mining complex models from arbitrarily large databases in constant time. *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 525-531, 2002.

[28] Lam, W., Bacchus, F. Using new data to refine Bayesian networks. *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 383-390, 1994.

[29] Langley, P. Order effects in incremental learning. *Learning in Humans and Machines: Towards an Interdisciplinary Learning Science*, eds., Reimann, P. and Spada, H., Elsevier, Amsterdam, 1995.

[30] Lauritzen, S.L. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19:191-201, 1995.

[31] Lauritzen, S.L., Spiegelhalter, D.J. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society B*, 50(2):157-224, 1988.

[32] Mauro, N., Esposito, F., Ferilli, S., Basile, T.M.A. Avoiding order effects in incremental learning. *Advances in Artificial Intelligence*, 3673:110–121, 2005.

[33] Mitchell, T.M. Machine Learning. McGraw-Hill Series in Computer Science. The McGraw-Hill Companies, Inc., 1997.

[34] Moore, A.W., Lee, M.S. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67-91, 1998.

[35] Moore, A.W., Schneider, J., Deng, K. Efficient locally weighted polynomial regression predictions. *Proceedings of the Fourteenth International Conference on Machine Learning*, 236-244, 1997.

[36] Neapolitan, R.E. Learning Bayesian Networks. Prentice Hall, Upper Saddle River, NJ, 2003.

[37] Pearl, J. Probabilistic Reasoning in Intelligent Systems. Morgan Kaufmann, San Mateo, CA, 1988.

[38] Pearl, J., Verma, T.S. A statistical semantics for causation. *Statistics and Computing*, 2:91-95, 1991.

[39] Provost, F.J., Kolluri, V. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131-169, 1999.

[40] Roure, J. Incremental methods for Bayesian network structure learning. Ph.D. Dissertation proposal. U. Politecnica de Cataluña. 2004.

[41] Roure, J. An incremental algorithm for tree-shaped Bayesian network learning. *Proceedings of the fifteenth European Conference of Artificial Intelligence*, 350-354, 2002.

[42] Roure, J. Incremental learning of tree augmented naïve Bayes classifiers. *Proceedings of the Eighth Ibero-American Conference of Artificial Intelligence*, 2527:32-41, 2002.

[43] Roure, J., Talavera, L. Robust incremental clustering with bad instance orderings: a new strategy. *Sixth Ibero-American Conference on Artificial Intelligence*, 136-147, 1998.

[44] Spiegelhalter, D.J., Lauritzen, S.L. Sequential updating of conditional probabilities on directed graphical structures. *Networks*, 20(5):579-605, 1990.

[45] Spirtes, P., Glymour, C., Scheimes, R. *Causation, Prediction and Search*. Springer-Verlag, New York, 116-125, 1993.

[46] Weiss, S.M., Kulikowski, C.A. Computer systems that learn. Morgan Kaufmann, San Mateo, CA, 1991.

[47] Witten, I.H., Frank, E. Data mining: practical machine learning tools and techniques (Second Edition). Morgan Kaufmann, San Mateo, CA, 2005.

[48] Yang, Y., Webb, G.I. Proportional k-Interval discretization for naïve-Bayes classifiers. *Lecture Notes in Computer Science*, 2167:564, 2001.

[49] Yang, Y., Webb, G.I. On why discretization works for naïve-Bayes classifiers. *Proceedings of the 16th AI 03*, Perth, Australia, 2903:440-452, 2003.

Apêndice I – Tabelas com os resultados do algoritmo AIP

Tabela 15: Primeiro cenário – Base de dados Alarm.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Alarm				
Conjunto inicial - 20000 casos				
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
20000	149,43 ± 0,90	97,28 ± 0,21	96,76 ± 0,08	96,97 ± 0,13

Tabela 16: Segundo cenário – Base de dados Alarm.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Alarm				
Conjunto inicial - 10000 casos				
20 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
10000	73,70 ± 0,70	80,68 ± 21,84	97,67 ± 1,53	90,76 ± 9,64
10500	0,04 ± 0,01	80,82 ± 21,95	97,63 ± 1,52	90,79 ± 9,67
11000	0,92 ± 0,32	80,68 ± 21,85	97,68 ± 1,54	90,76 ± 9,65
11500	1,24 ± 0,37	80,68 ± 21,86	97,69 ± 1,58	90,77 ± 9,65
12000	1,18 ± 0,25	81,42 ± 22,34	97,25 ± 1,29	90,81 ± 9,68
12500	1,22 ± 0,23	81,55 ± 22,49	97,18 ± 1,20	90,82 ± 9,69
13000	1,20 ± 0,02	81,49 ± 22,44	97,21 ± 1,22	90,82 ± 9,69
13500	1,35 ± 0,21	81,32 ± 22,33	97,31 ± 1,29	90,80 ± 9,68
14000	1,42 ± 0,21	81,32 ± 22,33	97,31 ± 1,29	90,80 ± 9,68
14500	1,41 ± 0,14	82,36 ± 23,02	96,64 ± 0,58	90,83 ± 9,70
15000	1,46 ± 0,12	81,28 ± 22,31	97,32 ± 1,25	90,80 ± 9,68
15500	1,47 ± 0,08	82,22 ± 22,94	96,71 ± 0,63	90,81 ± 9,69
16000	1,50 ± 0,03	81,38 ± 22,38	97,30 ± 1,26	90,82 ± 9,69
16500	1,50 ± 0,01	81,42 ± 22,42	97,27 ± 1,27	90,82 ± 9,69
17000	1,50 ± 0,07	81,38 ± 22,38	97,25 ± 1,19	90,79 ± 9,67
17500	1,50 ± 0,11	81,28 ± 22,65	97,26 ± 1,28	90,76 ± 9,79
18000	1,50 ± 0,16	82,12 ± 23,21	96,69 ± 0,60	90,76 ± 9,79
18500	1,46 ± 0,19	82,62 ± 22,40	96,69 ± 0,60	90,96 ± 9,46
19000	1,54 ± 0,25	82,11 ± 23,21	96,70 ± 0,61	90,76 ± 9,79
19500	1,32 ± 0,23	82,12 ± 23,20	96,68 ± 0,64	90,75 ± 9,81
20000	1,30 ± 0,28	82,22 ± 23,26	96,65 ± 0,61	90,78 ± 9,82
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
10000	75,02 ± 4,98	80,68 ± 21,84	97,67 ± 1,53	90,76 ± 9,64
10500	78,88 ± 5,83	80,82 ± 21,95	97,63 ± 1,52	90,79 ± 9,67
11000	82,74 ± 6,77	72,46 ± 23,45	97,31 ± 1,37	87,20 ± 10,16
11500	87,43 ± 5,76	71,91 ± 24,11	97,29 ± 1,38	86,96 ± 10,39
12000	90,92 ± 5,28	72,10 ± 24,22	97,06 ± 1,35	86,90 ± 10,50
12500	94,84 ± 5,37	72,14 ± 24,29	97,01 ± 1,26	86,89 ± 10,51
13000	99,07 ± 5,65	81,49 ± 22,44	97,21 ± 1,22	90,81 ± 9,69
13500	102,79 ± 5,49	81,32 ± 22,33	97,31 ± 1,29	90,80 ± 9,68
14000	106,51 ± 5,66	81,32 ± 22,33	97,31 ± 1,29	90,80 ± 9,68
14500	110,75 ± 5,18	82,36 ± 23,02	96,64 ± 0,58	90,83 ± 9,70
15000	114,03 ± 5,59	81,28 ± 22,31	97,32 ± 1,25	90,79 ± 9,68
15500	118,43 ± 5,46	82,22 ± 22,94	96,71 ± 0,63	90,81 ± 9,69
16000	122,82 ± 5,72	81,38 ± 22,38	97,30 ± 1,26	90,82 ± 9,69
16500	126,10 ± 5,60	81,42 ± 22,42	97,27 ± 1,27	90,82 ± 9,69
17000	131,34 ± 6,35	81,38 ± 22,38	97,25 ± 1,19	90,79 ± 9,67
17500	135,23 ± 6,92	81,28 ± 22,65	97,26 ± 1,28	90,75 ± 9,79
18000	139,33 ± 6,72	82,12 ± 23,21	96,69 ± 0,60	90,76 ± 9,79
18500	143,27 ± 7,25	82,62 ± 22,40	96,69 ± 0,60	90,96 ± 9,46
19000	147,92 ± 8,00	82,11 ± 23,21	96,70 ± 0,61	90,76 ± 9,79
19500	150,74 ± 8,14	82,12 ± 23,20	96,68 ± 0,64	90,75 ± 9,81

20000	156,38 ± 7,48	96,72 ± 0,27	96,84 ± 0,31	96,79 ± 0,15

Tabela 17: Terceiro cenário – Base de dados Alarm.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Alarm				
Conjunto inicial - 5000 casos				
30 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	37,28 ± 0,51	78,35 ± 24,34	96,21 ± 0,31	88,94 ± 10,07
5500	0,48 ± 0,03	77,72 ± 23,79	96,65 ± 1,12	88,94 ± 10,08
6000	0,46 ± 0,06	76,35 ± 22,59	97,51 ± 1,38	88,90 ± 10,01
6500	0,48 ± 0,12	77,74 ± 24,04	96,81 ± 1,03	89,05 ± 9,92
7000	0,40 ± 0,14	77,26 ± 24,79	96,87 ± 1,04	88,89 ± 10,19
7500	0,63 ± 0,26	77,88 ± 23,66	97,00 ± 0,84	89,22 ± 9,74
8000	0,45 ± 0,27	78,74 ± 22,52	96,77 ± 0,92	89,43 ± 9,46
8500	0,31 ± 0,25	78,61 ± 22,51	97,07 ± 0,83	89,56 ± 9,33
9000	0,22 ± 0,22	79,20 ± 21,69	96,93 ± 0,99	89,71 ± 9,20
9500	0,10 ± 0,01	78,72 ± 22,46	96,89 ± 0,99	89,49 ± 9,49
10000	0,15 ± 0,28	78,73 ± 22,48	96,88 ± 0,90	89,49 ± 9,46
10500	0,61 ± 0,51	78,75 ± 22,50	96,86 ± 0,90	89,49 ± 9,46
11000	1,03 ± 0,01	78,72 ± 22,54	96,86 ± 0,90	89,48 ± 9,47
11500	1,33 ± 0,41	78,72 ± 22,53	96,87 ± 0,90	89,49 ± 9,46
12000	1,28 ± 0,32	77,72 ± 23,56	96,99 ± 0,90	89,15 ± 9,94
12500	1,29 ± 0,28	78,37 ± 24,05	96,55 ± 0,33	89,15 ± 9,95
13000	1,34 ± 0,23	77,86 ± 23,66	96,88 ± 0,93	89,14 ± 9,93
13500	1,26 ± 0,02	77,75 ± 23,80	96,89 ± 0,93	89,10 ± 9,98
14000	1,35 ± 0,11	77,74 ± 23,79	96,90 ± 0,92	89,10 ± 9,99
14500	1,41 ± 0,12	77,74 ± 23,79	96,89 ± 0,93	89,10 ± 10,00
15000	1,48 ± 0,11	77,91 ± 24,83	96,48 ± 0,22	88,92 ± 10,22
15500	1,52 ± 0,06	77,09 ± 25,89	96,49 ± 0,22	88,60 ± 10,64
16000	1,49 ± 0,01	77,85 ± 24,91	96,49 ± 0,21	88,91 ± 10,24
16500	1,50 ± 0,05	78,13 ± 24,57	96,49 ± 0,21	89,02 ± 10,10
17000	1,45 ± 0,07	78,12 ± 24,57	96,49 ± 0,21	89,02 ± 10,10
17500	1,42 ± 0,11	78,12 ± 24,57	96,48 ± 0,23	89,01 ± 10,11
18000	1,44 ± 0,17	78,11 ± 24,57	96,48 ± 0,22	89,01 ± 10,11
18500	1,51 ± 0,22	78,13 ± 24,54	96,45 ± 0,25	89,00 ± 10,12
19000	1,33 ± 0,23	78,13 ± 24,54	96,46 ± 0,25	89,00 ± 10,12
19500	1,32 ± 0,26	78,13 ± 24,54	96,46 ± 0,25	89,00 ± 10,12
20000	1,15 ± 0,02	78,11 ± 24,61	96,70 ± 0,29	89,13 ± 9,97
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	38,30 ± 6,39	78,35 ± 24,34	96,21 ± 0,31	88,94 ± 10,07
5500	42,56 ± 8,03	77,72 ± 23,79	96,65 ± 1,12	88,94 ± 10,08
6000	46,20 ± 8,49	76,35 ± 22,59	97,51 ± 1,38	88,90 ± 10,01
6500	49,33 ± 6,55	77,74 ± 24,04	96,81 ± 1,03	89,05 ± 9,92
7000	52,49 ± 6,15	77,26 ± 24,79	96,87 ± 1,04	88,89 ± 10,20
7500	56,01 ± 4,94	77,88 ± 23,66	97,00 ± 0,84	89,22 ± 9,74
8000	60,29 ± 6,83	78,74 ± 22,52	96,77 ± 0,92	89,43 ± 9,46
8500	63,64 ± 6,44	91,23 ± 13,03	97,57 ± 1,10	94,99 ± 5,61
9000	67,55 ± 6,11	92,23 ± 13,37	97,05 ± 0,90	95,09 ± 5,63
9500	71,92 ± 7,68	92,27 ± 13,41	97,01 ± 0,91	95,08 ± 5,62

10000	76,50 ± 9,82	92,29 ± 13,40	97,00 ± 0,83	95,08 ± 5,62
10500	80,69 ± 12,46	78,75 ± 22,50	96,86 ± 0,90	89,49 ± 9,46
11000	84,27 ± 10,37	78,72 ± 22,54	96,86 ± 0,90	89,48 ± 9,47
11500	87,11 ± 11,94	78,72 ± 22,53	96,87 ± 0,90	89,49 ± 9,46
12000	91,60 ± 11,27	77,72 ± 23,56	96,99 ± 0,90	89,15 ± 9,94
12500	95,54 ± 11,69	78,37 ± 24,05	96,55 ± 0,33	89,15 ± 9,95
13000	101,81 ± 14,79	82,07 ± 22,74	96,93 ± 0,89	90,88 ± 9,49
13500	108,54 ± 20,19	77,75 ± 23,80	96,89 ± 0,93	89,10 ± 9,99
14000	114,91 ± 24,91	92,24 ± 13,39	97,08 ± 0,84	95,11 ± 5,63
14500	116,50 ± 19,72	96,54 ± 1,54	97,06 ± 0,82	96,85 ± 0,18
15000	121,13 ± 22,67	95,78 ± 2,15	96,61 ± 0,09	96,27 ± 0,89
15500	124,94 ± 23,56	97,13 ± 0,21	96,71 ± 0,22	96,88 ± 0,18
16000	126,86 ± 19,01	97,08 ± 0,29	96,72 ± 0,16	96,87 ± 0,18
16500	131,93 ± 21,72	97,08 ± 0,29	96,72 ± 0,17	96,86 ± 0,19
17000	136,88 ± 24,81	97,13 ± 0,21	96,69 ± 0,20	96,87 ± 0,18
17500	137,62 ± 18,65	97,16 ± 0,23	96,64 ± 0,15	96,85 ± 0,17
18000	145,85 ± 28,18	87,77 ± 20,00	96,53 ± 0,33	92,96 ± 8,32
18500	152,73 ± 32,00	97,17 ± 0,24	96,64 ± 0,14	96,85 ± 0,17
19000	153,34 ± 24,99	97,15 ± 0,26	96,66 ± 0,11	96,85 ± 0,17
19500	151,90 ± 14,82	97,15 ± 0,26	96,66 ± 0,10	96,86 ± 0,17
20000	160,73 ± 30,24	96,01 ± 1,78	97,29 ± 1,08	96,77 ± 0,17

Tabela 18: Quarto cenário – Base de dados Alarm.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Alarm				
Conjunto inicial - 3000 casos				
34 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	22,17 ± 0,41	82,21 ± 20,43	96,87 ± 1,46	90,90 ± 8,96
3500	0,41 ± 0,13	82,19 ± 20,36	96,92 ± 1,47	90,93 ± 8,96
4000	0,44 ± 0,09	82,06 ± 20,56	97,01 ± 1,58	90,92 ± 9,06
4500	0,48 ± 0,06	81,90 ± 20,70	97,14 ± 1,41	90,94 ± 9,03
5000	0,50 ± 0,01	81,87 ± 20,68	97,15 ± 1,53	90,93 ± 9,02
5500	0,50 ± 0,03	82,07 ± 20,83	97,07 ± 1,44	90,96 ± 9,02
6000	0,46 ± 0,08	81,22 ± 22,03	97,20 ± 1,34	90,70 ± 9,49
6500	0,48 ± 0,13	81,18 ± 22,02	97,26 ± 1,30	90,72 ± 9,50
7000	0,40 ± 0,15	81,24 ± 22,06	97,18 ± 1,29	90,69 ± 9,49
7500	0,46 ± 0,23	81,57 ± 22,27	97,00 ± 1,24	90,72 ± 9,51
8000	0,23 ± 0,01	81,57 ± 22,29	97,00 ± 1,22	90,72 ± 9,50
8500	0,24 ± 0,20	82,13 ± 21,39	97,01 ± 1,22	90,96 ± 9,14
9000	0,29 ± 0,31	82,05 ± 21,40	97,05 ± 1,21	90,95 ± 9,15
9500	0,09 ± 0,01	81,49 ± 22,30	97,05 ± 1,21	90,72 ± 9,52
10000	0,17 ± 0,41	80,67 ± 23,58	97,06 ± 1,21	90,39 ± 10,03
10500	0,75 ± 0,52	80,67 ± 23,58	97,06 ± 1,21	90,39 ± 10,03
11000	1,14 ± 0,27	80,69 ± 23,59	97,05 ± 1,21	90,39 ± 10,04
11500	1,10 ± 0,02	80,70 ± 23,60	97,06 ± 1,21	90,40 ± 10,04
12000	1,21 ± 0,22	80,67 ± 23,60	97,09 ± 1,20	90,41 ± 10,04
12500	1,32 ± 0,26	80,65 ± 23,59	97,12 ± 1,25	90,42 ± 10,05
13000	1,45 ± 0,28	80,71 ± 23,64	97,08 ± 1,27	90,42 ± 10,05
13500	1,42 ± 0,22	80,48 ± 24,02	97,08 ± 1,27	90,33 ± 10,21
14000	1,46 ± 0,18	80,44 ± 23,98	97,09 ± 1,19	90,32 ± 10,19
14500	1,55 ± 0,13	80,41 ± 23,96	97,13 ± 1,24	90,33 ± 10,20
15000	1,46 ± 0,08	80,50 ± 24,03	97,08 ± 1,28	90,34 ± 10,20
15500	1,49 ± 0,03	81,27 ± 22,61	97,13 ± 1,26	90,68 ± 9,65
16000	1,50 ± 0,03	80,46 ± 23,90	97,14 ± 1,26	90,35 ± 10,17
16500	1,50 ± 0,08	80,54 ± 23,95	97,07 ± 1,22	90,34 ± 10,17
17000	1,54 ± 0,12	80,48 ± 23,91	97,11 ± 1,20	90,35 ± 10,17
17500	1,43 ± 0,15	80,47 ± 23,90	97,13 ± 1,20	90,35 ± 10,17
18000	1,38 ± 0,17	80,45 ± 23,94	97,13 ± 1,19	90,35 ± 10,18
18500	1,35 ± 0,21	81,24 ± 22,70	97,12 ± 1,20	90,66 ± 9,67
19000	1,44 ± 0,30	81,24 ± 22,70	97,13 ± 1,20	90,66 ± 9,68
19500	1,35 ± 0,32	81,23 ± 22,70	97,13 ± 1,20	90,66 ± 9,68
20000	1,18 ± 0,24	80,44 ± 23,97	97,12 ± 1,21	90,33 ± 10,20
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	23,45 ± 0,46	82,21 ± 20,43	96,87 ± 1,46	90,91 ± 8,96
3500	27,48 ± 0,69	82,19 ± 20,36	96,92 ± 1,47	90,93 ± 8,96
4000	31,32 ± 0,62	82,06 ± 20,56	97,01 ± 1,58	90,92 ± 9,06
4500	35,84 ± 0,72	81,90 ± 20,70	97,14 ± 1,41	90,94 ± 9,03
5000	39,82 ± 0,88	81,87 ± 20,68	97,15 ± 1,53	90,93 ± 9,02
5500	43,86 ± 0,82	82,07 ± 20,83	97,07 ± 1,44	90,96 ± 9,02

6000	47,77 ± 1,48	81,22 ± 22,03	97,20 ± 1,34	90,70 ± 9,49
6500	52,77 ± 1,45	81,18 ± 22,02	97,26 ± 1,30	90,72 ± 9,50
7000	56,27 ± 1,23	81,24 ± 22,06	97,18 ± 1,29	90,70 ± 9,49
7500	60,58 ± 1,86	95,85 ± 1,99	97,19 ± 1,07	96,65 ± 0,18
8000	64,65 ± 1,62	91,18 ± 14,43	97,14 ± 1,10	94,72 ± 6,06
8500	69,07 ± 2,35	95,85 ± 1,94	97,20 ± 1,05	96,65 ± 0,17
9000	72,98 ± 1,91	82,05 ± 21,40	97,05 ± 1,21	90,95 ± 9,15
9500	76,00 ± 1,54	81,49 ± 22,30	97,05 ± 1,21	90,72 ± 9,52
10000	79,68 ± 1,55	75,78 ± 24,86	97,00 ± 1,25	88,37 ± 10,63
10500	85,42 ± 5,14	95,76 ± 1,96	97,24 ± 1,05	96,64 ± 0,19
11000	87,91 ± 1,83	90,91 ± 15,06	97,17 ± 1,10	94,63 ± 6,32
11500	91,55 ± 1,78	95,78 ± 1,98	97,24 ± 1,05	96,64 ± 0,20
12000	95,92 ± 2,15	90,87 ± 15,19	97,19 ± 1,09	94,62 ± 6,38
12500	100,16 ± 1,73	90,85 ± 15,18	97,23 ± 1,15	94,63 ± 6,38
13000	105,22 ± 3,14	95,80 ± 2,05	97,26 ± 1,11	96,67 ± 0,19
13500	109,17 ± 2,52	95,79 ± 2,04	97,26 ± 1,11	96,67 ± 0,19
14000	113,10 ± 2,58	95,69 ± 1,92	97,32 ± 0,99	96,66 ± 0,20
14500	118,04 ± 2,41	95,67 ± 1,95	97,36 ± 1,04	96,67 ± 0,19
15000	122,08 ± 3,16	95,75 ± 2,02	97,31 ± 1,10	96,68 ± 0,18
15500	126,22 ± 3,35	95,70 ± 1,97	97,37 ± 1,06	96,69 ± 0,18
16000	130,52 ± 2,80	95,67 ± 1,96	97,37 ± 1,06	96,68 ± 0,18
16500	134,66 ± 2,67	95,75 ± 1,97	97,30 ± 1,03	96,67 ± 0,20
17000	139,27 ± 2,93	95,69 ± 1,92	97,34 ± 1,00	96,67 ± 0,20
17500	142,86 ± 4,00	95,68 ± 1,91	97,35 ± 0,99	96,67 ± 0,20
18000	146,68 ± 3,96	95,68 ± 1,92	97,35 ± 0,99	96,67 ± 0,20
18500	150,66 ± 3,31	95,71 ± 1,93	97,34 ± 1,00	96,68 ± 0,20
19000	155,98 ± 5,20	95,71 ± 1,93	97,35 ± 1,00	96,68 ± 0,20
19500	160,36 ± 5,90	95,70 ± 1,93	97,35 ± 1,00	96,68 ± 0,20
20000	162,97 ± 4,54	95,70 ± 1,93	97,35 ± 1,00	96,68 ± 0,20

Tabela 19: Quinto cenário – Base de dados Alarm.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Alarm				
Conjunto inicial - 1000 casos				
38 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	7,12 ± 1,01	78,86 ± 21,86	95,67 ± 0,57	88,83 ± 8,69
1500	0,22 ± 0,22	77,39 ± 20,62	96,69 ± 1,57	88,84 ± 8,69
2000	0,19 ± 0,02	77,27 ± 20,51	96,96 ± 1,63	88,95 ± 8,78
2500	0,54 ± 0,28	77,34 ± 20,57	96,93 ± 1,56	88,96 ± 8,79
3000	0,38 ± 0,20	77,34 ± 20,57	96,99 ± 1,57	89,00 ± 8,82
3500	0,37 ± 0,12	77,18 ± 20,43	97,13 ± 1,72	89,01 ± 8,84
4000	0,42 ± 0,11	77,09 ± 20,36	97,20 ± 1,67	89,02 ± 8,84
4500	0,50 ± 0,10	77,14 ± 20,40	97,28 ± 1,60	89,08 ± 8,90
5000	0,53 ± 0,06	77,22 ± 20,47	97,17 ± 1,77	89,05 ± 8,88
5500	0,46 ± 0,03	77,52 ± 20,77	97,01 ± 1,53	89,08 ± 8,90
6000	0,47 ± 0,05	75,04 ± 23,96	97,09 ± 1,45	88,12 ± 10,22
6500	0,46 ± 0,08	77,58 ± 20,82	97,05 ± 1,38	89,13 ± 8,93
7000	0,52 ± 0,27	77,58 ± 20,82	97,15 ± 1,30	89,19 ± 8,93
7500	0,43 ± 0,17	77,58 ± 20,82	97,15 ± 1,30	89,19 ± 8,93
8000	0,34 ± 0,18	77,58 ± 20,82	97,16 ± 1,29	89,19 ± 8,93
8500	0,33 ± 0,22	77,51 ± 20,76	97,16 ± 1,30	89,17 ± 8,90
9000	0,29 ± 0,24	77,45 ± 20,71	97,24 ± 1,34	89,19 ± 8,92
9500	0,15 ± 0,07	77,44 ± 20,70	97,19 ± 1,34	89,15 ± 8,89
10000	0,31 ± 0,37	77,49 ± 20,74	97,14 ± 1,27	89,15 ± 8,89
10500	0,39 ± 0,45	77,44 ± 20,70	97,16 ± 1,38	89,14 ± 8,88
11000	0,38 ± 0,50	74,96 ± 23,89	97,18 ± 1,40	88,14 ± 10,19
11500	0,85 ± 0,60	74,89 ± 23,82	97,26 ± 1,36	88,16 ± 10,20
12000	1,23 ± 0,57	74,85 ± 23,78	97,32 ± 1,40	88,17 ± 10,21
12500	1,15 ± 0,50	74,83 ± 23,77	97,32 ± 1,41	88,17 ± 10,20
13000	1,31 ± 0,32	74,89 ± 23,82	97,28 ± 1,34	88,17 ± 10,20
13500	1,27 ± 0,22	75,95 ± 24,75	96,56 ± 0,26	88,17 ± 10,21
14000	1,38 ± 0,26	75,91 ± 24,71	96,59 ± 0,31	88,18 ± 10,21
14500	1,42 ± 0,23	75,90 ± 24,70	96,61 ± 0,31	88,19 ± 10,22
15000	1,40 ± 0,17	75,89 ± 24,69	96,59 ± 0,29	88,17 ± 10,20
15500	1,48 ± 0,16	75,91 ± 24,71	96,59 ± 0,28	88,17 ± 10,21
16000	1,47 ± 0,13	75,90 ± 24,70	96,60 ± 0,30	88,18 ± 10,21
16500	1,56 ± 0,11	75,88 ± 24,68	96,60 ± 0,30	88,17 ± 10,20
17000	1,46 ± 0,11	75,93 ± 24,66	96,51 ± 0,42	88,14 ± 10,27
17500	1,42 ± 0,11	75,92 ± 24,66	96,51 ± 0,42	88,13 ± 10,27
18000	1,46 ± 0,16	75,91 ± 24,65	96,51 ± 0,41	88,13 ± 10,27
18500	1,37 ± 0,15	75,91 ± 24,65	96,51 ± 0,41	88,13 ± 10,27
19000	1,35 ± 0,18	75,91 ± 24,65	96,51 ± 0,41	88,13 ± 10,27
19500	1,34 ± 0,23	75,88 ± 24,68	96,60 ± 0,30	88,17 ± 10,21
20000	1,45 ± 0,33	76,05 ± 24,78	96,46 ± 0,37	88,16 ± 10,29
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	6,58 ± 0,89	78,86 ± 21,86	95,67 ± 0,57	88,83 ± 8,70
1500	9,38 ± 1,26	77,50 ± 20,70	97,39 ± 1,05	89,29 ± 9,04

2000	13,81 ± 0,44	95,55 ± 2,01	96,93 ± 1,14	96,37 ± 0,68
2500	17,25 ± 0,60	95,68 ± 1,84	97,22 ± 1,08	96,59 ± 0,13
3000	20,75 ± 0,89	87,18 ± 17,16	97,21 ± 1,23	93,13 ± 7,42
3500	24,53 ± 1,05	87,02 ± 17,09	97,37 ± 1,40	93,16 ± 7,43
4000	27,58 ± 0,98	78,29 ± 20,74	97,32 ± 1,48	89,58 ± 9,09
4500	31,27 ± 0,86	78,30 ± 20,81	97,36 ± 1,46	89,61 ± 9,12
5000	34,77 ± 0,50	78,30 ± 20,81	97,37 ± 1,49	89,61 ± 9,12
5500	37,73 ± 1,63	78,64 ± 21,11	97,18 ± 1,40	89,63 ± 9,14
6000	41,25 ± 1,75	70,25 ± 20,53	97,02 ± 1,50	86,12 ± 9,12
6500	44,44 ± 1,85	78,72 ± 21,18	97,15 ± 1,32	89,65 ± 9,15
7000	48,51 ± 0,92	78,82 ± 21,08	97,15 ± 1,32	89,69 ± 9,09
7500	52,86 ± 1,17	78,74 ± 21,18	97,13 ± 1,34	89,64 ± 9,15
8000	55,71 ± 1,73	70,32 ± 20,60	96,96 ± 1,44	86,12 ± 9,13
8500	59,68 ± 1,35	78,65 ± 21,10	97,13 ± 1,35	89,61 ± 9,12
9000	62,69 ± 1,80	78,56 ± 21,12	97,17 ± 1,41	89,60 ± 9,15
9500	66,33 ± 1,87	78,56 ± 21,09	97,17 ± 1,37	89,59 ± 9,12
10000	70,64 ± 2,15	78,31 ± 21,45	97,20 ± 1,24	89,51 ± 9,20
10500	73,71 ± 2,23	78,28 ± 21,38	97,24 ± 1,34	89,52 ± 9,21
11000	76,79 ± 2,22	78,28 ± 21,38	97,24 ± 1,37	89,53 ± 9,22
11500	81,68 ± 1,30	78,20 ± 21,31	97,28 ± 1,36	89,52 ± 9,22
12000	85,55 ± 1,14	78,15 ± 21,28	97,32 ± 1,42	89,52 ± 9,22
12500	89,19 ± 1,81	77,00 ± 22,71	97,33 ± 1,42	89,05 ± 9,81
13000	92,75 ± 1,54	77,06 ± 22,75	97,30 ± 1,34	89,06 ± 9,81
13500	96,29 ± 1,36	78,14 ± 23,62	96,55 ± 0,34	89,06 ± 9,81
14000	99,36 ± 1,65	78,09 ± 23,59	96,60 ± 0,38	89,07 ± 9,82
14500	103,27 ± 1,92	78,11 ± 23,60	96,61 ± 0,37	89,08 ± 9,82
15000	106,97 ± 1,60	76,96 ± 25,09	96,59 ± 0,34	88,60 ± 10,41
15500	109,69 ± 1,64	78,12 ± 23,61	96,61 ± 0,30	89,08 ± 9,79
16000	113,71 ± 2,09	78,11 ± 23,60	96,62 ± 0,31	89,08 ± 9,79
16500	117,81 ± 1,94	78,11 ± 23,60	96,62 ± 0,31	89,08 ± 9,79
17000	120,47 ± 1,61	77,89 ± 23,85	96,63 ± 0,31	89,00 ± 9,89
17500	124,45 ± 2,21	77,89 ± 23,85	96,63 ± 0,31	89,00 ± 9,89
18000	127,89 ± 2,18	77,83 ± 23,93	96,63 ± 0,32	88,98 ± 9,92
18500	131,32 ± 2,45	76,75 ± 25,33	96,63 ± 0,32	88,54 ± 10,49
19000	134,90 ± 2,64	77,46 ± 24,44	96,62 ± 0,31	88,82 ± 10,13
19500	138,62 ± 1,91	77,45 ± 24,47	96,63 ± 0,31	88,82 ± 10,14
20000	141,92 ± 1,92	96,77 ± 0,19	96,83 ± 0,06	96,80 ± 0,11

Tabela 20: Primeiro cenário – Base de dados Ásia.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Asia				
Conjunto inicial - 10000 casos				
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
10000	0,50 ± 0,18	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00

Tabela 21: Segundo cenário – Base de dados Ásia.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Asia				
Conjunto inicial - 5000 casos				
10 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	0,24 ± 0,21	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
5500	0,05 ± 0,01	100,00 ± 0,00	99,61 ± 0,21	99,97 ± 0,02
6000	0,06 ± 0,01	100,00 ± 0,00	99,61 ± 0,21	99,97 ± 0,02
6500	0,24 ± 0,37	100,00 ± 0,00	99,61 ± 0,21	99,97 ± 0,02
7000	0,15 ± 0,28	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
7500	0,07 ± 0,02	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
8000	0,09 ± 0,02	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
8500	0,17 ± 0,26	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
9000	0,17 ± 0,26	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
9500	0,25 ± 0,35	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
10000	0,09 ± 0,01	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	0,22 ± 0,20	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
5500	0,37 ± 0,31	100,00 ± 0,00	99,61 ± 0,21	99,97 ± 0,02
6000	0,37 ± 0,30	100,00 ± 0,00	99,61 ± 0,21	99,97 ± 0,02
6500	0,21 ± 0,01	100,00 ± 0,00	99,61 ± 0,21	99,97 ± 0,02
7000	0,44 ± 0,29	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
7500	0,50 ± 0,28	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
8000	0,25 ± 0,01	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
8500	0,30 ± 0,15	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
9000	0,50 ± 0,24	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
9500	0,50 ± 0,22	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01
10000	0,43 ± 0,19	100,00 ± 0,00	99,81 ± 0,10	99,98 ± 0,01

Tabela 22: Terceiro cenário – Base de dados Ásia.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Asia				
Conjunto inicial - 3000 casos				
14 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	0,18 ± 0,25	100,00 ± 0,00	99,57 ± 0,41	99,96 ± 0,03
3500	0,04 ± 0,01	100,00 ± 0,00	99,57 ± 0,41	99,96 ± 0,03
4000	0,05 ± 0,01	100,00 ± 0,00	99,71 ± 0,11	99,98 ± 0,01
4500	0,14 ± 0,29	100,00 ± 0,00	99,71 ± 0,11	99,98 ± 0,01
5000	0,05 ± 0,01	100,00 ± 0,00	99,71 ± 0,11	99,98 ± 0,01
5500	0,06 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6000	0,15 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6500	0,15 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7000	0,06 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7500	0,15 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8000	0,16 ± 0,27	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8500	0,08 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9000	0,08 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9500	0,08 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
10000	0,08 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	0,18 ± 0,25	100,00 ± 0,00	99,57 ± 0,41	99,96 ± 0,03
3500	0,19 ± 0,25	100,00 ± 0,00	99,57 ± 0,41	99,96 ± 0,03
4000	0,21 ± 0,24	100,00 ± 0,00	99,71 ± 0,11	99,98 ± 0,01
4500	0,29 ± 0,30	100,00 ± 0,00	99,71 ± 0,11	99,98 ± 0,01
5000	0,29 ± 0,29	100,00 ± 0,00	99,71 ± 0,11	99,98 ± 0,01
5500	0,30 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6000	0,25 ± 0,20	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6500	0,38 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7000	0,38 ± 0,27	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7500	0,40 ± 0,26	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8000	0,39 ± 0,25	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8500	0,36 ± 0,20	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9000	0,41 ± 0,21	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9500	0,41 ± 0,20	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
10000	0,39 ± 0,16	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00

Tabela 23: Quarto cenário – Base de dados Ásia.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Asia				
Conjunto inicial - 1000 casos				
18 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	0,14 ± 0,29	100,00 ± 0,00	99,37 ± 0,51	99,95 ± 0,04
1500	0,12 ± 0,30	100,00 ± 0,00	99,22 ± 0,38	99,94 ± 0,03
2000	0,03 ± 0,01	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
2500	0,03 ± 0,01	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
3000	0,13 ± 0,29	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
3500	0,04 ± 0,01	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
4000	0,04 ± 0,01	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
4500	0,05 ± 0,01	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
5000	0,05 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
5500	0,14 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6000	0,23 ± 0,37	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6500	0,24 ± 0,37	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7000	0,15 ± 0,28	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7500	0,15 ± 0,27	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8000	0,07 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8500	0,07 ± 0,01	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9000	0,16 ± 0,27	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9500	0,17 ± 0,27	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
10000	0,17 ± 0,27	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	0,13 ± 0,29	100,00 ± 0,00	99,37 ± 0,51	99,95 ± 0,04
1500	0,05 ± 0,01	100,00 ± 0,00	99,22 ± 0,38	99,94 ± 0,03
2000	0,16 ± 0,27	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
2500	0,17 ± 0,26	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
3000	0,18 ± 0,25	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
3500	0,27 ± 0,33	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
4000	0,13 ± 0,01	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
4500	0,22 ± 0,22	100,00 ± 0,00	99,66 ± 0,13	99,97 ± 0,01
5000	0,23 ± 0,22	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
5500	0,23 ± 0,20	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6000	0,23 ± 0,17	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
6500	0,28 ± 0,19	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7000	0,29 ± 0,18	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
7500	0,35 ± 0,23	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8000	0,48 ± 0,26	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
8500	0,38 ± 0,21	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9000	0,38 ± 0,20	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
9500	0,33 ± 0,13	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00
10000	0,46 ± 0,20	100,00 ± 0,00	99,76 ± 0,00	99,98 ± 0,00

Tabela 24: Primeiro cenário – Base de dados Credit.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Credit				
Conjunto inicial - 10000 casos				
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
10000	3,51 ± 0,06	72,54 ± 0,34	56,87 ± 2,03	65,34 ± 0,75

Tabela 25: Segundo cenário – Base de dados Credit.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Credit				
Conjunto inicial - 5000 casos				
10 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	1,44 ± 0,28	70,88 ± 2,43	59,56 ± 3,10	65,68 ± 0,11
5500	0,27 ± 0,33	69,62 ± 1,42	61,16 ± 0,13	65,73 ± 0,71
6000	0,35 ± 0,37	69,48 ± 1,70	61,97 ± 1,83	66,03 ± 0,08
6500	0,20 ± 0,24	69,48 ± 1,70	61,97 ± 1,83	66,03 ± 0,08
7000	0,21 ± 0,23	70,74 ± 0,95	60,58 ± 1,10	66,07 ± 0,01
7500	0,15 ± 0,01	70,74 ± 0,95	60,58 ± 1,10	66,07 ± 0,01
8000	0,37 ± 0,33	72,13 ± 0,22	58,17 ± 0,17	65,72 ± 0,20
8500	0,43 ± 0,35	72,13 ± 0,22	58,17 ± 0,17	65,72 ± 0,20
9000	0,25 ± 0,20	72,13 ± 0,22	58,17 ± 0,17	65,72 ± 0,20
9500	0,26 ± 0,20	72,64 ± 0,05	54,51 ± 2,10	64,31 ± 0,94
10000	0,43 ± 0,32	72,64 ± 0,05	54,51 ± 2,10	64,31 ± 0,94
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	1,49 ± 0,17	70,88 ± 2,43	59,56 ± 3,10	65,68 ± 0,11
5500	1,24 ± 0,21	69,62 ± 1,42	61,16 ± 0,13	65,73 ± 0,71
6000	1,74 ± 0,50	69,48 ± 1,70	61,97 ± 1,83	66,03 ± 0,08
6500	2,30 ± 0,27	69,48 ± 1,70	61,97 ± 1,83	66,03 ± 0,08
7000	2,51 ± 0,19	70,74 ± 0,95	60,58 ± 1,10	66,07 ± 0,01
7500	2,48 ± 0,03	70,74 ± 0,95	60,58 ± 1,10	66,07 ± 0,01
8000	2,40 ± 0,15	72,13 ± 0,22	58,17 ± 0,17	65,72 ± 0,19
8500	2,44 ± 0,32	72,13 ± 0,22	58,17 ± 0,17	65,72 ± 0,19
9000	2,34 ± 0,47	72,13 ± 0,22	58,17 ± 0,17	65,72 ± 0,19
9500	3,35 ± 0,35	72,64 ± 0,05	54,51 ± 2,10	64,31 ± 0,94
10000	3,51 ± 0,21	72,64 ± 0,05	54,51 ± 2,10	64,31 ± 0,94

Tabela 26: Terceiro cenário – Base de dados Credit.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Credit				
Conjunto inicial - 3000 casos				
14 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	0,69 ± 0,41	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
3500	0,24 ± 0,36	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
4000	0,16 ± 0,27	72,79 ± 2,28	63,10 ± 6,79	68,34 ± 4,35
4500	0,09 ± 0,01	72,79 ± 2,28	63,10 ± 6,79	68,34 ± 4,35
5000	0,10 ± 0,03	72,79 ± 2,28	63,10 ± 6,79	68,34 ± 4,35
5500	0,19 ± 0,25	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
6000	0,17 ± 0,21	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
6500	0,28 ± 0,33	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
7000	0,13 ± 0,02	73,27 ± 1,77	63,65 ± 6,21	68,85 ± 3,81
7500	0,35 ± 0,35	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
8000	0,21 ± 0,22	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
8500	0,36 ± 0,34	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
9000	0,29 ± 0,30	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
9500	0,23 ± 0,21	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
10000	0,23 ± 0,20	73,55 ± 1,48	61,58 ± 8,40	68,04 ± 4,66
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	0,61 ± 0,38	73,55 ± 1,48	61,58 ± 8,40	68,05 ± 4,66
3500	1,43 ± 0,34	73,12 ± 1,03	53,87 ± 0,27	64,28 ± 0,69
4000	1,38 ± 0,11	72,37 ± 1,83	55,40 ± 1,33	64,57 ± 0,38
4500	1,51 ± 0,02	72,37 ± 1,83	55,40 ± 1,33	64,57 ± 0,38
5000	1,40 ± 0,14	72,04 ± 1,48	57,68 ± 1,08	65,44 ± 1,30
5500	1,17 ± 0,02	72,79 ± 0,69	56,16 ± 2,68	65,15 ± 1,60
6000	1,81 ± 0,42	69,59 ± 2,69	59,97 ± 6,70	65,17 ± 1,62
6500	2,17 ± 0,03	74,05 ± 2,01	53,46 ± 0,16	64,59 ± 1,01
7000	2,39 ± 0,15	73,22 ± 1,71	57,66 ± 0,09	66,07 ± 0,89
7500	2,49 ± 0,03	73,49 ± 1,42	55,59 ± 2,09	65,27 ± 1,73
8000	2,48 ± 0,17	73,49 ± 1,42	55,59 ± 2,09	65,27 ± 1,73
8500	2,44 ± 0,34	72,79 ± 0,69	56,16 ± 2,68	65,15 ± 1,60
9000	2,53 ± 0,53	73,49 ± 1,42	55,59 ± 2,09	65,27 ± 1,73
9500	3,43 ± 0,34	73,49 ± 1,42	55,59 ± 2,09	65,27 ± 1,73
10000	3,45 ± 0,15	72,79 ± 0,69	56,16 ± 2,68	65,15 ± 1,60

Tabela 27: Quarto cenário – Base de dados Credit.
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Credit				
Conjunto inicial - 1000 casos				
18 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	0,40 ± 0,26	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
1500	0,04 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
2000	0,14 ± 0,29	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
2500	0,14 ± 0,28	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
3000	0,15 ± 0,28	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
3500	0,07 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
4000	0,07 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
4500	0,08 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
5000	0,17 ± 0,26	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
5500	0,18 ± 0,26	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
6000	0,18 ± 0,25	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
6500	0,11 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
7000	0,27 ± 0,33	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
7500	0,20 ± 0,24	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
8000	0,13 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
8500	0,43 ± 0,37	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
9000	0,15 ± 0,01	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
9500	0,29 ± 0,29	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
10000	0,23 ± 0,22	75,82 ± 0,30	70,01 ± 0,27	73,15 ± 0,04
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	0,34 ± 0,24	75,82 ± 0,30	70,01 ± 0,27	73,16 ± 0,04
1500	0,50 ± 0,17	75,82 ± 0,30	70,01 ± 0,27	73,16 ± 0,04
2000	0,52 ± 0,06	75,82 ± 0,30	70,01 ± 0,27	73,16 ± 0,04
2500	0,51 ± 0,06	75,82 ± 0,30	70,01 ± 0,27	73,16 ± 0,04
3000	0,74 ± 0,53	72,26 ± 4,91	68,48 ± 1,71	70,53 ± 3,44
3500	1,26 ± 0,21	66,42 ± 0,12	63,65 ± 2,45	65,15 ± 1,19
4000	1,48 ± 0,16	67,36 ± 0,69	61,69 ± 4,13	64,75 ± 1,53
4500	1,50 ± 0,03	72,70 ± 0,19	58,69 ± 0,42	66,26 ± 0,09
5000	1,39 ± 0,15	72,70 ± 0,19	58,69 ± 0,42	66,26 ± 0,09
5500	1,30 ± 0,30	73,63 ± 0,99	56,73 ± 2,11	65,87 ± 0,43
6000	1,81 ± 0,42	72,70 ± 0,19	58,69 ± 0,42	66,26 ± 0,09
6500	2,30 ± 0,27	72,21 ± 0,23	61,24 ± 1,78	67,17 ± 0,69
7000	2,39 ± 0,16	72,21 ± 0,23	61,24 ± 1,78	67,17 ± 0,69
7500	2,47 ± 0,13	73,63 ± 0,99	56,73 ± 2,11	65,87 ± 0,43
8000	2,41 ± 0,19	73,63 ± 0,99	56,73 ± 2,11	65,87 ± 0,43
8500	2,30 ± 0,28	73,63 ± 0,99	56,73 ± 2,11	65,87 ± 0,43
9000	2,61 ± 0,69	73,63 ± 0,99	56,73 ± 2,11	65,87 ± 0,43
9500	3,25 ± 0,19	73,94 ± 0,59	55,18 ± 0,11	65,32 ± 0,27
10000	3,43 ± 0,17	73,94 ± 0,59	55,18 ± 0,11	65,32 ± 0,27

Tabela 28: Primeiro cenário – Base de dados Engine Fuel System
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Engine Fuel System				
Conjunto inicial - 10000 casos				
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
10000	0,48 ± 0,10	99,48 ± 0,00	99,84 ± 0,01	99,66 ± 0,01

Tabela 29: Segundo cenário – Base de dados Engine Fuel System
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Engine Fuel System				
Conjunto inicial - 5000 casos				
10 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	0,67 ± 0,24	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
5500	0,06 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
6000	0,07 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
6500	0,06 ± 0,00	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
7000	0,07 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
7500	0,08 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
8000	0,08 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
8500	0,59 ± 0,43	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
9000	0,09 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
9500	0,10 ± 0,01	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
10000	0,09 ± 0,02	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
5000	0,51 ± 0,32	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
5500	0,32 ± 0,25	99,34 ± 0,11	99,82 ± 0,02	99,58 ± 0,04
6000	0,29 ± 0,18	99,34 ± 0,11	99,82 ± 0,02	99,58 ± 0,04
6500	0,30 ± 0,17	99,34 ± 0,11	99,82 ± 0,02	99,58 ± 0,04
7000	0,32 ± 0,15	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
7500	0,34 ± 0,13	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
8000	0,39 ± 0,16	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
8500	0,49 ± 0,19	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
9000	0,53 ± 0,15	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
9500	0,55 ± 0,13	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03
10000	0,54 ± 0,12	99,32 ± 0,08	99,82 ± 0,02	99,57 ± 0,03

Tabela 30: Terceiro cenário – Base de dados Engine Fuel System
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Engine Fuel System				
Conjunto inicial - 3000 casos				
14 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	0,21 ± 0,23	99,38 ± 0,03	99,89 ± 0,02	99,63 ± 0,03
3500	0,14 ± 0,29	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
4000	0,05 ± 0,01	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
4500	0,05 ± 0,01	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
5000	0,15 ± 0,28	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
5500	0,15 ± 0,28	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
6000	0,15 ± 0,27	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
6500	0,15 ± 0,26	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
7000	0,25 ± 0,36	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
7500	0,08 ± 0,01	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
8000	0,09 ± 0,02	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
8500	0,18 ± 0,26	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
9000	0,34 ± 0,40	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
9500	0,17 ± 0,24	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
10000	0,18 ± 0,25	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
3000	0,28 ± 0,30	99,38 ± 0,03	99,89 ± 0,02	99,63 ± 0,03
3500	0,29 ± 0,30	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
4000	0,23 ± 0,21	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
4500	0,31 ± 0,27	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
5000	0,26 ± 0,20	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
5500	0,33 ± 0,24	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
6000	0,34 ± 0,23	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
6500	0,35 ± 0,21	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
7000	0,36 ± 0,19	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
7500	0,42 ± 0,21	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
8000	0,46 ± 0,19	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
8500	0,50 ± 0,18	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
9000	0,47 ± 0,16	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
9500	0,49 ± 0,15	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03
10000	0,51 ± 0,13	99,38 ± 0,03	99,86 ± 0,03	99,62 ± 0,03

Tabela 31: Quarto cenário – Base de dados Engine Fuel System
Casos: número de instâncias da base de dados; Tempo: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções; DP: Desvio Padrão; MICC(0): Média do Índice de Classificação Correta para a classe 0; MICC(1): Média do Índice de Classificação Correta para a classe 1; MICCT: Média do Índice de Classificação Correta Total (considerando-se as duas classes);

Base Engine Fuel System				
Conjunto inicial - 1000 casos				
18 conjuntos de 500 casos				
	Incremental			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	0,14 ± 0,29	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
1500	0,03 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
2000	0,03 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
2500	0,03 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
3000	0,13 ± 0,29	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
3500	0,22 ± 0,39	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
4000	0,05 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
4500	0,05 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
5000	0,50 ± 0,47	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
5500	0,06 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
6000	0,15 ± 0,28	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
6500	0,24 ± 0,36	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
7000	0,15 ± 0,28	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
7500	0,17 ± 0,27	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
8000	0,08 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
8500	0,08 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
9000	0,09 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
9500	0,09 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
10000	0,17 ± 0,25	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
	Batelada			
Casos	Tempo ± DP	MICC(0) ± DP	MICC(1) ± DP	MICCT ± DP
1000	0,05 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
1500	0,06 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
2000	0,08 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
2500	0,10 ± 0,01	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
3000	0,20 ± 0,24	99,45 ± 0,04	99,80 ± 0,00	99,62 ± 0,02
3500	0,29 ± 0,30	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
4000	0,15 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
4500	0,63 ± 0,31	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
5000	0,25 ± 0,20	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
5500	0,20 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
6000	0,23 ± 0,01	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
6500	0,61 ± 0,24	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
7000	0,55 ± 0,24	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
7500	0,37 ± 0,18	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
8000	0,42 ± 0,18	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
8500	0,40 ± 0,15	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
9000	0,40 ± 0,13	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
9500	0,39 ± 0,09	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03
10000	0,42 ± 0,09	99,45 ± 0,04	99,82 ± 0,02	99,64 ± 0,03

Apêndice II – Tabelas com os resultados do algoritmo ABC

Tabela 32: Cenário Estático Alarm.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Alarm			
Cenário Estático			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	304,45 ± 3,91	43,11 ± 0,62	98,56 ± 0,07
200.000	63,10 ± 0,91	214,30 ± 2,15	98,55 ± 0,01
300.000	63,23 ± 0,95	213,92 ± 1,99	98,57 ± 0,01
400.000	63,37 ± 0,88	214,32 ± 2,39	98,57 ± 0,01
500.000	63,62 ± 0,83	213,98 ± 1,92	98,58 ± 0,01
600.000	63,60 ± 0,79	214,40 ± 2,53	98,58 ± 0,01
700.000	64,00 ± 0,82	214,43 ± 1,70	98,58 ± 0,00
800.000	63,99 ± 0,74	214,27 ± 2,03	98,59 ± 0,00
900.000	64,25 ± 0,67	214,56 ± 2,23	98,60 ± 0,00
1.000.000	64,32 ± 0,64	214,38 ± 2,00	98,59 ± 0,00
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	303,93 ± 2,08	43,00 ± 0,43	98,53 ± 0,05
200.000	471,73 ± 4,51	84,75 ± 0,22	98,56 ± 0,06
300.000	619,27 ± 4,85	127,71 ± 0,88	98,62 ± 0,03
400.000	745,14 ± 2,91	169,44 ± 0,24	98,59 ± 0,03
500.000	852,46 ± 7,32	211,94 ± 0,47	98,58 ± 0,02
600.000	981,28 ± 48,29	258,38 ± 13,81	98,58 ± 0,03
700.000	1.091,79 ± 65,32	297,53 ± 1,61	98,60 ± 0,04
800.000	1.166,36 ± 10,46	339,02 ± 0,73	98,61 ± 0,02
900.000	1.270,68 ± 7,29	382,36 ± 1,87	98,60 ± 0,02
1.000.000	1.363,91 ± 7,36	424,28 ± 1,14	98,59 ± 0,02

Tabela 33: Cenário Dinâmico Alarm.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Alarm			
Cenário Dinâmico			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	308,78 ± 4,44	42,50 ± 0,26	98,49 ± 0,06
200.000	67,31 ± 0,84	212,15 ± 1,68	96,73 ± 0,46
300.000	68,30 ± 0,60	212,64 ± 2,19	96,61 ± 0,49
400.000	68,34 ± 1,68	213,05 ± 2,36	96,52 ± 0,51
500.000	69,02 ± 0,63	212,65 ± 2,40	95,82 ± 0,56
600.000	68,41 ± 1,26	213,10 ± 2,62	94,79 ± 0,71
700.000	68,52 ± 0,70	212,98 ± 2,72	93,15 ± 1,12
800.000	67,40 ± 1,01	213,02 ± 2,48	91,96 ± 1,44
900.000	71,04 ± 4,07	14.804,64 ± 16.138,54	93,75 ± 2,04
1.000.000	69,47 ± 3,62	11.235,66 ± 18.018,03	92,77 ± 0,90
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	309,80 ± 2,78	42,59 ± 0,15	98,50 ± 0,06
200.000	477,95 ± 4,23	82,26 ± 0,16	98,47 ± 0,04
300.000	645,03 ± 6,73	123,80 ± 0,70	98,44 ± 0,04
400.000	795,70 ± 7,82	167,72 ± 0,66	98,49 ± 0,03
500.000	950,15 ± 9,90	217,91 ± 1,09	98,40 ± 0,02
600.000	1.201,54 ± 9,20	295,75 ± 0,14	95,60 ± 0,02
700.000	1.580,76 ± 9,13	323,48 ± 0,73	97,54 ± 0,03
800.000	1.821,20 ± 88,64	379,63 ± 33,18	96,74 ± 0,03
900.000	2.078,11 ± 52,08	444,43 ± 48,23	96,37 ± 0,02
1.000.000	2.458,60 ± 72,83	531,94 ± 1,69	92,99 ± 0,04

Tabela 34: Cenário Estático Ásia.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Asia			
Cenário Estático			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	54,65 ± 7,65	1,92 ± 0,16	66,83 ± 0,19
200.000	22,39 ± 4,16	9,72 ± 1,32	66,68 ± 0,03
300.000	22,91 ± 4,44	9,96 ± 1,73	66,69 ± 0,02
400.000	22,63 ± 4,06	10,10 ± 1,61	66,62 ± 0,01
500.000	23,07 ± 4,47	10,28 ± 1,99	66,56 ± 0,01
600.000	23,03 ± 4,63	10,36 ± 2,10	66,60 ± 0,01
700.000	23,12 ± 4,52	10,41 ± 2,34	66,59 ± 0,01
800.000	23,57 ± 4,15	10,43 ± 1,81	66,55 ± 0,01
900.000	24,27 ± 4,59	9,86 ± 0,89	66,55 ± 0,00
1.000.000	24,17 ± 4,32	10,08 ± 0,92	66,54 ± 0,00
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	29,06 ± 1,89	3,60 ± 0,09	66,80 ± 0,31
200.000	61,89 ± 3,42	7,18 ± 0,09	66,94 ± 0,21
300.000	95,04 ± 3,78	10,70 ± 0,22	66,76 ± 0,15
400.000	127,04 ± 3,87	14,13 ± 0,36	66,80 ± 0,09
500.000	159,12 ± 3,72	18,20 ± 0,44	66,76 ± 0,09
600.000	191,75 ± 5,73	22,15 ± 0,71	66,71 ± 0,11
700.000	225,20 ± 6,54	25,77 ± 0,54	66,77 ± 0,11
800.000	246,91 ± 16,48	29,55 ± 0,37	66,65 ± 0,05
900.000	287,90 ± 11,21	33,39 ± 0,93	66,65 ± 0,06
1.000.000	324,70 ± 8,49	37,14 ± 0,93	66,66 ± 0,07

Tabela 35: Cenário Dinâmico Ásia.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Asia			
Cenário Dinâmico			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	52,04 ± 0,73	1,82 ± 0,05	67,03 ± 0,23
200.000	23,12 ± 0,22	8,83 ± 0,04	55,54 ± 0,04
300.000	23,36 ± 0,24	9,55 ± 0,11	59,23 ± 0,02
400.000	23,71 ± 0,34	9,92 ± 0,12	66,11 ± 0,01
500.000	24,03 ± 0,21	9,78 ± 0,09	68,01 ± 0,01
600.000	24,11 ± 0,26	9,97 ± 0,12	69,21 ± 0,01
700.000	24,24 ± 0,34	9,80 ± 0,12	70,05 ± 0,01
800.000	24,41 ± 0,22	9,14 ± 0,06	70,65 ± 0,01
900.000	24,38 ± 0,20	9,94 ± 0,10	71,12 ± 0,01
1.000.000	24,51 ± 0,35	8,83 ± 0,09	71,49 ± 0,01
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	27,64 ± 2,24	3,53 ± 0,10	66,76 ± 0,27
200.000	59,89 ± 5,22	7,20 ± 0,67	66,86 ± 0,14
300.000	93,34 ± 13,42	11,59 ± 0,22	54,55 ± 0,06
400.000	124,91 ± 13,22	15,89 ± 1,18	54,48 ± 0,05
500.000	164,97 ± 44,75	20,73 ± 0,38	54,54 ± 0,05
600.000	184,18 ± 11,07	25,23 ± 0,69	57,05 ± 0,07
700.000	216,55 ± 22,01	33,95 ± 5,24	58,86 ± 0,07
800.000	249,07 ± 29,57	37,37 ± 0,90	61,68 ± 0,06
900.000	280,40 ± 28,49	46,46 ± 2,35	65,36 ± 0,09
1.000.000	312,70 ± 41,82	61,69 ± 21,26	68,27 ± 0,04

Tabela 36: Cenário Estático Credit.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Credit			
Cenário Estático			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	54,82 ± 0,91	3,45 ± 0,06	72,28 ± 0,23
200.000	31,08 ± 0,69	17,10 ± 0,18	72,23 ± 0,03
300.000	30,92 ± 0,88	17,12 ± 0,17	72,24 ± 0,02
400.000	31,19 ± 0,86	17,17 ± 0,17	72,15 ± 0,03
500.000	31,07 ± 0,82	17,20 ± 0,17	72,12 ± 0,02
600.000	31,14 ± 0,83	17,23 ± 0,17	72,16 ± 0,01
700.000	31,65 ± 1,74	17,37 ± 0,42	72,19 ± 0,02
800.000	32,55 ± 3,80	17,34 ± 0,25	72,17 ± 0,01
900.000	32,43 ± 3,27	17,43 ± 0,41	72,17 ± 0,01
1.000.000	31,58 ± 0,61	17,32 ± 0,17	72,17 ± 0,01
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	33,27 ± 1,51	7,31 ± 0,11	72,21 ± 0,13
200.000	67,67 ± 0,98	14,69 ± 0,24	72,09 ± 0,15
300.000	101,49 ± 0,93	22,04 ± 0,21	72,19 ± 0,14
400.000	134,82 ± 1,33	29,30 ± 0,43	72,35 ± 0,12
500.000	167,28 ± 1,72	37,17 ± 0,57	72,19 ± 0,14
600.000	200,26 ± 3,40	43,85 ± 0,63	72,27 ± 0,07
700.000	233,87 ± 2,87	51,25 ± 0,59	72,24 ± 0,11
800.000	265,61 ± 4,83	58,51 ± 0,70	72,11 ± 0,07
900.000	299,45 ± 5,87	65,88 ± 0,85	72,12 ± 0,06
1.000.000	330,83 ± 7,34	73,17 ± 0,91	72,11 ± 0,07

Tabela 37: Cenário Dinâmico Credit.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Credit			
Cenário Dinâmico			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	57,92 ± 5,29	3,48 ± 0,09	72,25 ± 0,19
200.000	34,58 ± 3,89	111,93 ± 101,82	70,84 ± 0,09
300.000	34,31 ± 3,44	17,69 ± 2,63	70,62 ± 0,17
400.000	34,81 ± 4,36	16,79 ± 0,79	70,46 ± 0,17
500.000	47,57 ± 3,58	870,11 ± 88,88	72,11 ± 0,04
600.000	38,85 ± 4,21	17,30 ± 0,48	71,72 ± 0,03
700.000	39,13 ± 5,30	17,32 ± 0,47	71,81 ± 0,02
800.000	39,05 ± 4,31	17,39 ± 0,58	71,03 ± 0,02
900.000	39,40 ± 5,23	17,60 ± 1,12	70,76 ± 0,02
1.000.000	39,36 ± 4,74	17,46 ± 0,64	70,96 ± 0,01
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	34,43 ± 1,57	7,58 ± 0,10	72,32 ± 0,19
200.000	70,12 ± 1,17	15,31 ± 0,46	72,03 ± 0,16
300.000	105,82 ± 1,87	22,76 ± 0,39	71,16 ± 0,13
400.000	140,64 ± 2,14	30,69 ± 0,52	71,65 ± 0,07
500.000	173,70 ± 2,11	38,11 ± 0,48	71,47 ± 0,08
600.000	208,62 ± 2,16	45,87 ± 0,50	71,34 ± 0,10
700.000	240,87 ± 9,14	62,26 ± 2,44	71,31 ± 0,07
800.000	275,70 ± 6,05	76,81 ± 1,18	71,87 ± 0,09
900.000	311,98 ± 6,56	99,84 ± 1,17	71,85 ± 0,07
1.000.000	341,82 ± 8,88	117,54 ± 1,54	72,31 ± 0,08

Tabela 38: Cenário Estático Engine Fuel System.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Engine Fuel System			
Cenário Estático			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	48,33 ± 0,47	2,27 ± 0,05	99,96 ± 0,02
200.000	19,98 ± 0,52	11,15 ± 0,12	99,97 ± 0,00
300.000	20,03 ± 0,77	11,17 ± 0,11	99,97 ± 0,00
400.000	20,18 ± 0,72	11,18 ± 0,11	99,97 ± 0,00
500.000	20,30 ± 0,74	11,28 ± 0,17	99,97 ± 0,00
600.000	20,34 ± 0,77	11,40 ± 0,52	99,97 ± 0,00
700.000	20,47 ± 0,73	11,28 ± 0,13	99,97 ± 0,00
800.000	20,48 ± 0,74	11,31 ± 0,13	99,97 ± 0,00
900.000	20,58 ± 0,75	11,31 ± 0,13	99,97 ± 0,00
1.000.000	20,63 ± 0,72	11,51 ± 0,40	99,97 ± 0,00
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	27,50 ± 2,95	4,64 ± 0,85	99,96 ± 0,01
200.000	58,52 ± 7,03	9,46 ± 1,80	99,96 ± 0,00
300.000	90,31 ± 10,53	13,94 ± 2,47	99,97 ± 0,01
400.000	122,55 ± 14,74	18,42 ± 3,37	99,97 ± 0,01
500.000	154,03 ± 18,67	23,16 ± 4,20	99,97 ± 0,00
600.000	185,35 ± 22,49	27,79 ± 5,12	99,97 ± 0,00
700.000	216,62 ± 29,01	32,82 ± 6,22	99,97 ± 0,00
800.000	245,99 ± 33,19	36,95 ± 6,62	99,97 ± 0,00
900.000	270,88 ± 34,34	41,91 ± 7,58	99,97 ± 0,00
1.000.000	306,16 ± 36,14	46,46 ± 8,52	99,97 ± 0,00

Tabela 39: Cenário Dinâmico Engine Fuel System.
Base de dados: número de registros da base de dados; Tapr: tempo médio, em segundos (s), de aprendizado considerando as 10 execuções, sem o tempo relativo à classificação; Tclass: tempo médio, em segundos (s), de classificação considerando as 10 execuções; MICCT: Média do Índice de Classificação Correta Total (considerando-se todas as classes); DP: Desvio Padrão.

Base de dados Engine Fuel System			
Cenário Dinâmico			
	ABC		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	50,28 ± 0,42	2,26 ± 0,03	99,95 ± 0,01
200.000	21,28 ± 0,70	11,27 ± 0,16	99,92 ± 0,00
300.000	21,62 ± 0,94	11,19 ± 0,12	99,90 ± 0,00
400.000	21,71 ± 0,90	11,25 ± 0,10	99,88 ± 0,00
500.000	21,79 ± 0,90	11,28 ± 0,11	99,86 ± 0,00
600.000	21,94 ± 0,95	11,28 ± 0,12	99,84 ± 0,00
700.000	22,03 ± 0,90	11,48 ± 0,51	99,82 ± 0,00
800.000	22,11 ± 0,74	11,51 ± 0,50	99,75 ± 0,00
900.000	22,18 ± 1,00	11,35 ± 0,14	99,69 ± 0,00
1.000.000	22,17 ± 1,01	11,42 ± 0,14	99,63 ± 0,00
	K2		
Base de dados	Tapr ± DP	Tclass ± DP	MICCT ± DP
100.000	19,00 ± 0,47	2,25 ± 0,06	99,96 ± 0,01
200.000	39,36 ± 0,45	4,50 ± 0,08	99,95 ± 0,01
300.000	60,13 ± 0,46	6,71 ± 0,09	99,95 ± 0,01
400.000	80,51 ± 0,57	8,95 ± 0,12	99,95 ± 0,01
500.000	100,74 ± 0,19	11,98 ± 0,15	99,95 ± 0,00
600.000	121,18 ± 0,61	14,31 ± 0,15	99,94 ± 0,00
700.000	140,62 ± 1,43	16,67 ± 0,12	99,94 ± 0,00
800.000	161,61 ± 1,32	23,01 ± 0,53	99,91 ± 0,00
900.000	180,58 ± 5,24	28,86 ± 1,47	99,89 ± 0,01
1.000.000	200,88 ± 1,20	35,14 ± 1,22	99,86 ± 0,00